

MALLOW 2010

Proceedings of The Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)

Lyon, France, from the 30th of August to the 2nd of September, 2010.

ORGANIZATION COMMITTEE

Organization chairs

- Olivier Boissier, ENSM, Saint-Etienne, France
- Amal El Fallah Seghrouchni, LIP6, Paris, France
- Salima Hassas, LIESP, Lyon, France
- Nicolas Maudet, LAMSADE, Paris, France

Organization committee

- Samir Aknine, LIESP, Lyon, France
- Aurélie Bénier, LIP6, Paris, France
- Fabien Badeig, LAMSADE, Paris, France
- Flavien Balbo, LAMSADE, Paris, France
- Zahia Guessoum, LIP6, Paris, France
- Gauthier Picard, ENSM, Saint-Etienne, France
- Laurent Vercouter, ENSM, Saint-Etienne, France
- Caroline Wintergerst, IAE Lyon 3, France

Steering Committee

- Cristina Baroglio, University of Torino, Italy
- Rafael H. Bordini (chair), INF-UFRGS, Brazil
- Mehdi Dastani, Utrecht Univeristy, Netherlands
- Virginia Dignum, TU Delft, Netherlands
- João Leite, Universidade Nova de Lisboa, Portugal
- John Lloyd, Australian National University, Australia
- Brian Logan, University of Nottingham, UK
- Pablo Noriega, IIIA-CSIC, Spain
- Munindar Singh, NCSU, USA
- Rineke Verbrugge, University of Groningen, Netherlands

PREFACE

These informal proceedings contain the contributions presented during MALLOW-2010 federation of workshops at Domaine Valpré near Lyon. MALLOW stands for "Multi-Agent Logics, Languages, and Organizations". This year the event federates 5 workshops:

- The Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN)
- The Workshop of the FIPA Design Process Documentation and Fragmentation Working Group (DPDF WG)
- The Workshop on LAnguages, methodologies and Development tools for multi-agent systems (LADS)
- The Workshop on Logics for Resource Bounded Agents (LRBA)
- The Workshop on Multi-Agent Systems and Simulation (MASS)

The contributed papers of each workshop have been selected independently by the corresponding scientific committees, for a grand total of 42 papers. We believe it provides a nice overview of the current research on some of the hot topics in multiagent systems.

In addition to these contributions, invited lectures by Bruce Edmonds (Manchester Metropolitan University Business School, UK), Andreas Herzig (IRIT, Toulouse, FR) and David Sadek (Orange, FR), are scheduled this year.

A panel organized by Alessandro Ricci (DEIS, Bologna, Italy) offers the opportunity for the participants to share and discuss their points of view on "Agent and Multi-agent Thinking in Mainstream Computer Science".

We take the opportunity to thank these invited speakers, the panel organizer and the participants for their contribution.

MALLOW-2010 is a third edition of a series initiated in 2007 in Durham, and pursued in 2009 in Turin. The objective, as initially stated, is to "provide a venue where: the cost of participation was minimum; participants were able to attend various workshops, so fostering collaboration and cross-fertilization; there was a friendly atmosphere and plenty of time for networking, by maximizing the time participants spent together".

We would like to thank the Valpré Domain for offering the accommodation infrastructure that strongly contributed this year to build this friendly atmosphere.

This MALLOW-2010 edition has been organized by four groups of the French community, developing researches on multi-agent systems: LAMSADE (Paris), LIESP (Lyon), LIP6 (Paris) and G2I/LSTI (Saint-Etienne).

The success of these first editions, with an attendance of 80-100 participants, is a clear encouragement to organize a follow-up. At the time of writing, the available figures indicate a similar attendance for this year, with participants coming from large variety of countries (e.g. Brazil, France, Germany, Italy, Japan, Netherlands, and United Kingdom).

Like in previous editions, the collocation of MALLOW with the European Summer School already proved to be beneficial for many young researchers. Indeed, about 25% of participants registered to attend both events and enjoy this unique opportunity to get in two weeks the background picture of the multiagent field (as provided by EASSS tutors), and some of the latest developments as discussed during MALLOW workshops.

Finally, a key to the success of MALLOW events is to offer an affordable price for the whole event. This year this was made possible thanks to the support of various partners that we wish to thank again here: AFIA (Artificial Intelligence French Association), FIPA, COST Agreement Technologies IC0801, ISLE, IXXI (Complex Systems Institute Rhône-Alpes), UPETEC (Emergence Technologies Self-Adaptive Software), Région Rhône-Alpes, Rhône Le Département, LIESP Laboratory, LAMSADE, LIP6, ENSM Saint-Etienne.



Nicoletta Fornara, George Vouros (eds.)

11th International Workshop on
Coordination, Organization,
Institutions and Norms
in Agent Systems

Lyon, France,
30th August - 2nd September 2010

Workshop Notes

Preface

The development of complex distributed AI systems with heterogeneous and diverse knowledge is a challenge. System components must interact, coordinate and collaborate to manage scale and complexity of task environments targeting persistency and maybe, evolution of systems. Managing scale and complexity requires organized intelligence; in particular intelligence manifested in organizations of agents, by individual strategies or collective behaviour. System architects have to consider: the inter-operation of heterogeneously designed, developed or discovered components (agents, objects/artefacts, services provided in an open environment); inter-connection which cross legal, temporal, or organizational boundaries; the absence of global objects or centralised controllers; the possibility that components will not comply with the given specifications; and embedding in an environment which is likely to change, with possible impact on individual and collective objectives.

The convergence of the requirement for intelligence with these operational constraints demands: coordination, the collective ability of heterogeneous and autonomous components to arrange or synchronise the performance of specified actions in sequential or temporal order; rational and open organization, a formal structure supporting or producing intentional forms of coordination, capable of managing changes in the environment in which it operates; institution, an organization where the performance of designated actions by empowered agents produces conventional outcomes; and norms, standards or patterns of behaviour in an institution established by decree, agreement, emergence, and so on.

The automation and distribution of intelligence is the subject of study in autonomous agents and multi-agent systems; the automation and distribution of intelligence for coordination, organization, institutions and norms is the interest of this workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN), in its eleventh edition. The COIN@MALLOW 2010 workshop is part of the COIN series of workshops <http://www.pcs.usp.br/coin/>.

This edition of COIN received fourteen high quality submissions, describing works by researchers coming from nine different countries, eight of which have been selected by the Programme Committee as regular papers and two of which have been selected by the Programme Committee as position papers. Each paper received at least three reviews in order to supply the authors with helpful feedback that could stimulate the research as well as foster discussion. COIN@AAMAS2010 and COIN@MALLOW2010 post-proceedings will be published soon in a single Springer LNCS volume.

We would like to thank all authors for their contributions, the members of the Steering Committee for the valuable suggestions and support, and the members of the Programme Committee for their excellent work during the reviewing phase.

August 4th, 2010

Nicoletta Fornara, George Vouros

Workshop Organisers

Nicoletta Fornara University of Lugano, Switzerland
George Vouros University of the Aegean, Greece

Programme Committee

Alexander Artikis National Centre for Scientific Research Demokritos, Greece
Guido Boella University of Torino, Italy
Olivier Boissier ENS Mines Saint-Etienne, France
Rafael Bordini Federal University of Rio Grande do Sul, Brazil
Amit Chopra University of Trento, Italy
Antonio Carlos da Rocha Costa Univ. Federal do Rio Grande FURG, Brazil
Marina De Vos University of Bath, UK
Virginia Dignum Delft University of Technology, The Netherlands
Jomi Fred Hubner Federal University of Santa Catarina, Brazil
Christian Lemaitre Universidad Autonoma Metropolitana, Mexico
Henrique Lopes Cardoso Universidade do Porto, Portugal
Eric Matsou Purdue, USA
John-Jules Meyer Utrecht University, The Netherlands
Pablo Noriega IIIA-CSI, Spain
Eugenio Oliveira Universidade do Porto, Portugal
Andrea Omicini University of Bologna, Italy
Sascha Ossowski URJC, Spain
Julian Padget University of Bath, UK
Jeremy Pitt Imperial College, London, UK
Juan Antonio Rodriguez Aguilar IIIA-CSIC, Spain
Jaime Sichman University of Sao Paulo, Brazil
Munindar P. Singh North Carolina State University, USA
Viviane Torres da Silva Universidade Federal Fluminense, Brazil
Kostas Stathis Royal Holloway, University of London, UK
Paolo Torroni University of Bologna, Italy
Leon van der Torre University of Luxembourg, Luxembourg
Birna van Riemsdijk Delf University of Technology, The Netherlands
Wamberto Vasconcelos University of Aberdeen, UK
Javier Vazquez-Salceda University Politecnica de Catalunya, Spain
Mario Verdicchio University of Bergamo, Italy
Danny Weyns Katholieke Universiteit Leuven, Germany
Pinar Yolum Bogazici University, Turkey

Additional Reviewers

Luciano Coutinho	Universidade de Campinas, Brazil
Akin Gunay	Bogazici University, Turkey
Ozgur Kafali	Bogazici University, Turkey

Steering Committee

Guido Boella	University of Torino, Italy
Olivier Boissier	ENS Mines Saint-Etienne, France
Nicoletta Fornara	University of Lugano, Switzerland
Christian Lemaitre	Universidad Autonoma Metropolitana, Mexico
Eric Matson	Purdue University, USA
Pablo Noriega	Artificial Intelligence Research Institute, Spain
Sascha Ossowski	Universidad Rey Juan Carlos, Spain
Julian Padget	University of Bath, UK
Jeremy Pitt	Imperial College London, UK
Jaime Sichman	University of Sao Paulo, Brazil
Wamberto Vasconcelos	University of Aberdeen, UK
Javier Vzquez Salceda	Universitat Politecnica de Catalunya, Spain
George Vouros	University of the Aegean, Greece

Table of Contents

Normative Monitoring: Semantics and Implementation	1
<i>Sergio Alvarez-Napagao, Huib Aldewereld, Javier Vazquez, Frank Dignum</i>	
Controlling multi-party interaction within normative multi-agent organizations	17
<i>Olivier Boissier, Flavien Balbo, Fabien Badeig</i>	
Norm Refinement and Design through Inductive Learning	33
<i>Domenico Corapi, Marina De Vos, Julian Padget, Alessandra Russo, Ken Satoh</i>	
Norm enforceability in Electronic Institutions?	49
<i>Natalia Criado, Estefania Argente, Antonio Garrido, Juan A. Gimeno, Francesc Igual, Vicente Botti, Pablo Noriega, Adriana Giret</i>	
Towards a Normative BDI Architecture for Norm Compliance	65
<i>Natalia Criado, Estefania Argente, Pablo Noriega, Vicent Botti</i>	
Generating Executable MAS-Prototypes from SONAR Specifications	82
<i>Michael Köhler-Bußmeier, Matthias Wester-Ebbinghaus, Daniel Moldt</i>	
Embodied Organizations: a unifying perspective in programming Agents, Organizations and Environments	98
<i>Michele Piunti, Olivier Boissier, Jomi F. Hüubner, Alessandro Ricci</i>	
Group intention = social choice + commitment	115
<i>Marija Slavkovic, Guido Boella, Gabriella Pigozzi, Leon van der Torre</i>	
Position Papers	
MERCURIO: An Interaction-oriented Framework for Designing, Verifying and Programming Multi-Agent Systems	134
<i>Matteo Baldoni, Cristina Baroglio, Federico Bergenti, Antonio Boccialatte, Elisa Marengo, Maurizio Martelli, Viviana Mascardi, Luca Padovani, Viviana Patti, Alessandro Ricci, Gianfranco Rossi, Andrea Santi</i>	
Contextual Integrity and Privacy Enforcing Norms for Virtual Communities	150
<i>Yann Krupa, Laurent Vercouter</i>	

Normative Monitoring: Semantics and Implementation

Sergio Alvarez-Napagao¹, Huib Aldewereld²,
Javier Vázquez-Salceda¹, and Frank Dignum²

¹ Universitat Politècnica de Catalunya

{salvarez,jvazquez}@lsi.upc.edu

² Universiteit Utrecht

{huib,dignum}@cs.uu.nl

Abstract. The concept of Normative Systems can be used in the scope of Multi-Agent Systems to provide reliable contexts of interactions between agents where acceptable behaviour is specified in terms of norms. Literature on the topic is growing rapidly, and there is a considerable amount of theoretical frameworks for normative environments, some in the form of Electronic Institutions. Most of these approaches focus on regulative norms rather than on substantive norms, and lack a proper implementation of the ontological connection between brute events and institutional facts. In this paper we present a formalism for the monitoring of both regulative (deontic) and substantive (constitutive) norms based on Structural Operational Semantics, its reduction to Production Systems semantics and our current implementation compliant to these semantics.

1 Introduction

In recent years, several researchers have argued that the design of multi-agent systems (MAS) in complex, open environments can benefit from social abstractions in order to cope with problems in coordination, cooperation and trust among agents, problems which are also present in human societies. One of such abstractions is *Normative Systems*. Research in Normative Systems focuses on the concepts of norms and normative environment (which some authors refer to as *institutions*) in order to provide normative frameworks to restrict or guide the behaviour of (software) agents. The main idea is that the interactions among a group of (software) agents are ruled by a set of explicit norms expressed in a computational language representation that agents can interpret. Although some authors only see norms as inflexible restrictions to agent behaviour, others see norms not as a negative, constraining factor but as an aid that guides the agents' choices and reduces the complexity of the environment, making the behaviour of other agents more predictable.

Until recently, most of the work on normative environments works with norm specifications that are static and stable, and which will not change over time. Although this may be good enough from the social (institutional) perspective, it is not appropriate from the agent perspective. During their lifetime, agents may enter and leave several interaction contexts, each with its own normative framework. Furthermore they may be operating in contexts where more than one normative specification applies. So we need

mechanisms where normative specifications can be added to the agents' knowledge base at run-time and be practically used in their reasoning, both to be able to interpret institutional facts from brute ones (by using constitutive norms to, e.g. decide if killing a person counts as *murder* in the current context) and to decide what ought to be done (by using regulative norms to, e.g. prosecute the murderer). In this paper we propose to use production systems to build a norm monitoring mechanism that can be used both by agents to perceive the current normative state of their environment, and for these environments to detect norm violations and enforce sanctions. Our basic idea is that an agent can configure, at a practical level, the production system at run-time by adding abstract organisational specifications and sets of counts-as rules.

In our approach, the detection of normative states is a passive procedure consisting in monitoring past events and checking them against a set of active norms. This type of reasoning is already covered by the declarative aspect of production systems, so no additional implementation in an imperative language is needed. Using a forward-chaining rule engine, events will automatically trigger the normative state - based on the operational semantics - without requiring a design on *how* to do it.

Having 1) a direct syntactic translation from norms to rules and 2) a logic implemented in an engine consistent with the process we want to accomplish, allows us to decouple normative state monitoring from the agent reasoning. The initial set of rules we have defined is the same for each type of agent and each type of organisation, and the agent will be able to transparently query the current normative state at any moment and reason upon it. Also this decoupling helps building third party/facilitator agents capable of observing, monitoring and reporting normative state change or even enforcing behaviour in the organisation.

In this paper we present a formalism for the monitoring of both regulative (deontic) and substantive (constitutive) norms based on Structural Operational Semantics (Section 2), its reduction to Production Systems semantics (Section 3) and our current implementation compliant to these semantics (Section 4). In Section 5 we compare with other related work and provide some conclusions.

2 Formal Semantics

In this section we discuss the formal semantics of our framework. First, in section 2.1, we give the semantics of institutions as the environment specifying the regulative and constitutive norms. Then, in section 2.2, we describe the details of how this institution evolves over time based on events, and how this impacts the monitoring process. This formalisation will be used in section 3 as basis of our implementation.

Through this paper, we will use as an example the following simplified traffic scenario:

1. A person driving on a street is not allowed to break a traffic convention.
2. In case (1) is violated, the driver must pay a fine.
3. In a city, to exceed 50kmh counts as breaking a traffic convention.

2.1 Preliminary definitions

Before giving a formal definition of institutions (see definition 4), we first define the semantics of the regulative and constitutive norms part of that institution (in definitions 1 and 3, respectively).

We assume the use of a predicate based propositional logic language \mathcal{L}_O with predicates and constants taken from an ontology O , and the logical connectives $\{\neg, \vee, \wedge\}$. The set of all possible well-formed formulas of \mathcal{L}_O is denoted as $wff(\mathcal{L}_O)$ and we assume that each formula from $wff(\mathcal{L}_O)$ is normalised in Disjunctive Normal Form (DNF). Formulas in $wff(\mathcal{L}_O)$ can be partially grounded, if they use at least one free variable, or fully grounded if they use no free variables.

In this paper we intensively use the concept of variable substitution. We define a substitution instance $\Theta = \{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \dots, x_i \leftarrow t_i\}$ as the substitution of the terms t_1, t_2, \dots, t_i for variables x_1, x_2, \dots, x_i in a formula $f \in wff(\mathcal{L}_O)$.

We denote the set of roles in a normative system as the set of constants R , where $R \subset O$, and the set of participants as P , where each participant enacts at least one role according to the ontology O .

As our aim is to build a normative monitoring mechanism that can work at real time, special care has been made to choose a norm language which, without loss of expressiveness, has operational semantics that can then be mapped into production systems. Based in our previous work and experience, our definition of *norm* in an extension of the norm language presented in [12]:

Definition 1. A norm n is a tuple $n = \langle f_A, f_M, f_\delta, f_D, f_w, w \rangle$, where

- $f_A, f_M, f_\delta, f_D, f_w \in wff(\mathcal{L}_O)$, $w \in R$,
- f_A, f_M, f_D respectively represent the activation, maintenance, and deactivation conditions of the norm, f_δ, f_w are the explicit representation of the deadline and target of the norm, and
- w is the subject of the norm.

In order to create an optimal norm monitor it is important to know which norms are active at each point in time, as only those are the ones that have to be traced (inactive norms can be discarded from the monitoring process until they become active again). The *activation condition* f_A specifies when a norm becomes active. It is also the main element in the norm instantiation process: when the conditions in the activating condition hold, the variables are instantiated, creating a new norm instance³ The *target condition* f_w describes the state that fulfills the norm (e.g. if one is obliged to pay, the payment being made fulfills the obligation). The *deactivating condition* f_D defines when the norm becomes inactive. Typically it corresponds to the *target condition* (e.g., fulfilling the norm instance deactivates that instance of the norm), but in some cases it also adds conditions to express other deactivating scenarios (e.g., when the norm becomes deprecated). The *maintenance condition* f_M defines the conditions that, when

³ One main differentiating aspect of our formalisation is that we include variables in the norm representation and we can handle multiple instantiations of the same norm and track them separately.

no longer hold, lead to a violation of the norm. Finally the *deadline* condition f_δ represents one or several deadlines for the norm to be fulfilled.

An example of a norm for the traffic scenario ("A person driving on a street is not allowed to break a traffic convention") would be formalised as follows

$$\begin{aligned} \mathbf{n1} := & \langle \text{enacts}(X, \text{Driver}) \wedge \text{is_driving}(X), \\ & \neg \text{traffic_violation}(X), \\ & \perp, \\ & \neg \text{is_driving}(X), \\ & \text{is_driving}(X) \wedge \neg \text{traffic_violation}(X), \\ & \text{Driver} \rangle, \end{aligned}$$

The activating condition states that each time an event appears where an individual enacting the *Driver* role drives (*is_driving*), then a new instance of the norm becomes active; the maintenance condition states that the norm will not be violated while no traffic convention is violated; this norm has no deadline, it is to apply at all times an individual is driving; the norm instance deactivates when the individual stops driving⁴; the target of this norm is that we want drivers not breaking traffic conventions; finally the subject of the norm is someone enacting the *Driver* role.

It is important to note here that, although our norm representation does not explicitly include deontic operators, the combination of the activation, deactivation and maintenance conditions is as expressive as conditional deontic statements with deadlines as the ones in [3]. It is also able to express unconditional norms and maintenance obligations (i.e. the obligation to keep some conditions holding for a period of time). To show that our representation can be mapped to conditional deontic representations, let us express the semantics of the norm in definition 1 in terms of conditional deontic statements. Given relations between the deadline and maintenance condition (that is, $f_\delta \rightarrow \neg f_M$, since the maintenance condition expresses more than the deadline alone) and between the target and the deactivation condition (i.e., $f_w \rightarrow f_D$, since the deactivation condition specifies that either the norm is fulfilled or something special has happened), we can formalise the norms of definition 1 as the equivalent deontic expression (using the formalism of [3]): $f_A \rightarrow [O_w(E_w f_w \leq \neg f_M) \cup f_D]$, where $E_a p$ means that agent a sees to it that (*stit*) p becomes true and \cup is the CTL* until operator. Intuitively, the expression states that after the norm activation, the subject is obliged to see to it that the target becomes true before the maintenance condition is negated (either the deadline is reached or some other condition is broken) until the norm is deactivated (which is either when the norm is fulfilled or has otherwise expired).

Since we are not reasoning about the (effects of) combinations of norms, we will not go into further semantical details here. The semantics presented in this deontic reduction are enough for understanding the monitoring process that is detailed in the remainder of the paper.

A set of norms is denoted as N . We define as *violation handling norms* those norms that are activated automatically by the violation of another norm:

⁴ Although the norm is to apply at all times an individual is driving, it is better to deactivate the norm each time the individual stops driving, instead to keep it active, to minimize the number of norm instances the monitor needs to keep track at all times.

Definition 2. A norm $n' = \langle f'_A, f'_M, f'_\delta, f'_D, f'_w, w' \rangle$ is a violation handling norm of $n = \langle f_A, f_M, f_\delta, f_D, f_w, w \rangle$, denoted as $n \rightsquigarrow n'$ iff $f_A \wedge \neg f_M \wedge \neg f_D \equiv f'_A$

Violation handling norms are special in the sense that they are only activated when another norm is violated. They are used as *sanctioning norms*, if they are to be fulfilled by the norm violating actor (e.g., the obligation to pay a fine if the driver broke a traffic sign), or as *reparation norms*, if they are to be fulfilled by an institutional actor (e.g. the obligation of the authorities to fix the broken traffic sign).

A norm is defined in an abstract manner, affecting all possible participants enacting a given role. Whenever a norm is active, we will say that there is a *norm instance* $ni = \langle n, \theta \rangle$ for a particular norm n and a substitution instance θ .

We define the *state of the world* s_t at a specific point of time t as the set of predicates holding at that specific moment, where $s_t \subseteq O$, and we will denote S as the set of all possible states of the world, where $S = \mathcal{P}(O)$. We will call *expansion* $F(s)$ of a state of the world s as the minimal subset of $wff(\mathcal{L}_O)$ that uses the predicates in s in combination of the logical connectives $\{\neg, \vee, \wedge\}$.

One common problem for the monitoring of normative states is the need for an interpretation of brute events as institutional facts, also called constitution of social reality[8]. The use of *counts-as rules* helps solving this problem. Counts-as rules are multi-modal statements of the form $[c](\gamma_1 \rightarrow \gamma_2)$, read as “in context c , γ_1 counts-as γ_2 ”. In this paper, we will consider a context as a set of predicates, that is, as a possible subset of a state of the world:

Definition 3. A counts-as rule is a tuple $c = \langle \gamma_1, \gamma_2, s \rangle$, where $\gamma_1, \gamma_2 \in wff(\mathcal{L}_O)$, and $s \subseteq O$.

A set of counts-as rules is denoted as C . Although the definition of counts-as in [8] assumes that both γ_1 and γ_2 can be any possible formula, in our work we limit γ_2 to a conjunction of predicates for practical purposes.

Definition 4. Following the definitions above, we define an institution as a tuple of norms, roles, participants, counts-as rules, and an ontology:

$$I = \langle N, R, P, C, O \rangle$$

An example of I for the traffic scenario would be formalised as follows:

$$\begin{aligned} \mathbf{N} &:= \{ \langle enacts(X, Driver) \wedge is_driving(X), \\ &\quad \neg traffic_violation(X), \perp, \neg is_driving(X), \\ &\quad is_driving(X) \wedge \neg traffic_violation(X), Driver \rangle, \\ &\quad \langle enacts(X, Driver) \wedge is_driving(X) \wedge traffic_violation(X), \\ &\quad \top, \\ &\quad paid_fine(X), Driver \rangle \} \\ \mathbf{R} &:= \{ Driver \}, \mathbf{P} := \{ Person_1 \} \\ \mathbf{C} &:= \{ \langle exceeds(D, 50), traffic_violation(D), is_in_city(D) \rangle \} \\ \mathbf{O} &:= \{ role, enacts, is_driving, is_in_city, \\ &\quad exceeds, traffic_violation, is_driving, paid_fine, \\ &\quad Person_1, role(Driver), enacts(Person_1, Driver) \} \end{aligned}$$

2.2 Normative Monitor

In this section we present a formalisation of normative monitoring based on Structural Operational Semantics.

From the definitions introduced in section 2.1, a *Normative Monitor* will be composed of the institutional specification, including norms, the current state of the world, and the current normative state.

In order to track the normative state of an institution at any given point of time, we will define three sets: an instantiation set IS , a fulfillment set FS , and a violation set VS , each of them containing norm instances $\{\langle n_i, \Theta_j \rangle, \langle n_{i'}, \Theta_{j'} \rangle, \dots, \langle n_{i''}, \Theta_{j''} \rangle\}$. We adapt the semantics for normative states from [11]:

Definition 5. *Norm Lifecycle:* Let $ni = \langle n, \Theta \rangle$ be a norm instance, where $n = \langle f_A, f_M, f_D, w \rangle$, and a state of the world s with an expansion $F(s)$. The lifecycle for norm instance ni is defined by the following normative state predicates:

$$\begin{aligned} \text{activated}(ni) &\Leftrightarrow \exists f \in F(s), \Theta(f_A) \equiv f \\ \text{maintained}(ni) &\Leftrightarrow \exists \Theta', \exists f \in F(s), \Theta'(f_M) \equiv f \wedge \Theta' \subseteq \Theta \\ \text{deactivated}(ni) &\Leftrightarrow \exists \Theta', \exists f \in F(s), \Theta'(f_D) \equiv f \wedge \Theta' \subseteq \Theta \\ \text{instantiated}(ni) &\Leftrightarrow ni \in IS \\ \text{violated}(ni) &\Leftrightarrow ni \in VS \\ \text{fulfilled}(ni) &\Leftrightarrow ni \in FS \end{aligned}$$

where IS is the instantiation set, FS is the fulfillment set, and VS is the violation set, as defined above.

For instance, for norm $n1$, the lifecycle is represented in Figure 1. The maintained state is not represented as it holds in both the activated and fulfilled states. The deactivated state is also not depicted because it corresponds in this case to the Fulfilled state.

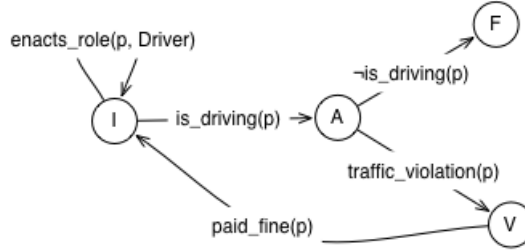


Fig. 1: Lifecycle for norm $n1$ in the traffic scenario: (I)nactive, (A)ctivated, (V)iolated, (F)ulfilled

Definition 6. A Normative Monitor M_I for an institution I is a tuple $M_I = \langle I, S, IS, VS, FS \rangle$ where

- $I = \langle N, R, P, C, O \rangle$,
- $S = \mathcal{P}(O)$,
- $IS = \mathcal{P}(N \times S \times \text{Dom}(S))$,
- $VS = \mathcal{P}(N \times S \times \text{Dom}(S))$, and
- $FS = \mathcal{P}(N \times S \times \text{Dom}(S))$.

The set Γ of possible configurations of a Normative Monitor M_I is $\Gamma = I \times S \times IS \times VS \times FS$.

However, the definition above does not take into account the dynamic aspects of incoming events affecting the state of the world through time. To extend our model we will assume that there is a continuous, sequential stream of events received by the monitor:

Definition 7. An event e is a tuple $e = \langle \alpha, p \rangle$, where

- $\alpha \in P^5$, and
- $p \in F$ and is fully grounded.

We define E as the set of all possible events, $E = \mathcal{P}(P \times F)$

Definition 8. The Labelled Transition System for a Normative Monitor M_I is defined by $\langle \Gamma, E, \triangleright \rangle$ where

- E is the set of all possible events $e = \langle \alpha, p \rangle$
- \triangleright is a transition relation such that $\triangleright \subseteq \Gamma \times E \times \Gamma$

The inference rules for the transition relation \triangleright are depicted in Figure 2.

3 Monitoring with production systems

In our approach, practical normative reasoning is based on a production system with an initial set of rules implementing the operational semantics described in Section 2.2. Production systems are composed of a set of rules, a working memory, and a rule interpreter or engine [2]. Rules are simple conditional statements, usually of the form *IF a THEN b*, where a is usually called left-hand side (*LHS*) and b is usually called right-hand side (*RHS*).

3.1 Semantics of production systems

In this paper we use a simplified version of the semantics for production systems introduced in [1].

Considering a set \mathcal{P} of predicate symbols, and an infinite set of variables \mathcal{X} , where a fact is a ground term, $f \in \mathcal{T}(\mathcal{P})$, and \mathcal{WM} is the *working memory*, a set of facts, a production rule is denoted `if p , c remove r add a` , or

⁵ α is considered to be the asserter of the event. Although we are not going to use this element in this paper, its use may be of importance when extending or updating this model.

Event processed:

$$\frac{e_i = \langle \alpha, p \rangle}{\langle \langle i, s, is, vs, fs \rangle, e_i \rangle, e_{i+1} \rangle \triangleright \langle \langle i, s \cup \{p\}, is, vs, fs \rangle, e_{i+1} \rangle} \quad (1)$$

Counts-as rule activation:

$$\frac{\exists \Theta, \exists f \in F(s), \exists \langle \gamma_1, \gamma_2, s_i \rangle \in C, s_i \subseteq s \wedge \Theta(\gamma_1) \equiv f \wedge \Theta(\gamma_2) \notin s}{\langle \langle \langle N, R, P, C, O \rangle, s, is, vs, fs \rangle, e \rangle \triangleright \langle \langle \langle N, R, P, C, O \rangle, s \cup \{\Theta(\gamma_2)\}, is, vs, fs \rangle, e \rangle} \quad (2)$$

Counts-as rule deactivation:

$$\frac{\exists \Theta, \exists f \in F(s), \exists \langle \gamma_1, \gamma_2, s_i \rangle \in C, s_i \not\subseteq s \wedge \Theta(\gamma_1) \equiv f \wedge \Theta(\gamma_2) \in s}{\langle \langle \langle N, R, P, C, O \rangle, s, is, vs, fs \rangle, e \rangle \triangleright \langle \langle \langle N, R, P, C, O \rangle, s - \{\Theta(\gamma_2)\}, is, vs, fs \rangle, e \rangle} \quad (3)$$

Norm instantiation:

$$\frac{\exists n = \langle f_A, f_M, f_D, w \rangle \in N \wedge \neg \exists n' \in N, n' \rightsquigarrow n \wedge \langle n, \Theta \rangle \notin is \wedge \exists \Theta, \exists f' \in F(s), f' \equiv \Theta(f_A)}{\langle \langle \langle N, R, P, C, O \rangle, s, is, vs, fs \rangle, e \rangle \triangleright \langle \langle \langle N, R, P, C, O \rangle, s, is \cup \{\langle n, \Theta \rangle\}, vs, fs \rangle, e \rangle} \quad (4)$$

Norm instance violation:

$$\frac{\exists n = \langle f_A, f_M, f_D, w \rangle \in N \wedge \langle n, \Theta' \rangle \in is \wedge \langle n, \Theta' \rangle \notin vs \wedge \neg(\exists \Theta, \exists f' \in F(s), f' \equiv \Theta(f_M) \wedge \Theta \subseteq \Theta') \wedge NR = \bigcup_{n \rightsquigarrow n'} \langle n', \Theta' \rangle}{\langle \langle \langle N, R, P, C, O \rangle, s, is, vs, fs \rangle, e \rangle \triangleright \langle \langle \langle N, R, P, C, O \rangle, s, (is - \{\langle n, \Theta' \rangle\}) \cup NR, vs \cup \{\langle n, \Theta' \rangle\}, fs \rangle, e \rangle} \quad (5)$$

Norm instance fulfilled:

$$\frac{\exists n = \langle f_A, f_M, f_D, w \rangle \in N \wedge \langle n, \Theta' \rangle \in is \wedge \exists \Theta, \exists f' \in F(s), f' \equiv \Theta(f_D) \wedge \Theta \subseteq \Theta'}{\langle \langle \langle N, R, P, C, O \rangle, s, is, vs, fs \rangle, e \rangle \triangleright \langle \langle \langle N, R, P, C, O \rangle, s, is - \{\langle n, \Theta' \rangle\}, vs, fs \cup \langle n, \Theta' \rangle \rangle, e \rangle} \quad (6)$$

Norm instance violation repaired:

$$\frac{\exists n, n' \in N \wedge n \rightsquigarrow n' \wedge \langle n, \Theta \rangle \in vs \wedge n \rightsquigarrow n' \wedge \langle n', \Theta \rangle \in fs}{\langle \langle \langle N, R, P, C, O \rangle, s, is, vs, fs \rangle, e \rangle \triangleright \langle \langle \langle N, R, P, C, O \rangle, s, is, vs - \{\langle n, \Theta \rangle\}, fs \rangle, e \rangle} \quad (7)$$

Fig. 2: Inference rules for the transition relation \triangleright

$$p, c \Rightarrow r, a,$$

consisting of the following components:

- A set of positive or negative patterns $p = p^+ \cup p^-$ where a pattern is a term $p_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and a negated pattern is denoted $\neg p_i$. p^- is the set of all negated patterns and p^+ is the set of the remaining patterns
- A proposition c whose set of free variables is a subset of the pattern variables: $Var(c) \subseteq Var(p)$.
- A set r of terms whose instances could be intuitively considered as intended to be removed from the working memory when the rule is fired, $r = \{r_i\}_{i \in I_r}$, where $Var(r) \subseteq Var(p^+)$.

- A set a of terms whose instances could be intuitively considered as intended to be added to the working memory when the rule is fired, $a = \{a_i\}_{i \in I_a}$, where $Var(a) \subseteq Var(p)$.

Definition 9. A set of positive patterns p^+ matches to a set of facts \mathcal{S} and a substitution σ iff $\forall p \in p^+, \exists t \in \mathcal{S}, \sigma(p) = t$. Similarly, a set of negative patterns p^- mismatches a set of facts \mathcal{S} iff $\forall -p \in p^-, \forall t \in \mathcal{S}, \forall \sigma, \sigma(p) \neq t$.

A production rule $p \Rightarrow r$, a is (σ, \mathcal{WM}') -fireable on a working memory \mathcal{WM} when p^+ matches with \mathcal{WM}' and p^- mismatches with \mathcal{WM} , where \mathcal{WM}' is a minimal subset of \mathcal{WM} , and $\mathcal{T} \models \sigma(c)$.

Definition 10. The application of a (σ, \mathcal{WM}') -fireable rule on a working memory \mathcal{WM} leads to the new working memory $\mathcal{WM}'' = (\mathcal{WM} - \sigma(r)) \cup \sigma(a)$.

Definition 11. A general production system \mathcal{PS} is defined as $\mathcal{PS} = \langle \mathcal{P}, \mathcal{WM}_0, \mathcal{R} \rangle$, where \mathcal{R} is a set of production rules over $\mathcal{H} = \langle \mathcal{P}, \mathcal{X} \rangle$.

3.2 Reduction

In order to formalise our Normative Monitor as a production system, we will need to define several predicates to bind norms to their conditions: *activation*, *maintenance*, *deactivation*, and to represent normative state over norm instances: *violated*, *instantiated*, and *fulfilled*. We will also use a predicate for the arrival of events: *event*, and a predicate to represent the fact that a norm instance is a violation handling norm instance of a violated instance: *repair*. For the handling of the DNF clauses, we will use the predicates *holds* and *has_clause*.

Definition 12. The set of predicates for our production system, for an institution $I = \langle N, R, P, C, O \rangle$, is:

$\mathcal{P}_I := O \cup \{activated, maintained, deactivated, violated, instantiated, fulfilled, event, repair, holds, has_clause, countsas\}$

The initial working memory \mathcal{WM}_0 should include the institutional specification in the form of the formulas included in the counts-as rules and the norms in order to represent the possible instantiations of the predicate *holds*, through the use of the predicate *has_clause*.

First of all, we need to have the bindings between the norms and their formulas available in the working memory. For each norm $n = \langle f_A, f_M, f_D, w \rangle$, these bindings will be:

$\mathcal{WM}_n := \{activation(n, f_A), maintenance(n, f_M), deactivation(n, f_D)\}$

As we assume the formulas from $wff(\mathcal{L}_O)$ to be in DNF form:

Definition 13. We can interpret a formula as a set of conjunctive clauses $f = \{f_1, f_2, \dots, f_N\}$, of which only one of these clauses f_i holding true is necessary for f holding true as well:

$r^h := has_clause(f, f') \wedge holds(f', \Theta) \Rightarrow \emptyset, \{holds(f, \Theta)\}$

For example, if $f = (p_1(x) \wedge p_2(y) \wedge \dots \wedge p_i(z)) \vee \dots \vee (q_1(w) \wedge q_2(x) \wedge \dots \wedge q_j(y))$, then the initial facts to be in \mathcal{WM}_0 will be:

$$\mathcal{WM}_0 := \bigcup_{f' \in f} \text{has_clause}(f, f') = \{\text{has_clause}(f, f_1), \dots, \text{has_clause}(f, f_2)\}$$

Also, we have to include the set of repair norms by the use of the predicate *repair*, and the counts-as definitions by the use of the predicate *countsas*.

Definition 14. *The initial working memory \mathcal{WM}_I for an institution $I = \langle N, R, P, C, O \rangle$ is:*

$$\begin{aligned} \mathcal{WM}_I := & \bigcup_{n \rightsquigarrow n'}^{n \in N} \text{repair}(n, n') \cup \\ & \bigcup_{n = \langle f_A, f_M, f_D, w \rangle \in N} (\mathcal{WM}_n \cup \mathcal{WM}_{f_A} \cup \mathcal{WM}_{f_M} \cup \mathcal{WM}_{f_D}) \cup \\ & \bigcup_{c = \langle \gamma_1, \gamma_2, s \rangle \in C} (\{\text{countsas}(\gamma_1, \gamma_2, s)\} \cup \mathcal{WM}_{\gamma_1} \cup \mathcal{WM}_s) \end{aligned}$$

The rule for the detection of a holding formula is defined as $r_f^{hc} = [f] \Rightarrow \emptyset, \{\text{holds}(f, \sigma)\}$, where we denote as $[f]$ the propositional content of a formula $f \in \text{wff}(\mathcal{L}_O)$ which only uses predicates from O and the logical connectives \neg and \wedge , and σ as the substitution set of the activation of the rule. Following the previous example:

$$r_{f_1}^{hc} = p_1(x) \wedge p_2(y) \wedge \dots \wedge p_i(z) \Rightarrow \emptyset, \{\text{holds}(f_1, \{x, y, z\})\}$$

$$r_{f_2}^{hc} = q_1(w) \wedge q_2(x) \wedge \dots \wedge q_i(y) \Rightarrow \emptyset, \{\text{holds}(f_2, \{w, x, y\})\}$$

Similarly as in Definition 14:

Definition 15. *The set of rules R_I^{hc} for detection of holding formulas for an institution $I = \langle N, R, P, C, O \rangle$ is:*

$$R_I^{hc} := \bigcup_{n = \langle f_A, f_M, f_D, w \rangle \in N} (\bigcup_{f \in \{f_A, f_M, f_D\}} r_f^{hc}) \cup \bigcup_{c = \langle \gamma_1, \gamma_2, s \rangle \in C} (\bigcup_{f \in \gamma_1} r_f^{hc})$$

By using the predicate *holds* as defined above, we can translate the inference rules from Section 2.2. Please note that the rules are of the form $p, c \Rightarrow r, a$ as shown in Section 3.1. However, as we only need the c part to create a constraint proposition in the rules for norm instance violation and fulfillment, c is omitted except for these two particular cases.

Definition 16. *Translated rules (see Figure 2)*

Rule for event processing (1):

$$r^e = \text{event}(\alpha, p) \Rightarrow \emptyset, \{[p]\}$$

Rule for counts-as rule activation (2):

$$\begin{aligned} r^{ca} = & \text{countsas}(\gamma_1, \gamma_2, c) \wedge \text{holds}(\gamma_1, \Theta) \wedge \text{holds}(c, \Theta') \wedge \neg \text{holds}(\gamma_2, \Theta) \\ \Rightarrow & \emptyset, \{\Theta([\gamma_2])\} \end{aligned}$$

Rule for counts-as rule deactivation (3):

$$\begin{aligned} r^{cd} = & \text{countsas}(\gamma_1, \gamma_2, c) \wedge \text{holds}(\gamma_1, \Theta) \wedge \neg \text{holds}(c, \Theta') \wedge \text{holds}(\gamma_2, \Theta) \\ \Rightarrow & \{\Theta([\gamma_2])\}, \emptyset \end{aligned}$$

Rule for norm instantiation (4):

$$\begin{aligned} r^{ni} = & \text{activation}(n, f) \wedge \text{holds}(f, \Theta) \wedge \neg \text{instantiated}(n, \Theta) \wedge \neg \text{repair}(n', n) \\ \Rightarrow & \emptyset, \{\text{instantiated}(n, \Theta)\} \end{aligned}$$

Rule for norm instance violation (5):

$$\begin{aligned} r^{nv} = & \text{instantiated}(n, \Theta) \wedge \text{maintenance}(n, f) \wedge \neg \text{holds}(f, \Theta') \wedge \text{repair}(n, n'), \\ \forall \Theta', \Theta' \subseteq & \Theta \end{aligned}$$

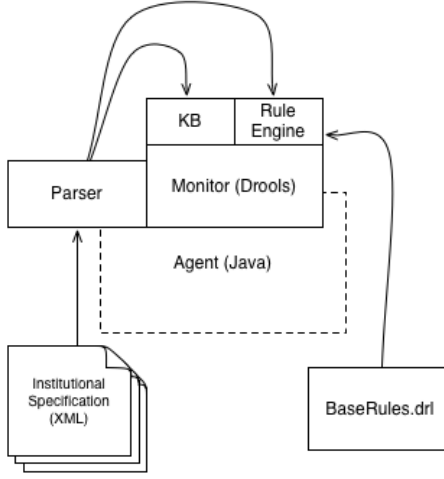


Fig. 3: Architecture of the DROOLS implementation

$\Rightarrow \{instantiated(n, \Theta)\}, \{violated(n, \Theta), instantiated(n', \Theta)\}$

Rule for norm instance fulfillment (6):

$r^{nf} = deactivated(n, f) \wedge instantiated(n, \Theta) \wedge subseteq(\Theta', \Theta) \wedge holds(f, \Theta')$,

$\Theta' \subseteq \Theta$

$\Rightarrow \{instantiated(n, \Theta)\}, \{fulfilled(n, \Theta)\}$

Rule for norm instance violation repaired (7):

$r^{nr} = violated(n, \Theta) \wedge repair(n, n') \wedge fulfilled(n', \Theta')$

$\Rightarrow \{violated(n, \Theta)\}, \emptyset$

Definition 17. Following Definitions 13, 15 and 16, the set of rules for an institution $I = \langle N, R, P, C, O \rangle$ are:

$$\mathcal{R}_I := R_I^{hc} \cup \{r^h, r^e, r^{ca}, r^{cd}, r^{ni}, r^{nv}, r^{nf}, r^{nr}\}$$

Definition 18. The production system \mathcal{PS}_I for an institution I will be, from Definitions 12, 14 and 17:

$$\mathcal{PS}_I := \langle \mathcal{P}_I, \mathcal{WM}_I, \mathcal{R}_I \rangle$$

4 Implementation

A prototype of our normative reasoner has been implemented as a DROOLS program. DROOLS is an open-source Object-Oriented rule engine for declarative reasoning in Java [14]. Its rule engine is an implementation of the forward chaining inference Rete algorithm [4]. The use of Java objects inside the rule engine allows for portability and an easier communication of concepts with the reasoning of agents coded in Java.

```

rule "holds"
when
  HasClause(f : formula, f2 : clause)
  Holds(formula == f2, theta : substitution)
then
  insertLogical(new Holds(f, theta));
end

rule "event processed"
when
  Event(a : asserter, p : content)
then
  insertLogical(p);
end

rule "counts-as activation"
when
  CountsAs(g1 : gamma1, g2 : gamma2, s : context)
  Holds(formula == g1, theta : substitution)
  Holds(formula == s, theta2 : substitution)
  not Holds(formula == g2, substitution == theta)
then
  Formula f;

  f = g2.substitute(theta);
  insert(f);
end

rule "counts-as deactivation"
when
  CountsAs(g1 : gamma1, g2 : gamma2, s : context)
  Holds(formula == g1, theta : substitution)
  not Holds(formula == s, theta2 : substitution)
  Holds(formula == g2, substitution == theta)
  f : Formula(content == g2, grounding == theta)
then
  retract(f);
end

rule "norm instantiation"
when
  Activation(n : norm, f : formula)
  Holds(formula == f, theta : substitution)
  not Instantiated(norm == n, substitution == theta)
  not Repair(n2 : norm, repairNorm == n)
then
  insert(new Instantiated(n, theta));
end

rule "norm instance violation"
when
  ni : Instantiated(n : norm, theta : substitution)
  Maintenance(norm == n, f : formula)
  not (SubsetEQ(theta2 : subset, superset == theta)
  and Holds(formula == f, substitution == theta2))
  Repair(norm == n, n2 : repairNorm)
then
  retract(ni);
  insert(new Violated(n, theta));
  insert(new Instantiated(n2, theta));
end

rule "norm instance fulfillment"
when
  Deactivation(n : norm, f : formula)
  ni : Instantiated(norm == n, theta : substitution)
  SubsetEQ(theta2 : subset, superset == theta)
  Holds(formula == f, substitution == theta2)
then
  retract(ni);
  insert(new Fulfilled(n, theta));
end

rule "norm instance violation repaired"
when
  ni : Violated(n : norm, theta : substitution)
  Repair(norm == n, n2 : repairNorm)
  Fulfilled(norm == n2, substitution == theta)
then
  retract(ni);
end

rule "subseteq"
when
  Holds(f : formula, theta : substitution)
  Holds(f2 : formula, theta2 : substitution)
  eval(theta.containsAll(theta2))
then
  insertLogical(new SubsetEQ(theta2, theta));
end

```

Fig. 4: Translation of base rules to DROOLS

In DROOLS we can represent facts by adding them to the knowledge base as objects of the class *Predicate*. Predicates are dynamically imported from standardised Description Logic OWL-DL ontologies into Java objects using the tool *OWL2Java*[17], as subclasses of a specifically designed *Predicate* class. The following shows an example of the insertion of $enacts_role(p, Driver)$ into the knowledge base to express that p (represented as object p of the domain and instantiating a participant) is in fact enacting the role *driver*:

```
ksession.insert(new Enacts(p, Driver.class));
```

DROOLS programs can be initialised with a rule definition file. However, its working memory and rule base can be modified at run-time by the Java process that is running the rule engine. We take advantage of this by keeping a fixed base, which is a file with fixed contents implementing the rules from Definition 13 and 16, which are independent of the institution, and having a parser for institutional definitions that will feed the rules from Definition 15, which are dependent on the institution (see Figure 3). The institutional definitions we currently use are based on an extension of the XML language presented in [12].

The base rules (see Definitions 13 and 16) has been quite straightforward and the translation is almost literal. The contents of the reusable DROOLS file is shown in Figure 4. The last rule of the Figure is the declarative implementation of the predicate *SubsetEQ* to represent the comparison of substitutions instances $\theta \subseteq \theta'$, needed for the cases of norm instance violation and fulfillment. In our implementation in *Drools*, substitution instances are implemented as *Set<Value>* objects, where *Value* is a tuple $\langle String, Object \rangle$.

The rest of the rules (see Definitions 15) are automatically generated from the institutional specifications and inserted into the DROOLS rule engine. An example of two generated rules for the traffic scenario is shown in Figure 5.

```
rule "N1_activation_1"
when
  n : Norm(id == "N1")
  Activation(norm == n, f : formula)
  Enacts(X : p0, p1 == "Driver")
  IsDriving(p0 == X)
then
  Set<Value> theta = new Set<Value>();
  theta.add(new Value("X", X));
  insert(new Holds(f.getClause(0), theta));
end

rule "C1_1"
when
  c : CountsAs(g1 : gammal)
  Exceeds(D : p0, 50 : p1)
then
  Set<Value> theta = new Set<Value>();
  theta.add(new Value("D", D));
  insert(new Holds(g1.getClause(0), theta));
end
```

Fig. 5: Rules for the traffic scenario

The initial working memory is also automatically generated by inserting objects (facts) into the DROOLS knowledge base following Definition 14. An example for the traffic scenario is also shown in Figure 6. Please note that this is not an output of the parser, but a representation of what it would execute at run-time.

```

ksession.insert (norm1);
ksession.insert (norm2);
ksession.insert (new Repair(norm1, norm2));
ksession.insert (new Activation(norm1, fn1a));
ksession.insert (new Maintenance(norm1, fn1m));
ksession.insert (new Deactivation(norm1, fn1d));
ksession.insert (new HasClause(fn1a, fn1a1));
ksession.insert (new HasClause(fn1m, fn1m1));
ksession.insert (new HasClause(fn1d, fn1d1));
/* ...same for norm2... */
ksession.insert (new CountsAs(clg1, clg2, cls));
ksession.insert (new HasClause(clg1, clg11));
ksession.insert (new HasClause(clg2, clg21));
ksession.insert (new HasClause(cls, cls1));

```

Fig. 6: Facts for the traffic scenario

5 Conclusions and Related Work

The implementation of rule-based norm operationalisation has already been explored in previous research. Some approaches [13,15] directly define the operationalisation of the norms as rules of a specific language, not allowing enough abstraction to define norms at a high level to be operationalised in different rule engine specifications. [5] introduces a translation scheme, but it is bound to Jess by using specific constructs of this language and it does not support constitutive norms.

Other recent approaches like [6] define rule-based languages with expressive constructs to model norms, but they are bound to a proper interpreter and have no grounding on a general production system, requiring the use of an intentionally crafted or modified rule engine. For example, in [7,9], obligations, permissions and prohibitions are asserted as facts by the execution of the rules, but the actual monitoring is out of the base rule engine used.

[16] introduces a language for defining an organisation in terms of roles, norms, and sanctions. This language is presented along with an operational semantics based on transition rules, thus making its adoption by a general production system straightforward. Although a combination of counts-as rules and sanctions is used in this language, it is not expressive enough to support regulative norms with conditional deontic statements.

We solve these issues by combining a normative language [12] with a reduction to a representation with clear operational semantics based on the framework in [11] for deontic norms and the use of counts-as rules for constitutive norms. The formalism presented in this paper uses logic conditions that determine the state of a norm (active,

fulfilled, violated). These conditions can be expressed in propositional logic and can be directly translated into *LHS* parts of rules, with no special adaptation needed. The implementation of the operational semantics in a production system to get a practical normative reasoner is thus straightforward. This allows agents for dynamically changing its institutional context at any moment, by *feeding* the production system with a new abstract institutional specification.

Our intention is not to design a general purpose reasoner for normative agents, but a practical reasoner for detecting event-driven normative states. This practical reasoner can then be used as a component not only by normative agents, but also by monitors or managers. Normative agents should deal with issues such as planning and future possibilities, but monitors are focused on past events. For such a practical reasoner, the expressivity of actions languages like *C+* is not needed, and a simple yet efficient solution is to use production systems, as opposed to approaches more directly related to offline verification or model checking, such as [10].

Mere syntactical translations are usually misleading in the sense that rule language specific constructs are commonly used, constraining reusability [13,5,7]. However, as we have presented in this paper a reduction to a general version of production system semantics, any rule engine could fit our purposes. There are several production system implementations available, some widely used by the industry, such as JESS, DROOLS, SOAR or PROVA. In most of these systems rules are syntactically and semantically similar, so switching from one to the other would be quite simple. As production systems dynamically compile rules to efficient structures, they can be used as well to validate and verify the consistency of the norms. As opposed to [7,9], our reduction ensures that the whole monitoring process is carried out entirely by a general production system, thus effectively decoupling normative state detection and agent reasoning.

DROOLS is an open-source powerful suite supported by JBoss, the community, and the industry, and at the same time it is lightweight enough while including key features that we are or will be using in future work. As an advantage over other alternatives, it includes features relevant to our topic, e.g. event processing, workflow integration. Its OO approach makes it easy to be integrated with imperative code (Java), and OWL-DL native support is expected in a short time.

Our implementation is already being used in several use cases with large amounts of events and it is available at <http://ict-alive.svn.sf.net/viewvc/ict-alive/OrganisationLevel/trunk/Monitoring/> under a GPL license. As future work we expect to extend the semantics in order to support first-order logic norm conditions, and to perform an analysis on the algorithmic complexity of our implementation.

Acknowledgements

This work has been partially supported by the FP7-215890 ALIVE project. J. Vázquez-Salceda's work has been also partially funded by the Ramón y Cajal program of the Spanish Ministry of Education and Science.

References

1. Cirstea, H., Kirchner, C., Moossen, M., Moreau, P.E.: Production Systems and Rete Algorithm Formalisation. Tech. Rep. ILOG, INRIA Lorraine, INRIA Rocquencourt, Manifico (2004)
2. Davis, R., King, J.: An overview of production systems. Tech. rep., Stanford Artificial Intelligence Laboratory, Report No. STAN-CS-75-524 (1975)
3. Dignum, F., Broersen, J., Dignum, V., Meyer, J.J.: Meeting the Deadline: Why, When and How. In: Formal Approaches to Agent-Based Systems, Lecture Notes in Computer Science 3228, pp. 30–40. Springer Berlin / Heidelberg (2005)
4. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence* 19(1), 17–37 (1982)
5. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: Implementing norms in electronic institutions. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. pp. 667–673. Utrecht, Netherlands (2005)
6. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems* 18(1), 186–217 (2009)
7. Governatori, G.: Representing business contracts in RuleML. *International Journal of Cooperative Information Systems* 14(2-3), 181–216 (2005)
8. Grossi, D.: Designing invisible handcuffs: Formal investigations in institutions and organizations for multi-agent systems. Thesis, Universiteit Utrecht (2007)
9. Hübner, J.F., Boissier, O., Bordini, R.H.: Normative programming for organisation management infrastructures. In: Workshop on Coordination, Organization, Institutions and Norms in Agent Systems in Online Communities (COIN@MALLOW 2009) (2009)
10. Kyas, M., Prisacariu, C., Schneider, G.: Run-time monitoring of electronic contracts. In: Proceedings of 6th International Symposium on Automated Technology for Verification and Analysis, Lecture Notes in Computer Science 5311, pp. 397–407. Springer Berlin / Heidelberg (2008)
11. Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a formalisation of electronic contracting environments. In: Coordination, Organizations, Institutions and Norms in Agent Systems IV, Lecture Notes in Computer Science 5428, pp. 156–171. Springer Berlin / Heidelberg (2009)
12. Panagiotidi, S., Vázquez-Salceda, J., Alvarez-Napagao, S., Ortega-Martorell, S., Willmott, S., Confalonieri, R., Storms, P.: Intelligent Contracting Agents Language. In: Proceedings of the Symposium on Behaviour Regulation in Multi-Agent Systems (BRMAS 2008) at AISB 2008. vol. 1, p. 49. Aberdeen, Scotland (2008)
13. Paschke, A., Dietrich, J., Kuhla, K.: A Logic Based SLA Management Framework. In: Proceedings of the 4th Semantic Web Conference (ISWC 2005). pp. 68–83. Galway, Ireland (2005)
14. Proctor, M., Neale, M., Frandsen, M., Jr., S.G., Tirelli, E., Meyer, F., Verlaenen, K.: Drools documentation. JBoss (2008)
15. Strano, M., Molina-Jimenez, C., Shrivastava, S.: A rule-based notation to specify executable electronic contracts. In: Proceedings of the International Symposium on Rule Representation, Interchange and Reasoning on the Web (RuleML2008), Lecture Notes in Computer Science 5321, pp. 81–88. Springer Berlin / Heidelberg (2008)
16. Tinnemeier, N., Dastani, M., Meyer, J.J.: Roles and norms for programming agent organizations. In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009). vol. 1, pp. 121–128. Budapest, Hungary (2009)
17. Zimmermann, M.: OWL2Java (2009)

Controlling multi-party interaction within normative multi-agent organizations

Olivier Boissier¹, Flavien Balbo^{2,3}, and Fabien Badeig^{2,3}

¹ Ecole Nationale Supérieure des Mines,
158 Cours Fauriel, 42100 Saint-Etienne, France
`Olivier.Boissier@emse.fr`

² Université Paris-Dauphine - LAMSADE,
Place du Maréchal De Lattre de Tassigny, F-75775 Paris 16 Cedex, France
`balbo@lamsade.dauphine.fr`

³ INRETS - GRETIA,
2, Rue de la Butte Verte, 93166 Noisy Le Grand, France
`badeig@inrets.fr`

Abstract. Multi-party communications taking place within organizations lead to different interaction modes between agents (e.g. (in)direct communication between roles, (in)direct communication restricted to a group, etc). Fully normative organisations need to regulate and control those modes as they do for agents' behaviors. This paper proposes to extend the normative organization model *MOISE* in order to specify such interaction modes between autonomous agents participating to an organization. This specification has two purposes: (*i*) to make the multi-agent organization able to monitor the interaction between the agents, (*ii*) to make the agents able to reason on these modes as they can do on norms. The paper is focused on the first point. We illustrate with a crisis management application how this extension has been implemented thanks to a specialization of the EASI interaction model.

1 Introduction

In a Multi-Agent System (MAS), interaction and organization play key and essential roles. A MAS is often described as composed of agents situated in a shared environment interact directly or indirectly with each other to execute and cooperate in a distributed and decentralized setting according to an organization. The EASI model⁴ [7] proposes a multi-party environment based interaction model and is therefore able to support the complexity of the interaction within an organization. On one hand, agents are able to send messages to other agents situated in the environment and, on the other hand, any agent situated in the environment is able to perceive the exchanged message. It is thus possible to consider, for the same message, direct, indirect and overhearing communications. If needed, the EASI model preserves the privacy of the interaction, more details are given in [7].

⁴ Environment as Active Support for Interaction

However, it is not possible to make the agents aware of this interaction settings. There is no declarative representation usable at the agent level.

Agents in a MAS are often structured along one organization that helps and/or constraints their cooperation schemes. Current proposals offer modeling languages usable either by agents either by an organization management system dedicated to the regulation and supervision of the agents within the defined organization. The *MOISE* model [5] is one of these proposals. Its organization modeling language is composed of two dimensions – structural and functional - connected to each other by a normative specification. Such a feature makes it possible to easily extend the model with new dimensions. Currently, there doesn't exist any dimension dedicated to the definition of interaction modes within the organization. It is thus not possible to govern the agents interaction modes resulting from the multi-party communications offered in the EASI framework.

In this paper, our aim is to propose a unified model for interaction and organization. To reach this objective we propose the enrichment of the *MOISE* organization modeling language with a new and independent dimension connected to the other ones by the normative specification. This way it is thus possible to make the agents able of reasoning on their use of the interaction modes offered in the EASI platform. The next step will be to develop such reasoning capabilities in the agents. In this paper we focus on the presentation of the unified model and how it is translated to be monitored by the facilities offered by the EASI platform. The MAS designer will be able to use the resulting specification of both the organization and the interaction and to get the corresponding support environment.

The paper is organized as follows. In section 2, we present the background of the proposal and motivate our choices. In section 3, we expose how *MOISE* organization modeling language has been extended to specify the interaction modes proposed by EASI. The section 4 describes how this specification is mapped to the EASI model. In section 5, we show the expressing capabilities of the proposal with different examples issued of a crisis management application. Before conclusion, we compare our proposal to the current related approaches.

2 Background

In this paper, we consider a crisis management application where different dedicated emergency services must be coordinated in order to solve a crisis situation. The main difficulty in the modeling of such an application consists in the definition of the interaction constraints between those services, given that each service has the possibility to decide on its own which interaction mode to use. We use this application all along the paper to illustrate the components of our proposal. In the following sections, we specify parts of the multi-agent organization governing this application thanks to the models *MOISE* and EASI.

2.1 Moise

The MOISE framework [4] is composed of an organization modeling language, an organization management infrastructure and organization based reasoning mechanisms at the agent level. In this paper, we focus on the organization modeling language. Our aim is to use it with the EASI platform in order to specify and regulate the different interaction modes available on this platform (see next section).

The organization modeling language considers the specification of an organization along three independent dimensions⁵: structural (SS), functional (FS) and normative (NS). Whereas SS and FS are independent, NS defines a set of norms binding elements of both specifications. The aim is that the agents enact the behaviors specified in NS when participating to the organization. The organization modeling language is accompanied by a graphical language (cf. Fig. 1, 2) and XML is used to store the organizational specifications.

Structural Dimension: The structural dimension specifies the *roles*, *groups*, and *links* of an organization. It is defined with the following tuple: $\langle \mathcal{R}, \sqsubset, rg \rangle$ with \mathcal{R} set of the roles, \sqsubset , inheritance relation between roles, rg organization root group specification. The definition of this group gives the *compatibility* relations between roles, the maximal and minimal *cardinality* of agents that can endorse roles within the group, the *links* connecting roles to each other (communication, authority, acquaintance) and *sub-groups*. In NS , the role is used to bind a set of constraints on behaviors that the agent commits to satisfy as soon it endorses the role.

In the crisis application, we define (cf. Fig. 1) two main groups which correspond to the tactical spheres used in a crisis management: decision-making sphere (**Decision-making**) and operational sphere (**Operational**). For each of them, we define the roles **manager** and **operator** inheriting the generic role **role-player**. These roles are specialized respectively in **coordinator**, **leader_D** for the group **Decision-making** and **leader_S** for the subgroups of group **Operational**. The role **coordinator** (resp. **leader_D**) can be played by only and only one agent - 1.1 - (resp. several agents - 1.* -). A compatibility link connects the role **leader_D** to **leader_S** meaning that any agent playing **leader_D** will be able to play also the role **leader_S**. Six communication links (cf. l_1 to l_6) have been defined between these roles (e.g. l_1 communication link between **coordinator** and **leader_D**).

Functional Dimension: The functional dimension is defined by $\langle \mathcal{M}, \mathcal{G}, \mathcal{S} \rangle$ with \mathcal{M} set of *missions*, consistent grouping of collective or individual goals. A mission defines all the goals an agent commits to when participating in the execution of a social scheme by the way of the roles that they endorse. \mathcal{G} is the set of the *collective or individual goals* to be satisfied and \mathcal{S} is the set of *social schemes*, tree-like structurations of the goals into plans.

⁵ In this paper, we will provide the only necessary details in order to globally understand the model as well as the proposed extensions. For further details, readers should refer to <http://moise.sourceforge.net/>.

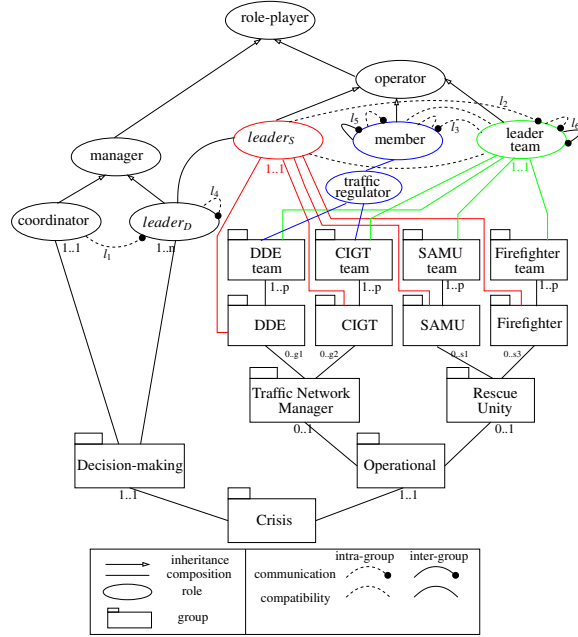


Fig. 1. Partial graphical view of the structural specification for the crisis management application

The Fig. 2 illustrates a social scheme of FS expressing the collective plan for deciding within the crisis management application. According to it, agents should aggregate the different information in relation to the crisis situation Refining crisis perception, Safeguarding zone by executing one of the two social schemes (scheme 1 or scheme 2 that are not detailed here) and execute the scheme 3. The different goals are organized into missions.

Normative Dimension: The normative dimension \mathcal{NS} defines a set of norms as: $\langle id, c, \rho, dm, m \rangle$ with id norm identifier, c activation condition of the norm⁶, ρ role concerned by the deontic modality, dm deontic modality (obligation or permission), m mission. A normative expression can be read as : “when c holds, any agent playing role ρ has dm to commit on the mission m ”. Within this language, norms are either a *permission*, either an *obligation* for a role to commit to a mission. Goals are indirectly connected to roles since a mission is a set of goals. Interdictions are supposed to exist by default: if the normative specification doesn’t have any permission or obligation for a couple mission, role, any agent playing the role is forbidden to commit to mission. A norm becomes in the *active* state (resp. *inactive*) as soon as the condition c holds (resp. doesn’t

⁶ Predicates bearing on the current organization state (e.g. *plays*, *committed*, etc) and/or bearing on particular configurations of the application.

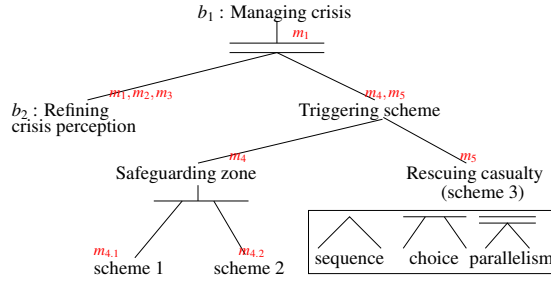


Fig. 2. Graphical view of the social scheme for decision within the crisis management application. Goals are the nodes of the tree. Missions to which the goals are assigned are in red.

hold). When the norm is active, the deontic expression attached to the norm can be verified. The norm can thus become *fulfilled* or *unfulfilled*.

For instance, in the crisis management application, the norm obliging agents playing the role $leaders_S$ in the group Traffic Network Management (TNM) to safeguard the zone where the accident took place (mission m_4) is: $\langle n_1, c_1, leaders_S, obligation, m_4 \rangle$ where c_1 is $plays(bearer, leader_S, TNM)$. The term *bearer* refers to the agent that will play the role “bearer” in the context of the obligation issued from the instantiation of the norm in the organization entity (see below) and *plays* is a predicate satisfied when the agent plays the $leaders_S$ in an instance of group TNM. When the zone is secured, the agents playing the same role within the context of the group Rescue Unity (RU) deploys the intervening scheme (mission m_5) following the norm: $\langle n_2, c_2, leader_D, obligation, m_5 \rangle$ where c_2 is $plays(bearer, leader_S, RU)$.

Organizational Entity: An organizational entity (OE) is defined from the organizational specification OS and a set of agents \mathcal{A} by the following tuple: $\langle OS, \mathcal{A}, \mathcal{GI}, \mathcal{SI}, \mathcal{O} \rangle$ where \mathcal{GI} is the set of concrete groups of the organization, i.e. groups dynamically created from the group specification of the OS , \mathcal{SI} is the set of concrete social schemes dynamically created in the OE from the social schemes specification in the OS and \mathcal{O} is the set of obligations issued from the norms \mathcal{NS} attached to agents of \mathcal{A} whose conditions are satisfied [6].

2.2 Easi

The multi-party interaction model EASI supports the management of direct, indirect and overhearing communications in a MAS [7]. For cognitive agents, the common point between all these communication modes consists in the *routing* of the messages by identifying *which agent* should obtain *which message* and in *which context*. Solving this problem requires taking into account both sides of the sender and potential receivers. To this aim, EASI manages meta-informations on the MAS (agents, messages, context) in the communication environment and use them to help the agents to interact. The interaction model EASI is thus

defined by $\langle \Omega, \mathcal{D}, P, \mathcal{F} \rangle$ with $\Omega = \{\omega_1, \dots, \omega_m\}$ the set of entities ($\mathcal{A} \subset \Omega$ and $MSG \subset \Omega - \mathcal{A}$ set of agents and MSG set of messages -), $\mathcal{D} = \{d_1, \dots, d_m\}$ set of domain descriptions of the properties, $P = \{p_1, \dots, p_n\}$ set of properties, and $\mathcal{F} = \{f_1, \dots, f_k\}$ set of filters.

Entity: The entities are the meta-information on the MAS that EASI manages. An entity $\omega_i \in \Omega$ is defined by $\langle e_r, e_d \rangle$ where e_r is a reference to an element of the MAS and e_d is the description of that element. An element of the MAS can be agents (\mathcal{A}), messages (MSG) and a reference is its physical address on the platform or other objects such as URL, mailbox, The description e_d is defined by a set of couples $\langle p_i, v_j \rangle$ where $p_i \in P$ and v_j is the value of the property for this entity. Any agent of the MAS has its own processing and knowledge settings. It is connected to the communication environment by the way of its description that it stores and updates in this environment. This description e_d is used for the routing of the informations to the reference e_r .

Property: A property gives an information on an entity. A property $p_i \in P : \Omega \rightarrow d_j \cup \{unknown, null\}$ is a function whose description domain $d_j \in \mathcal{D}$ can be quantitative, qualitative or a finite set of data. The *unknown* value is used when the value of the property cannot be set, and *null* is used to state that the property is undefined in the given description. In order to simplify the notation, only the value of the description domain is given to specify a property.

For instance, in the crisis management application, the properties attached to agents and messages are *id, role, position, subject, sender* with:

- $id : \Omega \rightarrow N$,
- $role : \Omega \rightarrow \{coordinator, leader_S\}$,
- $position : \Omega \rightarrow N \times N$,
- $subject : \Omega \rightarrow \{alert, demand\}$,
- $sender : \Omega \rightarrow N$.

An agent a can have the following description $\{\langle role, coordinator \rangle\}$ and an agent b $\{\langle role, leader_S \rangle, \langle position, (10, 20) \rangle\}$ and a message m $\{\langle subject, alert \rangle, \langle position, (15, 20) \rangle\}$.

Filter: The filter identifies the entities according to their description (e_d) and realizes the interaction between the concrete objects (e_r). A filter $f_j \in \mathcal{F}$ is a tuple $f_j = \langle f_a, f_m, [f_C], n_f \rangle$ with n_f filter name. The assertion $f_a : A \rightarrow \{T, F\}$ identifies the receiving agents (*which agent*), the assertion $f_m : MSG \rightarrow \{T, F\}$ identifies the concerned messages (*which message*), and $f_C : \mathcal{P}(\Omega) \rightarrow \{T, F\}$ is an optional set of assertions identifying other entities of the context (*which context*).

Each agent $?r$ ('?' preceedings a letter denotes a variable) whose description validates f_a receives in its mailbox the message $?m$ that satisfies f_m if there exists a set of entities in the $?c$ such that f_C is true.

A filter is therefore valid for any tuple $\langle ?r \in \mathcal{A}, ?m \in MSG[, ?c \subset \Omega] \rangle$

For instance, in the crisis management application, the filter Fe sets the routing of the communication as follows ('=' is the comparison operator):

- The agents with the role `leaderD` that are situated in the crisis origin:
 $f_a : [role(?r) = leader_D] \wedge [position(?r) = (0, 0)]$
- should receive the alert messages:
 $f_m : [subject(?m) = alert] \wedge [sender(?m) = ?ide]$
- of the agent playing the role `coordinator`:
 $f_C : [id(?e) = ?ide] \wedge [role(?e) = coordinator]$

In this example, the description of the message sender ($?e$) that is identified thanks to the property *sender* in the message belongs to the context. Agents wishing to send or receive a message, update their description in the communication platform and add/retract dynamically in/from the environment filters that involve them. Thus the environment supports simultaneously direct interaction (including dyadic, broadcast multi-cast and group communication) and indirect interaction (including mutual-awareness and overhearing). If the filter is added by the future receiver of the message then the interaction is indirect: the depositary agent defines which message it wants to receive. If the filter is added by the future message sender then the interaction is direct: the depositary agent defines which agent it wants to contact.

According to the state of the different descriptions within the environment, the triggered filter will enable the routing of the messages in the different interaction modes towards the corresponding targets.

Even if EASI offers an advanced communication management by identifying precisely the interaction context, it cannot be used by the agents in order to reason on the causes of the interaction. For instance, the filter Fe will permit the routing of messages but the reasons of this requirement is not expressed within EASI. For the filter Fe , the choice of the communication mode may depend on the relations between the roles `coordinator` and `leaderD`: the *coordinator* sends messages to *leaders* (direct mode) for dedicated messages whereas the *leaders* listen to all the messages issued from the *coordinator* (indirect mode). Using this knowledge, an agent could reason on the current interactions. For instance, the coordinator may choose a direct interaction to handle certain informations and indirect interaction for others. The leaders can deduce the importance to the informations according to the filter used to receive informations. The specification of communications within an organizational model would help the agents to relate communication filters to the reasons that cause the use of such a communication channel.

3 Extending \mathcal{M} oise for Easi

In order to clearly specify the interaction modes used in EASI, we are going to enrich and extend the organization modeling language of \mathcal{M} oise with a new dimension. This new dimension is called *communication mode specification* (noted CS). It is dedicated to expressing the communication modes that will be used within the organization. As the other \mathcal{M} oise dimensions, we keep it independent of SS and FS . We use the same principle to connect it to the other dimensions and enrich the normative specification accordingly. The aim is to be able to

connect the communication modes to the structure and functioning of the organization by the way of norms. Those norms will be made accessible to the agents when interacting with other agents of the organization.

The organization specification is thus enriched into the following 4-uple: $\langle SS, FS, CS, NS \rangle$ with CS communication modes specification and NS the modified normative specification. We detail these two components in what follows.

3.1 Communication modes specification

The specification CS is composed of the set of communication modes $cm \in CS$ considered in the organization.

A communication mode is defined as: $\langle type, direction, protocol \rangle$ with *type*, the type of the communication mode (*direct* or *indirect*), *direction*, the message transmission direction (*unidirectional* or *bidirectional*), *protocol*, the interaction protocol that is used. The values of this last variable correspond to the name of the different interaction protocols that the designer wishes to be used and deployed in the organization (e.g. *FIPAREQUEST*, *Publish_Subscribe*, ...).

As we will see below, a communication mode qualifies the communication link defined in the structural specification between roles. The communication link is directed from the *initiator* of the communication - source of the link - to the *participant* - target of the link -. Therefore, a link can be considered as:

- a unidirectional channel, letting circulate messages in only one direction,
- a bidirectional channel, letting circulate messages in both directions from the initiator to the participant and inversely.

Orthogonal to these two directions, we consider the direct and indirect interaction models proposed within EASI.

In the crisis management application, we define, for instance, the two following communication modes $cm_{d,b}$ and $cm_{i,u}$:

$cm_{d,b}$: $\langle direct, bidirectional, FIPAREQUEST \rangle$

$cm_{i,u}$: $\langle indirect, unidirectional, PublishSubscribe \rangle$

where $cm_{d,b}$ is used to directly ask for information whereas $cm_{i,u}$ is used to provide informations to agents that will consult it when they want.

3.2 Communication Norms

In order to bind communication link and communication mode as defined in CS by making explicit the deontic modalities attached to their use, we generalize the writing of the norms described in the *MOISE* initial version: $\langle id, c, \rho, dm, object \rangle$ where *id* is norm identifier, *ca* the activation condition, ρ the role on which the deontic modality bears, *dm* the deontic modality (obligation or permission), *object* the subject of the norm.

Object of a norm: The object of a norm *object* is defined as the two following expressions:

- $do(m)$ in the case where a mission m has to be executed - case initially considered in *MOISE*,
- $use(l, cm, \alpha)$ in the case where the communication mode cm should be used for the link l in the context α .

Context: The context α defines the constraints bearing on the descriptions e_d of the entities $\omega_i \in \Omega$ (cf. Sec. 2.2) involved in the interaction using this communication link: sender, receiver, message. It is also possible to add additional descriptions issued from other entities of the MAS (e.g. requirements of the agent, ...). It is thus possible to use a mission as context of use of the link or a particular goal as context of use of this link. We will define in the following section the way we express these constraints when describing how EASI has been specialized to handle *MOISE*. When α 's status is T (true), the link is usable in any situation.

Let's consider the communication link l_1 used by the agents playing the role *coordinator* towards agents playing the role *leader_D* (eg. Fig. 1) in the crisis management application. Given the normative specification that we have defined, it is possible to bind to it the communication mode $cm_{i,u}$ defined above, by writing the following norm: $n_1 \langle n_1, c_1, coordinator, obligation, use(l_1, cm_{i,u}, T) \rangle$ with $c_1 : committed(m_1)$ to express that l_1 ought to be used by agents playing the role *coordinator* when they are committed to the mission m_1 . No particular context is attached to the use of the communication mode $cm_{i,u}$.

We can also attach to this link another communication mode $cm_{d,b}$, by defining a new norm $n_2 : \langle n_2, c_2, coordinator, obligation, use(l_1, cm_{d,b}, \alpha_2) \rangle$ with $c_2 : committed(m_4)$ by specifying a context α_2 (cf. following section for the syntax) stating that the communication on the link l_1 takes place for the sending of messages to agents belonging to group *CIGT* (Center of the Engineering and Management of the Traffic). The link l_1 can also be bound to the same constraints but for communication in the context α_3 stating the sending of messages from the agent playing role *coordinator* to agent belonging to group *TNM*: $\langle n_3, c_2, coordinator, obligation, use(l_1, cm_{d,b}, \alpha_3) \rangle$.

In the following, we will need to access to the different features of a communication link from the structural specification. We will use pointed notation $l_j.initiator$ (resp. $l_j.participant$) to access to the source role (resp. target) of the link l_j , and $l_j.group$ to access to the group in which l_j is defined.

4 Specializing Easi for *Moise*

Our objective is to generate filters for the communication environment from the specifications defined in the organization modeling language. These filters use informations on the organization. These informations should be stored in the description of the entities that are managed by the communication environment

in order to be accessible to the filters. In this section, we identify and define the necessary properties for describing the agents and messages in the communication environment. Then we describe a generic filter generated from the communication norms that we just defined in the previous section.

4.1 Properties

In order to connect organization and interaction, it is necessary to give a minimal description of an agent, of a message while incorporating this new dimension in them. Given the definition of an entity in section 2.2, we define the following properties that are accessible in the environment for each type of entity.

Agent Properties: The description of an agent is at least composed of the *id* and *org* properties, where:

- *id* returns the identifier of the agent ($id : A \rightarrow ID_A$ with ID_A set of agent identifiers),
- *org* returning the subset of organizational descriptions coming from the participation of the agent to the organization ($org : A \rightarrow \mathcal{P}(OC)$ with OC set of organization descriptions).

An organization description $oc_i \in OC$ is defined by: $oc_i = \langle ig : g, r, m, go \rangle$ with $ig \in \mathcal{IG}$, $g \in \{rg\} \cup rg.subgroups$, $r \in R$, $m \in M$, $go \in G$. *ig* is a concrete group created from the group specification *g* defined in the SS of the organization. The parameter *rg* and the sets *R*, *M*, *G* are defined in the organization specification (cf. Sec. 2.1).

For instance, in the crisis management application, the agent *a* described by $org(a) = \{\langle g1 : Decision.making, leader_D, m_2, b_2 \rangle, \langle g2 : DDE, leader_S, m_1, b_1 \rangle\}$, belongs to the group *g1* of type Decision-making and to a group *g2* of type DDE, in which it plays respectively role $leader_D$, committed to mission m_2 , trying to achieve goal b_2 and the role $leader_S$, committed to the mission m_1 trying to achieve the goal b_1 .

This description of an agent is minimal. We defined two management modes. Being related to the organization, these properties can be managed without being intrusive: management by the organization management infrastructure. However, if this set is complemented by specific properties related to the internal state of the agents, their management is ensured by the agents themselves. For instance, a property *availability* returning the availability of the agent has a value that is related to the choice of the agent itself.

Message Properties: In the same way, we specialize the description of a message with the following set of minimal properties *sender*, *receiver*, *subject*, *rc*, *sc* where:

- *sender* : $MSG \rightarrow ID_A$,
- *receiver* : $MSG \rightarrow \mathcal{P}(ID_A) \cup \{unknown\}$,
- *subject* : $MSG \rightarrow D_{subject} \cup \{unknown\}$, with $D_{subject} = G \cup R \cup \{expression\}$, *expression* is a string,

- $rc : MSG \rightarrow OC \cup \{unknown\}$ being the reception context,
- $sc : MSG \rightarrow OC \cup \{unknown\}$ being the sending context.

Using these properties, the sender gives informations on the organizational context in which the interaction takes place. For a message, each of these properties can receive a value or the value *unknown*. The more the sender specifies values of properties, the more precise will be the filter that can be used for the routing. We impose that the property *sender* doesn't get a value *unknown* in order to avoid anonymous messages.

Given these different properties, we have now the possibility of a routing ranging from indirect interaction, based on only the identifier of the sender, to one, focused on a subset of receivers (*receiver*) in a particular organizational context (*rc*), with a sender being in a given organizational context (*sc*) and the message with a particular content (*subject*).

The sender can also decide to define *patterns* for conditioning the routing along different organizational contexts. To this aim, it can use the symbol '*'*' as value for an element of the organizational context. This symbol denotes that the value is not a constraint in the choice of the tuples.

For instance, in the crisis management application, the expression $\langle _ : DDE, _ , m_2, _ \rangle$ defines an organizational pattern of *OC* such that the concrete group must be of type DDE and the mission is m_2 whatever are the values for roles and goals. The message mes_1 described below means that the sender whose identifier is a_1 and having the goal b_2 (sending context) sends a message to the agents a_2 and a_4 . In this case, the processing of the message is not constrained by the organizational states of the participating agents. They only have to be trying to achieve the organizational goal b_2 . $\langle \langle sender, a_1 \rangle, \langle receiver, \{a_2, a_4\} \rangle, \langle subject, demand \rangle, \langle rc, \langle _ : _ , _ , _ , b_2 \rangle \rangle, \langle sc, \langle _ : _ , _ , _ , b_2 \rangle \rangle \rangle$

For the sender, these are only possibilities since the routing of the message depends on the filters that are installed in the communication environment.

In fact, according to the filters that are installed in it, the routing of the message can lead to different situations: interaction as intended by the sender, no interaction or interaction not intended by the sender. For instance, the agent a_2 can receive the message although it doesn't have the goal b_2 in the case there exists a filter enabling the reception of messages from the agent a_1 , whatever are the values for the properties of the message.

In each message is stored the organizational context of its sending in order to enable the agents to filter them. An agent can thus choose to receive messages or to route them according to their organizational contexts without being imposed their use. Moreover, this definition of messages enables to consider the evolution of the organization state. Thus, a message kept in the environment can still be received by an agent in case of change of the organization state. For instance, an agent can be interested by any message whose receiving context concerns a role that it just endorsed. It is useful to keep an history of the past interaction

to better understand the current situation. Another advantage is to avoid that a message is missed because it has been sent before the agent has endorsed the role. In order to avoid a risk of confusion between messages, a property related to the time value of their emission or related to their life time can be added to the message description. This choice belongs to the system designer and is out of the scope of this paper.

4.2 From Communication Norms to Environment Filters

The activation of a norm for a communication link leads to the generation and addition of a filter in the environment. This filter is called *normative filter*. It corresponds to the exact translation of the norm as it is instantiated by the organization management system. Thanks to the organization management system, the agents are aware of the norm activation. Besides to the normative filters, the communication environment contains also filters set by the agents according to their activity in the system. In case of direct interaction, the sender knows that it can reach the agents identified as receiver in the norm. In case of indirect interaction, the receiver knows that it can receive messages identified in the norm.

A normative filter uses all the possible informations coming from the organizational specification and routes a message according to its *sc* and *rc* properties. The property *receiver* is not used in the generation process of a normative filter since it requires that the sender knows the identifiers of the agents. This is a too strong hypothesis. The same way, since the routing comes from the activation of a norm, the filter cannot constrain the subject of the message (*subject*) except additional conditions in the norms (context α of the object of the norm). The filter identifies a state of the context corresponding to the interaction. It is identical in the direct and indirect cases. We then propose a *generating pattern* that will be specialized for each activated communication norm.

Access to the organizational specification: The normative filter is created when the norm is activated as follows.

Let's first define the functions *initiator* and *participant* that access to the agents involved in the communication link defined in the object of the norm.

$$\text{initiator} : \mathcal{O} \rightarrow A \qquad \text{participant} : \mathcal{O} \rightarrow A$$

These functions return, for an instantiated norm, the agent (*initiator*), who initiates, or the one (*participant*), who participates, to the interaction. From these two functions, we express constraints on the descriptions of the agents. For instance $org(\text{initiator}(n_j))$ makes possible to access to the organizational context attached to the description of the agent initiating the communication in the context of the instantiated norm n_j in which it is involved.

Let's define the predicate $achieves_\alpha$ that is automatically generated from the constraints expressed in the context α of the object of a norm. This predicate checks that the context is satisfied given the initiator, participant, message and entity descriptions in the environment, given α :

$$achieves_\alpha : A \times MSG \times A \times \mathcal{P}(\Omega) \rightarrow \{T, F\}$$

Given the previous definitions, we are able now to express the generic normative filter $f_{n_k}(?p, ?m, \{?i, C\})$ for the receiver $?p$ of the message $?m$ sent by $?i$ in the context C . This filter has been generated from the activation of the norm n_k . The object of the norm bears on the communication link l_j . It is composed of assertions f_a that identifies the receiver of the message $?p$ according to its organizational context, f_m that identifies the message $?m$ according to its organizational context and f_c that identifies the organizational context of the sender and the constraints α of the norm n_k .

$$\begin{aligned} f_a &: \langle [org(?p) \ni \langle ?x : l_j.group, l_j.participant, -, - \rangle] \\ f_m &: \langle [sender(?m) = id(?i)] \wedge [sc(?m) = \langle ?y : l_j.group, l_j.initiator, -, - \rangle] \wedge [rc(?m) = \langle ?x : l_j.group, l_j.participant, -, - \rangle] \\ f_c &: \langle [org(?i) \ni \langle ?y : l_j.group, l_j.initiator, -, - \rangle] \wedge achieves_\alpha(?p, ?m, ?i, C) \rangle \end{aligned}$$

Let's consider again the norm n_2 of the crisis management application: $\langle n_2, committed(m_4), coordinator, obligation, use(l_1, cm_{d,b}, \alpha_2) \rangle$ with $\alpha_2 : [\langle - : CIGT, -, - \rangle \in org(participant(n_2))]$. The interaction is a direct and bidirectional one (cf. $cm_{d,b}$ of n_2). The sending agent deposits the first message. The two necessary filters have been generated and added thanks to the activation of n_2 .

The normative filter generated for n_2 for the interaction from initiator to participant is $f_{n_2}(?p, ?m, \{?i, C\})$: where :

$$\begin{aligned} f_a &: \langle [org(?p) \ni \langle ?x : Decision_making, leader_D, -, - \rangle] \\ f_m &: \langle [sender(?m) = id(?i)] \wedge [sc(?m) = \langle ?y : Decision_making, coordinator, -, - \rangle] \wedge [rc(?m) = \langle ?x : Decision_making, leader_D, -, - \rangle] \\ f_c &: \langle [org(?i) \ni \langle ?y : Decision_making, coordinator, -, - \rangle] \wedge [org(?p) \ni \langle - : CIGT, -, - \rangle] \rangle \end{aligned}$$

The normative filter from the participant to the initiator is $f_{n_2}(?i, ?m, \{?p, C\})$:⁷, where:

$$\begin{aligned} f_a &: \langle [org(?i) \ni \langle ?x : Decision_making, coordinator, -, - \rangle] \\ f_m &: \langle [sender(?m) = id(?p)] \wedge [sc(?m) = \langle ?y : Decision_making, leader_D, -, - \rangle] \wedge [rc(?m) = \langle ?x : Decision_making, coordinator, -, - \rangle] \\ f_c &: \langle [org(?p) \ni \langle ?y : Decision_making, leader_D, -, - \rangle] \wedge [org(?i) \ni \langle - : CIGT, -, - \rangle] \rangle \end{aligned}$$

This way, for two agents participating to the same concrete group, the message sent by the initiator agent a_1 processed by the filter f_{n_2} will have the following description: $\langle \langle sender, id(a_1) \rangle, \langle rc, \langle g1 : Decision_making, coordinator, -, - \rangle \rangle, \langle sc, \langle g1 : Decision_making, leader_D, -, - \rangle \rangle \rangle$

The message sent by the participant agent a_2 , processed by f_{n_2} will have the following description: $\langle \langle sender, id(a_2) \rangle, \langle rc, \langle g1 : Decision_making, leader_D, -, - \rangle \rangle, \langle sc, \langle g1 : Decision_making, coordinator, -, - \rangle \rangle, \rangle$

With these two filters, a communication channel has been created between agents having the roles coordinator and responsible in the group CIGT. The interaction model EASI has made possible to elaborate these filters. The MOISE model has made possible its use.

⁷ we continue to use the variable $?p$ for the participant in the interaction and $?i$ for the initiator given that the agent which is identified by the variable $?i$ who receives the message sent by $?p$.

5 Example

In this section, we illustrate and discuss the expressing capabilities of our proposal going back to the interaction modes attached to the communication link l_1 issued of the communication norms n_1 , n_2 , n_3 in the crisis management application described in the paper.

- $\langle n_1, c_1, coordinator, obligation, use(l_1, cm_{i,u}, T) \rangle$ with $c_1 : committed(m_1)$
- $\langle n_2, c_2, coordinator, obligation, use(l_1, cm_{d,b}, \alpha_2) \rangle$ with $c_2 : committed(m_4)$
and $\alpha_2 : \langle - : CIGT, -, -, - \rangle \in org(participant(n_2))$
- $\langle n_3, c_3, coordinator, obligation, use(l_1, cm_{d,b}, \alpha_3) \rangle$ with $c_3 : committed(m_4)$
and $\alpha_3 : \langle - : TNM, -, -, - \rangle \in org(participant(n_3))$

On these three norms, the differences bear on the *activation conditions* of the norm c_x , the *communication mode* $cm_{x,y}$ and the *communication context* specified in the object.

The norm n_1 whose activation condition bears on the management of the crisis (mission m_1) is activated during all the crisis management. The norms n_2 and n_3 are not active since the agents on which the norms bear are committed on the mission m_4 .

The predicate $achieves_\alpha$ of the normative filter f_{n_1} generated from the norm n_1 is always true ($\alpha_1 = T$). According to this norm, all the agents playing the role $leader_D$ (target of link l_1) must consult the informations set available by any agent playing the role $coordinator$. The norm n_2 imposes a direct interaction in the context of mission m_4 so that the coordinator is able to get informations on the state of the transportation network. According to this norm, any agent playing the $coordinator$ role can reach any agent playing the role $leader_D$ (target of link l_1) and being a member of concrete group of type CIGT. The normative filter f_{n_2} described in the previous section expresses these constraints. For the same mission m_4 , the coordinator requires information on the available resources in the services TNM. The normative filter f_{n_3} resulting from the activation of norm n_3 , enables the coordinator to reach any leader of each traffic network management service (TNM).

In our example, if the missions m_1 , m_4 are under examination, the normative filters corresponding to the three norms are simultaneously present in the environment. From the point of view of the agent playing the role $coordinator$, it means that it can route messages directly to the agents who are $leader_D$ in groups of type CIGT (n_2) and broaden their demand to agents playing the role $leader_D$ in the groups of type TNM (n_3) given its needs.

Let's turn to the agents playing the role $leader_D$ in the group Decision-making. If involved in the role $leader_S$ within the groups CIGT and TNM (let's notice that this situation is possible thanks to the compatibility link between both roles), the agents will receive the requests from the agent playing the role $coordinator$ and will be able to know that this is a direct interaction issued from the coordinator. The agents will be able to answer to this agent by using the normative filter created in case of bidirectional interaction. Thanks to norm n_1 , every agent playing the role $leader_D$ will receive the messages sent by the agents playing

the role `coordinator` via the filter f_{n_1} , building a common and shared knowledge (indirect interaction). According to their processing activity, the agents will be more or less aware of these messages.

This short example that we can't detail more, shows the richness of expressiveness of the interaction modes made possible by combining EASI and MOISE as described in this paper.

6 Related work

To our knowledge, there doesn't exist a similar support to interaction enabling, for the same communication, to consider simultaneously the direct and indirect interaction modes.

Considering related works to the indirect interaction, the general principle consists in the use of a shared data space that is integrated or not to the environment [8]. In this approach, the tuples that are deposit by the sender in the shared space are compared to patterns expressing the needs of the receivers. These works are focused on the accompanying coordination language and don't consider, at any moment, the organization or the state of the agents.

Dealing with the direct interaction model, several works propose to use an organizational structure in order to manage the communications. In [1], the agents are organised in a hierarchy where each level knows the skills of the agents belonging to the lower level in order to make possible for the sender, a routing of the messages according to the skills. However, it is not an organizational model that is usable by the agents. In the AGR model [3], the organization constrains the interactions according to the groups to which the agents participate. It supports a routing of the message according to the organizational model (group, role). However, the only interaction mode is the direct one and the agents don't have access to an explicit description of the different specifications.

Normative organization models have been proposed in the literature in order to regulate and control the communication between agents. However the specifications address the interaction protocols, i.e. the coordination of the interaction instead of interaction modes. The only considered interaction mode is the direct one (e.g. ISLANDER [2]). They don't consider the interaction at the level addressed in this paper.

7 Conclusions

In this paper, we have proposed a specification of interaction modes between agents within an organization. For that aim, we have extended and enriched the organization modeling language of the MOISE framework. We have also shown how the specifications have been used to generate and to configure dynamically the communication environment supported by the EASI platform. We have illustrated the use of this proposal in a crisis management application.

In the future, we intend to extend the considered interaction modes to over-hearing. We will also consider the communication between groups by extending

the scope of communication to groups by enriching and modifying the structural specification of *MOISE*. Thanks to these new primitives in the organization specification, we can turn to the development of reasoning mechanisms at the agent level to make agents able to reason on the interaction modes that they can use within the organization.

Acknowledgement

We would like to thank D. Trabelsi, H. Hadioui and J.F. Hübner for the fruitful discussions about the content of this paper.

References

1. N. Bensaïd and P. Mathieu. A hybrid architecture for hierarchical agents. In *Proc. of ICCIMA '97*, pages 91–95, 1997.
2. M. Esteva, J. A. Rodriguez-Aguiar, C. Sierra, P. Garcia, and J. L. Arcos. On the formal specification of electronic institutions. In Frank Dignum and Carles Sierra, editors, *Proceedings of the Agent-mediated Electronic Commerce*, LNAI 1191, pages 126–147, Berlin, 2001. Springer.
3. J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agents systems. In Y. Demazeau, editor, *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pages 128–135. IEEE Press, 1998.
4. J. F. Hübner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents. *Journal of Autonomous Agents and Multi-Agent Systems*, 20(3), 2010.
5. J. F. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In G. Bittencourt and G. L. Ramalho, editors, *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, volume 2507 of *LNAI*, pages 118–128, Berlin, 2002. Springer.
6. Jomi Fred Hübner, Olivier Boissier, and Rafael H. Bordini. Normative programming for organisation management infrastructures. COIN Workshop at MALLOW, 2009.
7. J. Saunier and F. Balbo. Regulated multi-party communications and context awareness through the environment. *Journal on Multi-Agent and Grid Systems*, 5(1):75–91, 2009.
8. L. Tummolini, C. Castelfranchi, A. Ricci, M. Viroli, and A. Omicini. "exhibitionists" and "voyeurs" do it better: A shared environment approach for flexible coordination with tacit messages. In *Proc. of Workshop on Environments for Multi-Agent Systems*, LNAI 3374, pages 215–231. Springer Verlag, 2004.

Norm Refinement and Design through Inductive Learning^{*}

Domenico Corapi¹, Marina De Vos², Julian Padget²,
Alessandra Russo¹, and Ken Satoh³

¹ Department of Computing, Imperial College London
{d.corapi, a.russo}@imperial.ac.uk

² Department of Computer Science, University of Bath
{mdv, jap}@cs.bath.ac.uk

³ National Institute of Informatics
ksatoh@nii.ac.jp

Abstract. In the physical world, the rules governing behaviour are debugged by observing an outcome that was not intended and the addition of new constraints intended to prevent the attainment of that outcome. We propose a similar approach to support the incremental development of normative frameworks (also called institutions) and demonstrate how this works through the validation and synthesis of normative rules using model generation and inductive learning. This is achieved by the designer providing a set of *use cases*, comprising collections of event traces that describe how the system is used along with the desired outcome with respect to the normative framework. The model generator encodes the description of the current behaviour of the system. The current specification and the traces for which current behaviour and expected behaviour do not match are given to the learning framework to propose new rules that revise the existing norm set in order to inhibit the unwanted behaviour. The elaboration of a normative system can then be viewed as a semi-automatic, iterative process for the detection of incompleteness or incorrectness of the existing normative rules, with respect to desired properties, and the construction of potential additional rules for the normative system.

1 Introduction

Norms and regulations play an important role in the governance of human society. Social rules such as laws, conventions and contracts prescribe and regulate our behaviour, however it is possible for us to break these rules at our discretion and face the consequences. By providing the means to describe and reason about norms in a computational context, normative frameworks (also called institutions or virtual organisations) may be applied to software systems allowing for automated reasoning about the consequences of socially acceptable and unacceptable behaviour, by monitoring the permissions, empowerment and obligations of the participants and generating violations when norms are not followed.

^{*} This work is partially supported through the EU Framework 7 project *ALIVE (FP7-IST-215890)*, and the EPSRC PRiMMA project (EP/F023294/1).

The formal model put forward in [9] and its corresponding operationalisation through Answer Set Programming (ASP) [3, 18] aims to support the top-down design of normative frameworks. *AnsProlog* is a knowledge representation language that allows the programmer to describe a problem and required properties on the solutions in an intuitive way. Programs consist of rules interpreted under the answer set semantics. Answer set solvers, like CLASP[17] or SMODELS[25], can be used to reason about the given *AnsProlog* specification, by returning *acceptable solutions* in the form of traces, as answer sets. In a similar way, the correctness of the specification with respect to given properties can be verified.

Currently, the elaboration of behavioural rules and norms is an error-prone process that relies on the manual efforts of the designer and would, therefore, benefit from automated support. In this paper, we present an inductive logic programming (ILP) [24] approach for the extraction of norms and behaviour rules from a set of use cases. The approach is intended as a design support tool for normative frameworks. Complex systems are hard to model and even if testing of properties is possible, sometimes it is hard to identify missing or incorrect rules. In some cases, e.g. legal reasoning, the abstract specification of the system can be in part given in terms of specific instances and use cases that ultimately drive the design process and are used to assess it. We propose a design support tool that employs *use-cases*, i.e. traces together with their expected normative behaviour, to assist in the revision of a normative framework. The system is correct when none of the traces are considered *disfunctional*, i.e. they match the expected normative behaviour. When a disfunctional trace is encountered the normative specification needs to be adjusted: the task is to refine the given description by learning missing norms and/or behavioural rules that, added to the description, entail the expected behaviour over the traces. We show how this task can be naturally represented as a non-monotonic ILP problem in which the partial description of the normative system provides the background knowledge and the expected behaviour comprises the examples. In particular, we show how a given *AnsProlog* program and traces can be reformulated into an ILP representation that makes essential use of negation in inducing missing parts of the specification. As the resulting learning problem is inherently non-monotonic, we use a non-monotonic ILP system, called TAL [12], to compute the missing specification from the traces and the initial description.

Given the declarative nature of ASP, the computational paradigm used for our normative frameworks, we needed to adopt a declarative learning approach as we aim to learn declarative specifications. This differs from other approaches, such as reinforcement learning whereby norms or policies are learned as outcomes of estimation and optimisation processes. Such types of policies are not directly representable in a declarative format and are quite different in nature from the work reported here.

The paper is organised as follows. Section 2 presents some background material on the normative framework, while Section 3 introduces the non-monotonic ILP system used in our proposed approach. Section 4 describes the *AnsProlog* modelling of normative frameworks. Section 5 illustrates how the revision task can be formulated into an ILP problem, and how the generated ILP hypothesis can be reformulated as norms and behaviour rules within the *AnsProlog* representation. In Section 6 we illustrate the flexibility and expressiveness of our approach through a number of different par-

tial specifications of a reciprocal file sharing normative framework. Section 7 relates our approach to existing work on learning norms with respects to changing/improved requirements. We conclude with a summary and remarks about future work.

2 Normative Frameworks

The concept of normative framework has become firmly embedded in the agent community as a necessary foil to the essential autonomy of agents, in just the same way as societal conventions and legal frameworks have grown up to constrain people. In both the physical and the virtual worlds, and the emerging combination of the two, the arguments in favour centre on the minimisation of disruptive behaviour and supporting the achievement of the goals for which the normative framework has been conceived and thus also the motivation for submission to its governance by the participants. While the concept remains attractive, its realisation in a computational setting remains a subject for research, with a wide range of existing logics [29, 1, 7, 9, 32] and tools [26, 14, 19].

2.1 Formal Model

To provide context for this paper, we give an outline of a formal event-based model for the specification of normative frameworks that captures all the essential properties, namely empowerment, permission, obligation and violation. Extended presentations appear in [9] and [10].

The essential elements of our normative framework are: (i) events (\mathcal{E}), that bring about changes in state, and (ii) fluents (\mathcal{F}), that characterise the state at a given instant. The function of the framework is to define the interplay between these concepts over time, in order to capture the evolution of a particular institution through the interaction of its participants. We distinguish two kinds of events: normative events (\mathcal{E}_{norm}), that are the events defined by the framework and exogenous (\mathcal{E}_{ex}), that are outside its scope, but whose occurrence triggers normative events in a direct reflection of the “counts-as” principle [21]. We further partition normative events into normative actions (\mathcal{E}_{act}) that denote changes in normative state and violation events (\mathcal{E}_{viol}), that signal the occurrence of violations. Violations may arise either from explicit generation, from the occurrence of a non-permitted event, or from the failure to fulfil an obligation. We also distinguish two kinds of fluents: *normative fluents* that denote normative properties of the state such as permissions \mathcal{P} , powers \mathcal{W} and obligations \mathcal{O} , and *domain fluents* \mathcal{D} that correspond to properties specific to the normative framework itself. The set of all fluents is denoted as \mathcal{F} . A normative state is represented by the fluents that hold true in this state. Fluents that are not presented are considered to be false. Conditions on a state are therefore expressed by a set of fluents that should be true or false. The set of possible conditions is referred to as $\mathcal{X} = 2^{\mathcal{F} \cup \neg\mathcal{F}}$.

Changes in state are achieved through the definition of two relations: (i) the generation relation, which implements counts-as by specifying how the occurrence of one (exogenous or normative) event generates another (normative) event, subject to the empowerment of the actor and the conditions on the state, and (ii) the consequence relation. This latter specifies the initiation and termination of fluents subject to the performance of some action in a state matching some expression. The generation relation is formally

defined as $\mathcal{G} : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{E}_{norm}}$, and the consequence relation as $\mathcal{C} : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{F}} \times 2^{\mathcal{F}}$. The fluents to be initiated as a result of an event E are often denoted by $\mathcal{C}^\uparrow(\phi, E)$ while the ones to be terminated are denoted by $\mathcal{C}^\downarrow(\phi, E)$.

The semantics of our normative framework is defined over a sequence, called a *trace*, of exogenous events. Starting from the initial state, each exogenous event is responsible for a state change, through initiation and termination of fluents. This is achieved by a three-step process: (i) the transitive closure of \mathcal{G} with respect to a given exogenous event determines all the generated (normative) events, (ii) to this all violations of events not permitted and obligations not fulfilled are added, giving the set of all events whose consequences determine the new state, (iii) the application of \mathcal{C} to this set of events identifies all fluents that are initiated and terminated with respect to the current state so giving the next state. For each trace, we can therefore compute a sequence of states that constitutes the model of the normative framework for that trace. This process is realised as a computational model through Answer Set Programming (see Section 4) and it is this representation that is the subject of the learning process described in Section 5.

3 Learning

Inductive Logic Programming (ILP) [24] is a machine learning technique concerned with the induction of logic theories from (positive and negative) examples and has been successfully applied to a wide range of problems [15]. Automatic induction of hypotheses represented as logic programs is one of the distinctive features of ILP. Moreover, the use of logic programming as representation language allows a principled representation of background information relevant to the learning. To refine normative theories we employ an ILP learning system, called TAL [12], that is able to learn non-monotonic theories, and can be employed to perform learning of new rules and the revision of existing rules. The TAL approach is based on mapping a given inductive problem into an abductive reasoning process. The current implementation of TAL relies on an extension of the abductive procedure SLDNFA [13] and preserves its semantics.

Definition 1. A non-monotonic ILP task is defined as $\langle E, B, S \rangle$ where E is a set of ground positive or negative literals, called examples, B is a background normal theory and S is a set of clauses called language bias. The normal theory $H \in ILP\langle E, B, S \rangle$, called hypothesis, is an inductive solution for the task $\langle E, B, S \rangle$, if $H \subseteq S$, H is consistent with B and $B \cup H \models E$.

B and H are normal theories and thus support negation as failure. The choice of an appropriate language bias is critical. In TAL the language bias S is specified by means of *mode declarations* [?].

Definition 2. A mode declaration is either a head or body declaration, respectively $modeh(s)$ and $modeb(s)$ where s is called a scheme. A scheme s is a ground literal containing place-markers. A place-marker is a ground function whose functor is one of the three symbols '+' (input), '-' (output), '#' (constant) and the argument is a constant called type.

Given a schema s , s^* is the literal obtained from s by replacing all place-markers with different variables X_1, \dots, X_n . A rule r is *compatible* with a set M of mode declarations iff (a) there is a mapping from each head/body literal l in r to a head/body declaration $m \in M$ with schema s such that each literal is subsumed by its corresponding s^* ; (b) each output place-marker is bound to an *output variable*; (c) each input place-marker is bound to an output variable appearing in the body or to a variable in the head; (d) every constant place-marker is bound to a constant; (e) all variables and constants are of the corresponding type. From a user perspective, mode declarations establish how rules in the final hypotheses are structured, defining literals that can be used in the head and in the body of a well-formed hypothesis. Although we show M in the running example of this paper for reference, the mode declarations can be concealed from the user and derived automatically. They can be optionally refined to constrain the search whenever the designer wants to employ useful information on the outcome of the learning to reduce the number of alternative hypotheses or improve performance.

4 Modelling Normative Frameworks

While the formal model of a normative framework allows for clear specification of a normative system, it is of little support to designers or users of these systems. In order to be able to do so, computational tools are needed. The first step is a computational model equivalent to the formal model. We have opted for a form of logic programming, called Answer Set Programming (ASP)[18]. Here we only present a short flavour of the language *AnsProlog*, and the interested reader is referred to [3] for in-depth coverage.

AnsProlog is a knowledge representation language that allows the programmer to describe a problem and the requirements on the solutions in an intuitive way, rather than the algorithm to find the solutions to the problem. The basic components of the language are atoms, elements that can be assigned a truth value. An atom can be negated using *negation as failure* so creating the literal *not a*. We say that *not a* is true if we cannot find evidence supporting the truth of a . If a is true then *not a* is false and vice versa. Atoms and literals are used to create rules of the general form: $a \leftarrow B, \text{not } C$, where a is an atom and B and C are set of atoms. Intuitively, this means *if all elements of B are known/true and no element of C is known/true, then a must be known/true*. We refer to a as the head and $B \cup \text{not } C$ as the body of the rule. Rules with empty body are called *facts*; A program in *AnsProlog* is a finite set of rules.

The semantics of *AnsProlog* are defined in terms of *answer sets*, i.e. assignments of true and false to all atoms in the program that satisfy the rules in a minimal and consistent fashion. A program has zero or more answer sets, each corresponding to a solution.

4.1 Mapping the formal model into *AnsProlog*

In this section we only provide a summary description of how the formal institutional model is translated in to *AnsProlog*. A full description of the model can be found in [9] together with completeness and correctness of model with respect to traces. Each program models the semantics of the normative framework over a sequence of n time

instants such that $t_i : 0 \leq i \leq n$. Events are considered to occur *between* these snapshots, where for simplicity we do not define the intervals at which events occur explicitly, and instead refer to the time instant at the start of the interval at which an event is considered to occur. Fluents may be true or false at any given instant of time, so we use atoms of the form $\text{holdsat}(f, t_i)$ to indicate that fluent f holds at time instant t_i . In order to represent changes in the state of fluents over time, we use atoms of the form $\text{initiated}(f, t_i)$ and $\text{terminated}(f, t_i)$ to denote the fact that fluent f was initiated or terminated, respectively, *between* time instants i and $i + 1$. We use atoms of the form $\text{occurred}(e, t_i)$ to indicate that event $e \in \mathcal{E}$ is considered to have occurred between instant t_i and t_{i+1} . These atoms denote events that occur in an external context or are generated by the normative framework. For exogenous events we additionally use atoms of the form $\text{observed}(e, t_i)$ to denote the fact that e has been observed.

The mapping of a normative framework consists of three parts: a base component which is independent of the framework being modelled, the time model and the framework specific component. The independent component deals with inertia of the fluents, the generation of violation events of un-permitted actions and unsatisfied obligations. The time model defines the predicates for time and is responsible for generating a single observed event at every time instance. In this paper we will focus solely on the representation of the specific features of the normative framework.

In order to translate rules in the normative framework relations \mathcal{G} and \mathcal{C} , we must first define a translation for expressions which may appear in these rules. The valuation of a given expression taken from the set \mathcal{X} depends on which fluents may be held to be true or false in the current state (at a give time instant). We translate expressions into ASP rule bodies as conjunctions of extended literals using negation as failure for negated expressions.

With all these atoms defined, mapping the generation function and the consequence relation of a specific normative framework becomes rather straightforward. The generation function specifies that an normative event e occurs at a certain instance ($\text{occurred}(e, t)$) when an another event e' occurs, the event e is empowered ($\text{holdsat}(\text{pow}(e), t)$) and a set of conditions on the state are satisfied ($\text{holdsat}(f, t)$ or **not** $\text{holdsat}(f, t)$). The rules for initiation ($\text{initiated}(f, t)$) and termination ($\text{terminated}(f, t)$) of a fluent f are triggered when a certain event e occurs ($\text{occurred}(e, t)$) and a set of conditions on the state are fulfilled. The initial state of our normative framework is encoded as simple facts ($\text{holdsat}(f, i00)$).

Figure 1 gives a summary of all *AnsProlog* rules that are generated for a specific normative framework, including the definition of all the fluents and events as facts. For a given expression $\phi \in \mathcal{X}$, we use the term $EX(\phi, T)$ to denote the translation of ϕ into a set of ASP literals of the form $\text{holdsat}(f, T)$ or **not** $\text{holdsat}(f, T)$.

In situations where the normative system consists of a number of agents whose actions can be treated in the same way (e.g. the rules for borrowing a book are the same for every member of alibrary) or where the state consists of fluents that can be treated in a similar way (e.g. the status of book), we can parameterise the events and fluents. This is represented in the *AnsProlog* program by function symbols (e.g. $\text{borrowed}(\text{Agent}, \text{Book})$) rather than terms. To allow for grounding, extra atoms to

$$\begin{aligned}
p \in \mathcal{F} &\Leftrightarrow \text{ifluent}(p). \\
e \in \mathcal{E} &\Leftrightarrow \text{event}(e). \\
e \in \mathcal{E}_{ex} &\Leftrightarrow \text{evtype}(e, \text{obs}). \\
e \in \mathcal{E}_{act} &\Leftrightarrow \text{evtype}(e, \text{act}). \\
e \in \mathcal{E}_{viol} &\Leftrightarrow \text{evtype}(e, \text{viol}). \\
\mathcal{C}^\uparrow(\phi, e) = P &\Leftrightarrow \forall p \in P \cdot \text{initiated}(p, T) \leftarrow \text{occurred}(e, I), EX(\phi, T). \\
\mathcal{C}^\downarrow(\phi, e) = P &\Leftrightarrow \forall p \in P \cdot \text{terminated}(p, T) \leftarrow \text{occurred}(e, I), EX(\phi, T). \\
\mathcal{G}(\phi, e) = E &\Leftrightarrow g \in E, \text{occurred}(g, T) \leftarrow \text{occurred}(e, T), \\
&\quad \text{holdsat}(\text{pow}(e), I), EX(\phi, T). \\
p \in S_0 &\Leftrightarrow \text{holdsat}(p, i00).
\end{aligned}$$

Fig. 1. The translation of normative framework specific rules into *AnsProlog*

ground these variables need to be added. Grounded versions of the atoms also need to be added to the program. An example of this can be found in Section 6.

5 Learning Normative Rules

5.1 Methodology

The development process is supported by a set of *use cases* U . Use cases represent instances of executions that are known to the designer and that drive the elaboration of the normative system. If the current formalisation of the system does not match the intended behaviour in the use case then the formalisation is still not complete or incorrect. Each use case $u \in U$ is a tuple $\langle T, C, O \rangle$ where T is a *trace* that specifies *all* the exogenous events occurring at all the time points considered ($\text{observed}(e, T)$); C are ground holdsat or occurred facts that the designer believes to be important and represents the *conditional expected output*; O are ground holdsat and occurred literals that represent the *expected output* of the use case.

The design process is iterative. A current formalisation of the model in *AnsProlog* is tested against a set of use cases. Together with the *AnsProlog* specification of the normative framework we add the observed events and a constraint indication that no answer set that does not satisfy O is acceptable. The latter is done by adding a constraint containing the negation of all the elements in O . If for some use cases the solver is not able to find an answer set (returns unsatisfiable), then a revision step is performed. All the use cases and the current formalisation are given as input to *TAL*. Possible revisions are provided to the designer who ultimately chooses which is the most appropriate. The success of the revision step depends on the state of the formalisation of the model. The set of supporting use cases can be extended as the design progresses to more accurate models.

In this paper we focus on the learning step and we show how a non-monotonic ILP system can be used to derive new rule. Refining existing rules (i.e. deleting rules or adding and delete conditions in rules) is a straightforward extension of the current framework. Though we do not discuss it in this paper, revision can be performed by extending the original rules with additional predicates that extend the search to deletion of conditions in rules and to exceptions as shown in [11].

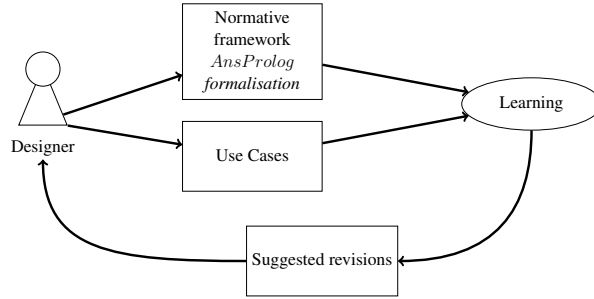


Fig. 2. Iterative design driven by use cases.

5.2 Mapping ASP to ILP

The differences between the *AnsProlog* program and the translation into a suitable representation for *TAL* is procedural and only involves syntactic transformations. Thus the difference in the two representations only consists in how the inference is performed. The two semantics coincide since the same logic program is encoded and the mapping of a normative framework has exactly one answer set when given a trace. If conditions are added this can be reduced to zero.

A normative model \mathcal{F} corresponds to a *AnsProlog* program $P_{\mathcal{F}}$ as described in Section 4. All the normal clauses contained in $P_{\mathcal{F}}$ are part of B ; the only differences involve time points, that are handled in B by means of a finite domain constraint solver. B also contains all the facts in C and T (negated facts are encoded by adding exceptions to the definitions of `holdsat` and `occurred`). The set of examples E contains the literals in O . Each $H \in ILP(E, B, S)$ represents a possible revision for \mathcal{P} and thus for the original normative model.

6 Example

To illustrate the capabilities of the norm learning mechanism, we have developed a relatively simple scenario that, at the same time, is complicated enough to demonstrate the key properties with little extraneous detail.

The active parties—agents—of the scenario each find themselves initially in the situation of having ownership of several (digital) objects—the blocks—that form part of some larger composite (digital) entity—a file. An agent may give a copy of one its blocks in exchange for a copy of another block with the aim of acquiring a complete set of all the blocks. For simplicity, in the situation we analyse here, we assume that initially each agent holds the only copy of a given block, and that there is only one copy of each block in the agent population. Furthermore, we do not take into account the possibility of exchanging a block for one that the agent already has. We believe that neither of these issues does more than complicate the situation by adding more states that would obscure the essential properties that we seek to demonstrate. Thus, we arrive at a statement of the example: two agents, Alice and Bob, each holding two blocks from a set of four and each having the goal of owning all four by downloading the blocks they miss from the other while sharing, with another agent, the ones it does.

```

% Normative and Domain Rules
initiated(hasBlock(Agent, Block), I) ←
    occurred(myDownload(Agent, Block), I), holdsat(live(filesharing), I).
initiated(perm(myDownload(Agent, Block)), I) ←
    occurred(myShare(Agent), I), holdsat(live(filesharing), I).
terminated(pow(filesharing, myDownload(Agent, Block)), I) ←
    occurred(myDownload(Agent, Block), I), holdsat(live(filesharing), I).
terminated(needsBlock(Agent, Block), I) ←
    occurred(myDownload(Agent, Block), I), holdsat(live(filesharing), I).
terminated(pow(filesharing, myDownload(Agent, Block)), I) ←
    occurred(misuse(Agent), I), holdsat(live(filesharing), I).
terminated(perm(myDownload(Agent, Block)), I) ←
    occurred(myDownload(Agent, Block), I), holdsat(live(filesharing), I).
occurred(myDownload(AgentA, Block), I) ←
    occurred(download(AgentA, AgentB, Block), I), holdsat(hasBlock(AgentB, Block), I),
    holdsat(pow(filesharing, myDownload(AgentA, Block)), I), AgentA! = AgentB.
occurred(myShare(AgentB), I) ←
    occurred(download(AgentA, AgentB, Block), I), holdsat(hasBlock(AgentB, Block), I),
    holdsat(pow(filesharing, myDownload(AgentA, Block)), I), AgentA! = AgentB.
occurred(misuse(Agent), I) ← occurred(viol(myDownload(Agent, Block)), I), i).

% Initial state
holdsat(pow(filesharing, myDownload(Agent, Block)), i0).
holdsat(pow(filesharing, myShare(Agent)), i0).
holdsat(perm(download(AgentA, AgentB, Block)), i0).
holdsat(perm(myDownload(Agent, Block)), i0).
holdsat(perm(myShare(Agent)), i0).
holdsat(hasBlock(alice, x1), i0).      holdsat(hasBlock(alice, x2), i0).
holdsat(hasBlock(bob, x3), i0).      holdsat(hasBlock(bob, x4), i0).
holdsat(needsBlock(alice, x3), i0).   holdsat(needsBlock(alice, x4), i0).
holdsat(needsBlock(bob, x1), i0).    holdsat(needsBlock(bob, x2), i0).
holdsat(live(filesharing), i0).

% fluent rules
holdsat(P, J) ← holdsat(P, I), not terminated(P, I), next(I, J).
holdsat(P, J) ← initiated(P, I), next(I, J).
occurred(E, I) ← evtype(E, ex), observed(E, I).
occurred(viol(E), I) ←
    occurred(E, I), not holdsat(perm(E), I), holdsat(live(X), I), evinst(E, X).
occurred(viol(E), I) ←
    occurred(E, I), evtype(E, inst), not holdsat(perm(E), I), event(viol(E)).

```

Fig. 3. Translation of the “sharing” normative framework into *AnsProlog* (types omitted).

We model this as a simple normative framework, where the brute event [20] of downloading a block initiates several normative events, but the act of downloading revokes the permission of that agent to download another block until it has shared (this the complementary action to download) a block with another agent. Violation of this norm results in the download power being revoked permanently. In this way reciprocity is assured by the normative framework. Initially, each agent is empowered and permitted to share and to download, so that either agent may initiate a download operation.

Fig. 3 shows the *AnsProlog* representation of the complete normative framework representing this scenario. In the following examples a variety of normative rules will be deliberately removed and re-learned.

6.1 Learning Setting

To show how different parts of the formal model can be learned we start from a correct specification and, after deleting some of the rules, we use TAL to reconstruct the missing parts based on a single use case. In our example TAL is set to learn hypotheses of at most three rules with at most three conditions. The choice of an upper bound on

the complexity (number of literals) of the rule ultimately rests on the final user. Alternatively, TAL can iterate on the complexity or perform a best first search that returns increasingly more complex solutions. We use the following mode declarations, M :

```

m1 : modeh(terminated(perm(myDownload(+agent, +block)), +instant)).
m2 : modeh(initiated(perm(myDownload(+agent, +block)), +instant)).
m3 : modeb(occurred(myDownload(+agent, +block), +instant)).
m4 : modeb(occurred(myDownload(+agent, -block), +instant)).
m5 : modeb(occurred(myShare(+agent), +instant)).
m6 : modeb((+agent!= +agent)).
m7 : modeb(holdsat(hasblock(+agent, +block), +instant)).
m8 : modeb(holdsat(pow filesharing(myDownload(+agent, +block)), +instant)).

```

The first two mode declarations state that terminate and initiate permission rules for the normative fluent *myDownload* can be learned. The other declarations constrain the structure of the body. The difference between $m3$ and $m4$ is that the former must refer to the same block as the one in the head of the rule while the latter introduces a possibly different block. $m8$ is an inequality constraint between agents. In general more mode declarations should be considered (e.g. initiation and termination of all types of fluents should be included) but the revision can be guided by the designer. For example new changes to a stable theory are more likely to contain errors and thus can be isolated in the revision process. The time to compute all the reported hypotheses ranges from 30 to 500 milliseconds on a 2.8 GHz Intel Core 2 Duo iMac with 2 GB of RAM.

The background knowledge B contains the rules in Fig. 3 together with the traces T given in the use cases. C in this example is empty to allow for the demonstration of the most general types of learning.

Learning a single terminate/initiate rule We suppose one of the *initiate* rules is missing from the current specification:

$$\textit{initiated}(\textit{perm}(\textit{myDownload}(\textit{Agent}, \textit{Block})), I) \leftarrow \textit{occurred}(\textit{myShare}(\textit{Agent}), I).$$

The designer inputs the following observed events that show how in a two agent scenario, one of the agents loses permission to download after downloading a block and reacquires it after providing a block for another agent. The trace T looks like:

$$\begin{aligned} &\textit{observed}(\textit{download}(\textit{alice}, \textit{bob}, x3), 0). \\ &\textit{observed}(\textit{download}(\textit{bob}, \textit{alice}, x1), 1). \end{aligned}$$

The expected output O is:

$$\begin{aligned} &\textit{not holdsat}(\textit{perm}(\textit{myDownload}(\textit{alice}, x4)), 1). \\ &\textit{holdsat}(\textit{perm}(\textit{myDownload}(\textit{alice}, x4)), 2). \end{aligned}$$

The trace is disfunctional if the expected output is not true in the answer set of $T \cup B$. The *ILLP* task is thus to find a set of rules H within the language bias specified by mode declarations in M such that given the background knowledge B in Fig. 3 and the

given expected output O as conjunction of literals, O is true in the only answer set of $B \cup T \cup H$ (if one exists). TAL produces the following hypotheses:

$$\begin{aligned} &initiated(perm(myDownload(A, -)), C) \leftarrow (H_1) \\ &occurred(myShare(A), C). \end{aligned}$$

and

$$\begin{aligned} &terminated(perm(myDownload(-, -)), -). (H_2) \\ &initiated(perm(myDownload(A, -)), C) \leftarrow \\ &occurred(myShare(A), C). \end{aligned}$$

The second solution is not the one intended but it still supports the use case. Note that according to current implementation, whenever a fluent f is both initiated and terminated at the same time point, f still holds at the subsequent time point.

Learning multiple rules In this scenario two rules are missing from the specification:

$$\begin{aligned} &initiated(perm(myDownload(Agent, Block)), I) \leftarrow \\ &occurred(myShare(Agent), I). \\ &terminated(perm(myDownload(Agent, Block2)), I) \leftarrow \\ &occurred(myDownload(Agent, Block1), I). \end{aligned}$$

We use the same T and O as previously. TAL produces the following hypotheses:

$$\begin{aligned} &terminated(perm(myDownload(A, -)), C) \leftarrow (H_1) \\ &occurred(myDownload(A, -), C). \\ &initiated(perm(myDownload(A, -)), C) \leftarrow \\ &occurred(myShare(A), C). \\ &terminated(perm(myDownload(-, -)), -). (H_2) \\ &initiated(perm(myDownload(A, -)), C) \leftarrow \\ &occurred(myShare(A), C). \end{aligned}$$

The second solution is consistent with the use case, but the designer can easily discard it, since the rule is not syntactically valid with respect to the normative framework: a fluent can only be terminated as a consequence of the occurrence of an event. Using more advanced techniques for the language bias specification it would be possible to rule out such a hypothesis.

Learning of undesired violation We assume the following rule is missing:

$$\begin{aligned} &initiated(perm(myDownload(Agent, Block)), I) \leftarrow \\ &occurred(myShare(Agent), I). \end{aligned}$$

This time we provide a different trace T :

$$\begin{aligned} &observed(download(alice, bob, x3), 0). \\ &observed(download(bob, alice, x1), 1). \\ &observed(download(alice, bob, x4), 2). \end{aligned}$$

As a result of the trace, a violation at time point 2 is implied that the designer knows to be undesired. The expected output is:

$$not\ occurred(viol(myDownload(alice, x4)), 2).$$

$$\begin{aligned}
\text{occurred}(\text{myShare}(A), B) &\leftarrow & (H1) \\
&\text{occurred}(\text{download}(C, A, E), B), A! = C, \\
&\text{holdsat}(\text{pow}(\text{filesharing}, \text{myDownload}(A, E)), B). \\
\text{occurred}(\text{myShare}(A), B) &\leftarrow & (H2) \\
&\text{occurred}(\text{download}(C, A, E), B), A! = C, \\
&\text{holdsat}(\text{pow}(\text{filesharing}, \text{myDownload}(A, E)), B), \\
&\text{holdsat}(\text{hasblock}(A, E), B). \\
\text{occurred}(\text{myShare}(A), B) &\leftarrow & (H3) \\
&\text{occurred}(\text{download}(C, A, E), B), A! = C, \\
&\text{holdsat}(\text{pow}(\text{filesharing}, \text{myDownload}(C, E)), B). \\
\text{occurred}(\text{myShare}(A), B) &\leftarrow & (H4) \\
&\text{occurred}(\text{download}(C, A, E), B), A! = C, \\
&\text{holdsat}(\text{pow}(\text{filesharing}, \text{myDownload}(C, E)), B), \\
&\text{holdsat}(\text{hasblock}(A, E), B). \\
\text{occurred}(\text{myShare}(A), B) &\leftarrow & (H5) \\
&\text{occurred}(\text{download}(C, A, E), B), A! = C, \\
&\text{holdsat}(\text{hasblock}(A, E), B). \\
\text{occurred}(\text{myShare}(A), B) &\leftarrow & (H6) \\
&\text{occurred}(\text{download}(C, A, E), B), \\
&\text{holdsat}(\text{pow}(\text{filesharing}, \text{myDownload}(C, E)), B).
\end{aligned}$$

Fig. 4. Proposals to revise the generate rule

The outcome of the learning consists of the following two possible solutions:

$$\begin{aligned}
\text{initiated}(\text{perm}(\text{myDownload}(A, -)), C) &\leftarrow (H_1) \\
&\text{occurred}(\text{myShare}(A), C). \\
\text{initiated}(\text{perm}(\text{myDownload}(-, -)), -) & (H_2)
\end{aligned}$$

that show how the missing rule is derived from the undesired violation. As in the previous scenario the designer can easily dismiss the second candidate.

Learning a generate rule To account for the different type of rules that need to be learned, the language bias is extended to consider learning of generate rules. The new mode declarations are:

$$\begin{aligned}
&\text{modeh}(\text{occurred}(\text{myShare}(+agent), +instant)). \\
&\text{modeb}(\text{occurred}(\text{download}(-agent, +agent, -block), +instant)).
\end{aligned}$$

We use the same trace and expected output as in the previous scenario (three observed events). The following rule is eliminated from the specification:

$$\begin{aligned}
\text{occurred}(\text{myShare}(\text{AgentB}), I) &\leftarrow \\
&\text{AgentA!} = \text{AgentB}, \\
&\text{occurred}(\text{download}(\text{AgentA}, \text{AgentB}, \text{Block}), I), \\
&\text{holdsat}(\text{hasblock}(\text{AgentB}, \text{Block}), I), \\
&\text{holdsat}(\text{pow}(\text{filesharing}, \text{myDownload}(\text{AgentA}, \text{Block})), I).
\end{aligned}$$

This is the most complicated case for the designer as a set of six different hypotheses are returned by TAL (see Figure 4). Knowing the semantics of the function symbol $\text{download}(\text{AgentA}, \text{AgentB}, \text{Block})$ as AgentA downloads from AgentB the designer should be able to select the most appropriate rule.

7 Related Work

The motivation behind this paper is the problem of how to converge upon a complete and correct normative framework *with respect to the intended range of application*, where in practice these properties may be manifested by incorrect or unexpected behaviour in use. Additionally, we would observe, from practical experience with our particular framework, that it is often desirable, as with much software development, to be able to develop and test incrementally—and regressively—rather than attempt verification once the system is (notionally) complete.

The literature seems to fall broadly into three categories: (a) concrete language frameworks (OMASE, Operetta, InstSuite, MOISE, Islander, OCeAN and the constraint approach of Garcia-Camino (full references to these are currently omitted because of page limitations)) for the specification of normative systems, that are typically supported by some form of model-checking, and in some cases allow for change in the normative structure; (b) logical formalisms, such as [16], that capture consistency and completeness via modalities and other formalisms like [5], that capture the concept of norm change, or [?] and [?]; (c) mechanisms that look out for (new) conventions and handle their assimilation into the normative framework over time and subject to the current normative state and the position of other agents [2, 8]. Essentially, the objective of each of the above is to realize a transformation of the normative framework to accommodate some form of shortcoming. These shortcomings can be identified in several ways: (a) by observing that a particular state is rarely achieved, which can indicate there is insufficient normative guidance for participants, or (b) a norm conflict occurs, such that an agent is unable to act consistently under the governing norms [23], or (c) a particular violation occurs frequently, which may indicate that the violation conflicts with an effective course of action that agents prefer to take, the penalty notwithstanding. All of these can be viewed as characterising emergent [28] approaches to the evolution of normative frameworks, where some mechanism, either in the framework, or in the environment, is used to revise the norms. In the approach taken here, the designer presents use cases that effectively capture their behavioural requirements for the system, in order to ‘fix’ bad states. This has an interesting parallel with the scheme put forward by Serrano and Saugar [30], where they propose the specification of incomplete theories and their management through incomplete normative states identified as “pending”. The framework lets designated agents resolve this category through the speech acts *allow* and *forbid* and scheme is formalised using an action language.

A useful categorisation of normative frameworks appears in [6]. Whether the norms here are ‘strong’ or ‘weak’ —the first guideline— depends on whether the purpose of the normative model is to develop the system specification or additionally to provide an explicit representation for run-time reference. Likewise, in respect of the remaining guidelines, it all depends on how the framework we have developed is actually used: we have chosen, for the purpose of this presentation, to stage norm refinement so that it is an off-line (in the sense of prior to deployment) process, while much of the discussion in [6] addresses run-time issues. Whether the process we have outlined here could effectively be a means for on-line mechanism design, is something we have yet to explore.

From an ILP perspective, we employ an ILP system that can learn logic programs with negation (stratified or otherwise). Though recently introduced and in its early stages of development *TAL* is the most appropriate choice to support this work for two main reasons: it is supported by completeness results, unlike other existing non-monotonic ILP systems ([27], [22]), and it can be tailored to particular requirements (e.g. different search strategies can address performance requirements). The approach presented in this paper is related to other recently proposed frameworks for the elaboration of formal specifications via inductive learning. Within the context of software engineering, [?] has shown how examples of desirable and undesirable behaviour of a software system can be used by an ILP system, together with an incomplete background knowledge of the envisioned system and its environment, to compute missing requirements specifications. A more general framework has been proposed [?] where desirable and undesirable behaviours are generated from counterexamples produced by model checking a given (incomplete) requirements specification with respect to given system properties. The learning of missing requirements has in this case the effect of eliminating the counterexamples by elaborating further the specification.

8 Conclusions and Future Work

We have presented an approach for learning norms and behavioural rules, via inductive logic programming, from example traces in order to guide and support the synthesis of a normative framework. This addresses a crucial problem in normative systems as the development of such specifications is in general a manual and error-prone task. The approach deploys an established inductive logic programming system [12] that takes in input an initial (partial) description of a normative system and use cases of expected behaviours provided by the designer and generates hypothesis in the form of missing norms and behavioural rules that together with the given description explain the use cases. Although the approach presented in this paper has been tailored for learning missing information, it can also be applied to computing revisions over the existing description. In principle this can be achieved by transforming the existing normative rules into defeasible rules with exceptions and using the same ILP system to compute exception rules. These exceptions would in essence be prescriptions for changes (i.e. addition and/or deletion of literals in the body of existing rules) in the current specification. An appropriate refactoring of the defeasible rules based on the learned exception rules would give a revised (non-defeasible) specification. In this case, the revision would be in terms of changes over the rules of a normative framework instead of changes over its belief state, as would be the case if a TMS approach were adopted.

There are several criticisms that can be levelled at the approach as it stands. Firstly, the design language is somewhat unfriendly: a proper tool would have a problem-oriented language, like *InstAL/QL* [10, 19]. A system designer would then start from an initial description of their normative framework with some use cases and receive automated suggestions of additional norms to include in the framework written in the same high-level language. The machinery described here, based on ASP syntax and ILP formulation, would then be used as a sound “back- end” computation to a formalism familiar to the system designer. Secondly, better control is needed over the rules that are

learned and over the filtering of incorrect rules; at present this depends on specialised knowledge of the learning process. This can to some extent be controlled through careful choice of and limits on the size of use cases—probably involving heuristics—to improve the effectiveness of the learning process in the search for relevant hypotheses and pruning of those potential solutions that cannot be translated back into the canonical form of the normative framework. Despite these issues, we believe we have identified an interesting path for automating and development and debugging of practical normative specifications and perhaps, in the long term, a mechanism for on-line norm evolution.

References

1. A. Artikis, M. Sergot, and J. Pitt. Specifying electronic societies with the Causal Calculator. In *Proceedings of workshop on agent-oriented software engineering iii (aose)*, LNCS 2585. Springer, 2003.
2. Alexander Artikis. Dynamic protocols for open agent systems. In Sierra et al. [31], pages 97–104.
3. Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge Press, 2003.
4. Guido Boella, Pablo Noriega, Gabriella Pigozzi, and Harko Verhagen, editors. *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
5. Guido Boella, Gabriella Pigozzi, and Leendert van der Torre. Normative framework for normative system change. In Sierra et al. [31], pages 169–176.
6. Guido Boella, Gabriella Pigozzi, and Leendert van der Torre. Normative systems in computer science - ten guidelines for normative multiagent systems. In *Normative Multi-Agent Systems*, 2009.
7. Guido Boella and Leendert van der Torre. Constitutive Norms in the Design of Normative Multiagent Systems, City College. In *Proceedings of the Sixth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA-VI)*, June 2005.
8. George Christelis and Michael Rovatsos. Automated norm synthesis in an agent-based planning environment. In Sierra et al. [31], pages 161–168.
9. Owen Cliffe, Marina De Vos, and Julian Padget. Answer set programming for representing and reasoning about virtual institutions. In *Computational Logic for Multi-Agents (CLIMA VII)*, volume 4371 of *LNAI*, pages 60–79. Springer, May 2006.
10. Owen Cliffe, Marina De Vos, and Julian A. Padget. Embedding landmarks and scenes in a computational model of institutions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *LNCS*, pages 41–57, September 2008.
11. D. Corapi, O. Ray, A. Russo, A.K. Bandara, and E.C. Lupu. Learning rules from user behaviour. In *5th Artificial Intelligence Applications and Innovations (AIAI 2009)*, April 2009.
12. D. Corapi, A. Russo, and E. Lupu. Inductive logic programming as abductive search. In *26th International Conference on Logic Programming, Leibniz International Proceedings in Informatics*. Schloss Dagstuhl Research Online Publication Server, 2010.
13. Marc Denecker and Danny De Schreye. Sldnfa: An abductive procedure for abductive logic programs. *J. Log. Program.*, 34(2):111–167, 1998.
14. V. Dignum. *A model for organizational interaction: based on agents, founded in logic*. PhD thesis, University of Utrecht, 2004.
15. Sašo Džroski. Relational data mining applications: an overview. pages 339–360, 2000.
16. Christophe Garion, Stéphanie Roussel, and Laurence Cholvy. A modal logic for reasoning on consistency and completeness of regulations. In Boella et al. [4].

17. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-Driven Answer Set Solving. In *Proceeding of IJCAI07*, pages 386–392, 2007.
18. Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):365–386, 1991.
19. Luke Hopton, Owen Cliffe, Marina De Vos, and Julian Padget. Instql: A query language for virtual institutions using answer set programming. In *Proceedings of the 10th International Workshop on Computational Logic in Multi-Agent Systems (ClimaX)*, IFI Technical Report Series, pages 87–104, September 2009.
20. John R. Searle. *The Construction of Social Reality*. Allen Lane, The Penguin Press, 1995.
21. Andrew J.I. Jones and Marek Sergot. A Formal Characterisation of Institutionalised Power. *ACM Computing Surveys*, 28(4es):121, 1996. Read 28/11/2004.
22. Tim Kimber, Krysia Broda, and Alessandra Russo. Induction on failure: Learning connected horn theories. In *LPNMR*, pages 169–181, 2009.
23. Martin Kollingbaum, Timothy Norman, Alun Preece, and Derek Sleeman. Norm conflicts and inconsistencies in virtual organisations. In *Proceedings of COIN 2006*, volume 4386 of *LNCS*, pages 245–258. Springer, 2007.
24. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
25. I. Niemelä and P. Simons. Smodels: An implementation of the stable model and well-founded semantics for normal LP. In *LPNMR*, volume 1265 of *LNAI*, pages 420–429. Springer, July 28–31 1997.
26. Juan A. Rodriguez-Aguilar. *On the Design and Construction of Agent-mediated Institutions*. PhD thesis, Universitat Autònoma de Barcelona, 2001.
27. Chiaki Sakama. Nonmonotonic inductive logic programming. In *LPNMR*, page 62, 2001.
28. Bastin Tony Roy Savarimuthu and Stephen Cranefield. A categorization of simulation works on norms. In Boella et al. [4].
29. Marek Sergot. (C+)++: An Action Language For Representing Norms and Institutions. Technical report, Imperial College, London, August 2004.
30. Juan-Manuel Serrano and Sergio Saugar. Dealing with incomplete normative states. In *Proceedings of COIN 2009*, volume 6069 of *LNCS*. Springer, 2010. in press.
31. Carles Sierra, Cristiano Castelfranchi, Keith S. Decker, and Jaime Simão Sichman, editors. *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 1*. IFAAMAS, 2009.
32. Munindar P. Singh. A social semantics for agent communication languages. In *Issues in Agent Communication*, pages 31–45. Springer-Verlag: Heidelberg, Germany, 2000.

Norm enforceability in Electronic Institutions?

Natalia Criado¹, Estefania Argente¹, Antonio Garrido¹, Juan A. Gimeno¹,
Francesc Igual¹, Vicente Botti¹, Pablo Noriega², Adriana Giret¹

¹ DSIC, Department of Information Systems and Computation,
Universitat Politècnica de Valencia,

² IIIA, Artificial Intelligence Research Institute,
CSIC, Spanish Scientific Research Council,

{ncriado, eargente, agarridot, jgimeno, figual, vbotti, agiret}@dsic.upv.es
pablo@iia.csic.es

Abstract. Nowadays Multi-Agent Systems require more and more regulation and normative mechanisms in order to assure the correct and secure execution of the interactions and transactions in the open virtual organization they are implementing. The Electronic Institution approach for developing Multi-Agent Systems implements some enforceability mechanisms in order to control norms execution and observance. In this paper we study a complex situation in a regulated environment in which the enforceability mechanisms provided by the current Electronic Institutions implementation cannot deal appropriately with norm observance. The analyzed situation is exemplified with a specific scenario of the *mWater* regulated environment, an electronic market for water-rights transfer. After this example is presented, we extrapolate it to a more generic domain while also addressing the main issues for its application in general scenarios.

1 Introduction

In general, norms represent an effective tool for achieving coordination and co-operation among the members of a society. They have been employed in the field of Multi-Agent Systems (MAS) as a formal specification of a deontic statement that aims at regulating the actions of software agents and the interactions among them. Thus, a *Normative MAS* (NMA) has been defined in [3] as follows:

”a MAS organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms and mechanisms to deliberate about norms and detect norm violation and fulfilment.”

According to this definition, the norm enforcement problem, faced by this paper, is one of the key factors in NMA. In particular, this paper faces with the enforcement of norms inside Electronic Institutions (EIs) that simulate real scenarios. EIs [19, 22, 8] represent a way to implement interaction conventions for agents who can establish commitments in open environments.

When real life problems are modelled by means of EI some of the norms are obtained by giving a computational interpretation to real legislation. In this process we have encountered two main problems:

- *Norm Inconsistency.* Usually the set of laws created by human societies in order to regulate a specific situation are contradictory and/or ambiguous. In particular, there are situations in which there is a general law (*regulative norm* [4]) which is controlled by a local law (*procedural norm* [4]). The problem arises when this local law does not ensure compliance of the more general law. This may be due to the existence of different levels of institutions which are working in the same system [11]. Thus, an elaborated process is necessary in order to determine which norms are active in a specific moment and how they are applied. Traditional methods for implementing norms in EI, which are based on the unambiguous interpretation of norms, are not suitable to overcome this problem.
- *Norm Controlling.* Even in absence of a conflict among norms, there is still the problem of norm controlling. Norm enforcement methods inside EI are based on the observation of these activities controlled by norms. In particular, there are norms whose violation cannot be observed since they regulate situations that take place out of the institution boundaries. Thus, violations are only detectable in presence of a conflict among agents.

In this paper we focus on the enforcement of these norms, which cannot be controlled by traditional techniques. Thus, we address the question of enforceability of non-observable norms inside EIs. In order to make more clear and understandable the problem addressed by this paper, it has been exemplified in the *mWater* scenario [5]. In addition, a first solution for overcoming the *mWater* concrete problem is shown. In particular, we propose the definition of a grievance scene for allowing normative conflicts to be solved within the *mWater* institution. However, this solution can be also extrapolated to generic domains.

This paper is structured as follows: the next section provides background on norm implementation, EIs and the implementation of norms inside EIs. Then a concrete example of the problem addressed by this paper is described. Finally, discussion and future works are described.

2 Background

This section firstly reviews the main methods for ensuring norm compliance in MAS and the techniques that can be employed for implementing these methods. Then, a brief description of the Electronic Institution framework is given, as well as a discussion on how norms are implemented and enforced in this framework.

2.1 Norm Implementation in Multiagent Systems

Norms allow legal issues to be modelled in electronic institutions and electronic commerce, MAS organizations, etc. Most of the works on *norms* in MAS have been proposed from a theoretical perspective. However, several works on norms from an operational point of view have recently arisen, which are focused on giving a computational interpretation of norms in order to employ them in the

design and execution of MAS applications. In this sense, norms must be interpreted or translated into mechanisms and procedures which are meaningful for the society [14]. Methods for ensuring norm compliance are classified into two categories: (i) *regimentation* mechanisms, which consist in making the violation of norms impossible, since these mechanisms prevent agents from performing actions that are forbidden by a norm; and (ii) *enforcement* mechanisms, which are applied after the detection of the violation of some norm, reacting upon it.

In a recent work [2], a taxonomy of different techniques for implementing effectively norms is proposed. On the one hand, the regimentation of norms can be achieved by two processes: (i) *mediation*, in which both the resources and communication channels are accessed through a reliable entity which controls agent behaviours and prevents agents from deviating from ideal behaviour; and (ii) *hard-wiring*, assuming that the agents' mental states are accessible and can be modified in accordance with norms. On the other hand, norm enforcement techniques are classified according to both the observer and the enforcer entity. Norms are *self-enforced* when agents observe their own behaviour and sanction themselves. Thus, norm compliance is both observed and enforced without the need of any additional party. In situations in which those agents involved by a transaction are responsible for detecting norm compliance (i.e. *second-party* observability) norms can be enforced by: (i) the *second-party* which applies sanctions and rewards; and (ii) a third entity which is an authority and acts as an *arbiter* or *judge* in the dispute resolution process. In the case of *third-party* observability, two different mechanisms for ensuring norm compliance can be defined according to the entity which is in charge of norm enforcing: (i) *social norms* are defended by the *society* as a whole; (ii) in *infrastructural enforcement* there are infrastructural entities which are authorities in charge of *monitoring* and enforcing norms by applying sanctions and rewards.

2.2 Electronic Institutions

Electronic Institutions (EI) are computational counterparts of conventional institutions [19, 22, 8]. Institutions are, in an abstract way, a set of conventions that articulate agent interactions [20]. In practice they are identified with the group of agents, standard practices, policies and guidelines, language, documents and other resources —the organization— that make those conventions work. *Electronic Institutions* are implementations of those conventions in such a way that autonomous agents may participate, their interactions are supported by the implementation and the conventions are enforced by the system on all participants. Electronic institutions are engineered as regulated open MAS environments. These MAS are open in the sense that the EI does not control the agents' decision-making processes and agents may enter and leave the EI at their own will. EIs are regulated in four ways. First, agents are capable of establishing and fulfilling commitments inside the institution, and those correspond to commitments in the real world. Second, only interactions that comply with the conventions have any consequence in the environment. Third, interactions are

organized as repetitive activities regulated by the institution and, last, interactions, in EIs, are always speech acts.

An EI is specified through: (i) a *dialogical framework* which fixes the context of interaction by defining roles and their relationships, a domain ontology and a communication language; (ii) *scenes* that establish interaction protocols of the agents playing a given role in that scene, which illocutions are admissible and under what conditions; (iii) *performative structures* that, like the script of a play, express how scenes are interrelated and how agents playing a given role move from one scene to another, and (iv) *rules of behaviour* that regulate how commitments are established and satisfied.

The HIA model has a platform for implementation of EIs. It has a graphical specification language, ISLANDER, in which the dialogical framework, performative structures and those norms governing commitments and the pre- and post- conditions of illocutions are specified [9]. It produces an XML file that is interpreted by AMELI [10], a middleware that handles agent messages to and from a communication language, like JADE, according to the ISLANDER specification [10]. In addition, EIDE [1] includes a monitoring and debugging tool, SIMDEI that keeps track of all interactions and displays them in different modes. There is also a tool, aBuilder, that, from the XML specification, generates, for each role, agent shells that comply with the communication conventions (the decision-making code is left to the agent programmer).

2.3 Norm Implementation in EI

Norm Regimentation. In AMELI, governors filter the actions of agents, letting them only to perform those actions that are permitted by the rules of society. Therefore, governors apply a regimentation mechanism, preventing the execution of prohibited actions and, therefore, preventing agents to violate their commitments.

This regimentation mechanism employed by governors makes use of a formalism based on rules for representing constraints on agent behaviours [13]. This formalism is conceived as a “machine language” for implementing other higher level normative languages. More specifically, it has been employed to enforce norms that govern EIs. The main features of the proposed “machine language” are: (i) it allows for the explicit definition and management of agent norms (i.e. prohibitions, obligations and permissions); (ii) it is a general purpose language not aimed at supporting a specific normative language; (iii) it is declarative and has an execution mechanism. For implementing this rule system, the Jess tool has been employed as an inference engine. Jess allows the development of Java applications with “reasoning” capabilities¹.

In open systems, not only the regimentation of all actions can be difficult, but also sometimes it is inevitable and even preferable to allow agents to violate norms [6]. Reasons behind desirability of norm violations are because it is impossible to take a thorough control of all their actions, or agents could

¹ <http://herzberg.ca.sandia.gov/jess/>

obtain higher personal benefits when violating norms, or norms may be violated by functional or cooperative motivations, since agents intend to improve the organization functionality through violating or ignoring norms. Therefore, all these situations require norms to be controlled by enforcement mechanisms. Next, works on the enforcement of norms inside EI are described.

Norm Enforcement. The enforcement of a norm by an institution requires the institution to be capable of recognizing the occurrence of the violation of the norm and respond to it [14]. Hence, checking activities may occur in several ways: *directly*, at any time, randomly or with periodical checks, or by using monitoring activities; or *indirectly*, allowing agents to denounce the occurrence of a violation and then checking their grievances.

Regarding direct norm enforcement, the institution itself is in charge of both observing and enforcing norms. Thus, in this approach there are infrastructural entities which act as norm observers and apply sanctions when a violation is detected. In [17, 12], distributed mechanisms for an institutional enforcement of norms are proposed. In particular, these works propose languages for expressing norms and software architectures for the distributed enforcement of these norms. More specifically, the work described in [17] presents an enforcement mechanism, implemented by the Moses toolkit [16], which is as general (i.e. it can implement all norms that are controllable by a centralized enforcement) and more scalable and efficient with respect to centralized approaches. However, one of the main drawbacks of this proposal is the fact that each agent has an interface that sends legal messages. Since norms are controlled by these local interfaces, norms can be only expressed in terms of messages sent or received by an agent; i.e. this framework does not support the definition of norms that affect an agent as a consequence of an action carried out independently by another agent. This problem is faced by Gaertner et al. in [12]. In this approach, Gaertner et al. propose a distributed architecture for enforcing norms in EI. In particular, dialogical actions performed by agents may cause the propagation of normative positions (i.e. obligations, permissions and prohibitions). These normative propositions are taken into account by the normative level; i.e. a higher level in which norm reasoning and management processes are performed in a distributed manner. In a more recent work, Modgil et al. [18] propose an architecture for monitoring norm-governed systems. In particular, this architecture is formed by trusted observers that report to monitors on states of interest relevant to the activation, fulfilment, violation and expiration of norms. This monitoring system is *corrective* in the sense that it allows norm violations to be detected and reacting to them.

Mixed Approaches. Finally, there are works which employ a mixed approach for controlling norms. In this sense, they propose the usage of regimentation mechanisms for ensuring compliance with norms that preserve the integrity of the application. Unlike this, enforcement is proposed to control norms that cannot be regimented due to the fact that they are not verifiable or their violation

may be desirable. In [7] an example on the mixed approach is shown. In particular, this work shows how norms that define the access to the organization infrastructure are controlled, whereas norms controlling other issues such as work domain norms are ignored. In particular, those norms that define permissions and prohibitions related to the access to the organization are regimented through mediation, whereas obligation norms are enforced following the institutional sanction mechanism.

The ORA4MAS [15] is another well known proposal that makes use of a mixed approach for implementing norms. The ORA4MAS proposal defines *artifacts* as first class entities to instrument the organisation for supporting agents activities within it. *Artifacts* are resources and tools that agents can create and use to perform their individual and social activities [21]. Regarding the implementation of norms in the ORA4MAS framework, regimentation mechanisms are implemented in artifacts that agents use for accessing the organization according to the mediation mechanism. Enforcement of norms has been implemented using third party observability, since the detection of norm violations is a functionality provided by artifacts. In addition, norms are enforced by third parties, since there are agents in charge of being informed about norm violations and carrying out the evaluation and judgement of these situations.

However, none of the above mentioned proposals allows norms which regulate activities taking place out of the institution scope to be controlled. In this case, norm compliance is non-observable by the institution and can only be detected when a conflict arises. Thus, in this paper we propose that both a *second-party* and *third-party* can observe non-compliant behaviour and start a grievance process which takes place inside the EI. Therefore, in this paper we face the problem of institutional enforcement of norms based on second-party and third-party observability. Next section provides a concrete instantiation of this problem inside a more specific case-study.

3 A concrete sample scenario in the *mWater* regulated environment

In this section we exemplify the problem of non-regimented norm enforcement in EI with *mWater*, a regulated MAS application for trading water-rights within a virtual market. In order to get a good understanding of the overall *mWater* functioning, we first describe the motivation of *mWater* and present a brief overview of its structure. Afterwards, the sample complex situation for norm enforcement in the current *mWater* EI implementation is analyzed.

3.1 *mWater* overall description

In countries like Spain, and particularly in its Mediterranean coast, there is a high degree of public awareness of the main consequences of the scarcity of water and the need of fostering efficient use of water resources. Two new mechanisms

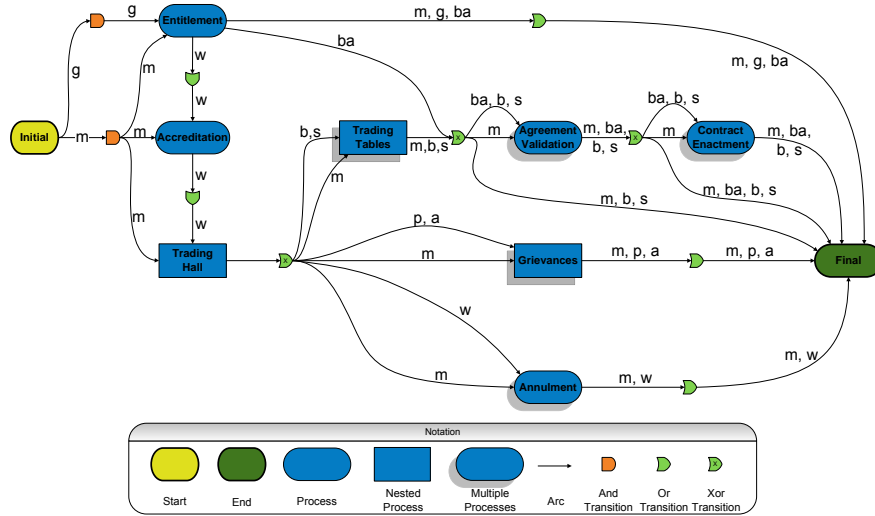


Fig. 1. *mWater* performative structure. Participating Roles: *g* - Guest, *w* - Water user, *b* - Buyer, *s* - Seller, *p* - Third Party, *m* - Market Facilitator, *ba* - Basin Authority.

for water management already under way are: a heated debate on the need and feasibility of transferring water from one basin to another, and, directly related to this proposal, the regulation of *water banks*². *mWater* is an agent-based electronic market of water-rights. Our focus is on demand and, in particular, on the type of regulatory and market mechanisms that foster an efficient use of water while preventing conflicts. The framework is a somewhat idealized version of current water-use regulations that articulate the interactions of those individual and collective entities that are involved in the use of water in a closed basin. The main focus of the work presented in this paper is on the regulated environment, which includes the expression and use of regulations of different sorts: from actual laws and regulations issued by governments, to policies and local regulations issued by basin managers, and to social norms that prevail in a given community of users.

For the construction of *mWater* we follow the IIIA *Electronic Institution* (EI) conceptual model [1]. For the actual specification and implementation of *mWater* we use the EIDE platform.

² The 2001 Water Law of the National Hidrological Plan (NHP) —‘Real Decreto Legislativo 1/2001, BOE 176’ (see www.boe.es/boe/dias/2001/07/24/pdfs/A26791-26817.pdf, in Spanish)— and its amendment in 2005 regulates the power of right-holders to engage in voluntary water transfers, and of basin authorities to setup water markets, banks, and trading centers for the exchange of water-rights in cases of drought or other severe scarcity problems.

Procedural conventions in the *mWater* institution are specified through a nested performative structure (Fig. 1) with multiple processes. The top structure, *mWaterPS*, describes the overall market environment and includes other performative structures; *TradingHall* provides updated information about the market and, at the same time, users and trading staff can initiate most trading and ancillary operations here; finally, *TradingTables* establishes the trading procedures. This performative structure includes a scene schema for each trading mechanism. Once an agreement on transferring a water-right has been reached it is "managed" according to the market conventions captured in *AgreementValidation* and *ContractEnactment* scenes. When an agreement is reached, *mWater* staff check whether the agreement satisfies some formal conditions and if so, a transfer contract is signed. When a contract becomes active, other right-holders and external stakeholders may initiate a *Grievance* procedure that may have an impact on the transfer agreement. This procedure is activated whenever any market participant believes there is an incorrect execution of a given norm and/or policy. *Grievance* performative structure includes different scenes to address such grievances or for the disputes that may arise among co-signers. On the other hand, if things proceed smoothly, the right subsists until maturity.

3.2 Complex scenario: The registration of water-right transfer agreements

In *mWater* we have three different types of regulations: (i) government norms, issued by the Spanish Ministry of Environment (stated in the National Hydrological Plan); (ii) basin or local norms, defined and regimented by the basin authorities; and (iii) social norms, stated by the members of a given user assembly and/or organization. The interplay among different norms from these three groups brings about complex situations in which there are non-regimented norms and, moreover, the non-compliance of the norm is not observable until a conflict appears. A very critical situation for the reliable execution of *mWater* appears when the following norms apply:

Government norm - (N0): A water-user can use a given volume of water from a given extraction point, if and only if he/she owns the specific water-right or has a transfer agreement that endows him/her.

Government norm - (N1): Every water-right transfer agreement must be registered within the fifteen days after its signing and wait for the Basin Authorities' approval in order to be executed.

Local norm - (N2): The registration process of a water-right transfer agreement is started voluntarily by the agreement signing parties.

Social norm - (N3): Whenever a conflict appears, a water user can start a grievance procedure in order to solve it.

Sample situation:

Let's suppose there is a water user *A* who has a water-right w_1 and wants to sell it. *A* starts a Trading Table inside the *TradingTables* process (see Fig. 1)

in order to sell w_1 . The water user B enters the Trading Table and, as a result, there is an agreement Agr_1 between A and B , by which B buys w_1 from A for the period $[t_1, t_2]$, and pays the quantity p_1 for such a transfer. A and B belong to $Basin_x$, in which norms $N1$, $N2$ and $N3$ apply. A and B do not register Agr_1 due to norm $N2$ (in other words, A and B do not go to the Agreement Validation scene of Fig. 1). Since there is no mechanism in $Basin_x$ by which water-right w_1 is blocked from A after its selling (due to Agr_1 is not registered and w_1 is still owned by A in time periods not overlapped with $[t_1, t_2]$), A continues to operate in the market. Afterwards A starts a new Trading Table to sell w_1 for period $[t_3, t_4]$, with $t_1 < t_3 < t_2$ and $t_4 > t_2$ (the new period $[t_3, t_4]$ is overlapped with $[t_1, t_2]$). In this second Trading Table A and C sign Agr_2 , by which A sells w_1 to C for the period $[t_3, t_4]$ and C pays p_2 to A . A and C belong to $Basin_x$. In this case C registers Agr_2 in the *Agreement Validation* scene, due to $N1$ and $N2$, and obtains the basin approval for executing Agr_2 . At time t_3 (the transfer starting time) C attempts to execute Agr_2 , but there is no water in the water transportation node, since B is also executing Agr_1 . At this moment C has a conflict with B , and in order to solve it he/she has to start a grievance procedure due to $N3$ (Grievances performative structure of Fig. 1).

This situation³ is an instantiated example of the one described above, in which there are non-regimented norms whose non-compliance is not observable and cannot be asserted until the conflict appears. The critical situation comes out due to the compliance procedure for agreement registration and second selling of the same water-right is not coercive.

The current development environment of EI we are using does not provide build-in support for non-coercive processes that are defined by non-regimented norms. Moreover, those situations in which it is not possible to observe the non-compliance of a norm until the resulting conflict appears are not supported either. Nevertheless, there are sample scenarios, like *mWater*, in which this behaviour is required. In the following section we analyze the EI implementation we have devised for this complex scenario.

3.3 Implementation

In this section our approach to solve the previously described complex scenario in *mWater* is described.

In order to include norm $N1$ in the current EI implementation of *mWater* we have designed the *Agreement Validation* scene (see Fig. 1) as a successor scene for any Trading Table. When any water user enters this scene, the Market Facilitator verifies the constraint of fifteen days from the agreement statement process related to norm $N1$. If this constraint is satisfied the water-right transfer agreement is forwarded to the Basin Authority who activates a Normative Reasoning

³ The scenario presented in this section happens in practice in Spain, due to the impossibility to monitor all the water transfer negotiations that may take place among the different water users. It can be considered as a loophole in the Spanish regulations. Nevertheless we are interested in modeling it due to its complexity and in order to simulate the "real" behaviour of the basin users.

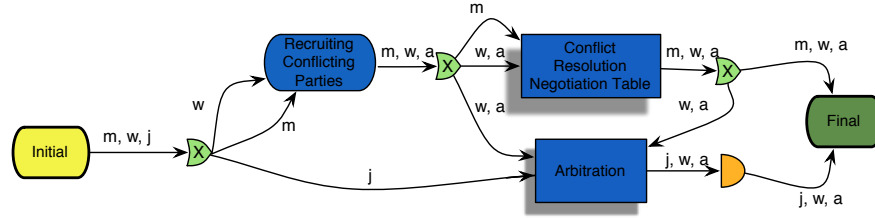


Fig. 2. Grievances performative structure

process in order to approve, or not, the agreement based on the basin normative regulation. If the agreement gets approved it is published in the Trading Hall in order for every water user of the basin to be informed of the transfer agreement.

On the other hand, norm $N2$ is automatically included in the $mWater$ institution due to the EIDE implementation feature by which no participating agent in the electronic institution can be forced to go to a given scene. For the particular $mWater$ example, neither the buyer nor the seller can be forced to go through the transition between the Trading Table scene and the Agreement Validation scene (see Fig. 1). This way, whenever the buyer and/or the seller goes to the Agreement Validation scene he/she starts the scene voluntarily, so norm $N2$ is satisfied.

The implementation of norm $N3$ requires a specific performative structure, named *Grievances* (Fig. 2), in order to deal with conflict resolution processes.

Finally, the observance of norm compliance is delegated to every water user. Hence, the enforceability of norm $N0$ is delegated to every water user.

Fig. 2 shows the different scenes of the complex Grievances performative structure. In this structure any conflict can be solved by means of two alternative processes (these processes are similar to those used in Alternative Dispute Resolutions and Online Dispute Resolutions [23, 24]). On the one hand, conflict resolution can be solved by means of negotiation tables (Conflict Resolution Negotiation Table performative structure). In this mechanism a negotiation table is created on demand whenever any water user wants to solve a conflict with other/s water user/s, negotiating with them with or without mediator. Such a negotiation table can use a different negotiation protocol, such as face to face, standard double auction, etc. On the other hand, arbitration mechanisms for conflict resolution can also be employed (Arbitration performative structure). In this last mechanism, a jury solves the conflict sanctioning the offenses.

There are three steps in the arbitration process (see Fig. 3). In the first one, the grievance is stated by the plaintive water user. In the second step, the different conflicting parties present their allegations to the jury. Finally, in the last step, the jury, after hearing the dispute, passes a sentence on the conflict. The difference among the two mechanisms for conflict resolution is that the arbitration process is binding, meanwhile the negotiation is not. In this way if

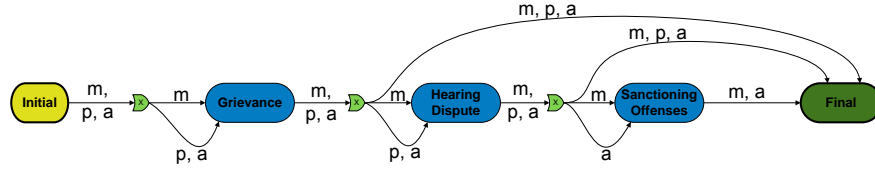


Fig. 3. Arbitration performative structure

any of the conflicting parties is not satisfied with the negotiation results he/she can activate an arbitration process in order to solve the conflict.

In the previously described complex scenario, when C cannot execute Agr_2 (because there is no water in the water transportation node), C believes that B is not complying norm $N0$. C believes there is a conflict because Agr_2 endows him/her to use the water, and moreover, there is no transfer agreement published in the Trading Hall that endows B to do the same. In order to enforce norm $N0$ and to execute Agr_2 , C starts a grievance procedure. In this procedure, water users C and B are recruited as conflicting parties and A as third party because he/she is the seller of w_1 as stated in Agr_2 (Recruiting Conflicting Parties scene of Fig. 2). Let's assume C chooses as conflict resolution mechanism arbitration, because he/she does not want to negotiate with B . After stating the grievance, C and B present their allegations to the jury. In this process B presents Agr_1 by which he/she believes there is fulfillment of norm $N0$. Nevertheless, in the last arbitration step, by means of a Normative Reasoning function, the jury analyzes the presented allegations and the normative regulations of the basin and deduces that there is an offense. Norm $N1$ was not complied by B and A , and moreover, A has sold the same water-right twice for an overlapped time period. In this last step, the jury imposes the corresponding sanctions to A and B .

Fig. 4 shows a snapshot of the $mWater$'s complex scenario implementation running on the AMELI execution environment of EIDE. The implementation we have devised for this complex situation in $mWater$ allows us to solve the described scenario. Moreover, when dealing with this scenario it is possible to observe the limitations of the current EIDE platform for supporting non-observability and enforceability of non-regimented norms. The implementation of $mWater$ we are discussing in this paper is developed with EIDE 2.11⁴, and includes all the components described in previous sections. Moreover, the information model that supports the execution of the EI is developed in MySQL and includes the different conceptual data required for the market execution. Fig. 5 shows a fragment of the relational model in which some elements are depicted such as: basin structure, water-right definition, agreement, and conflict resolution table configuration, among others.

$mWater$ is devised as a simulation tool for helping the basin policy makers to evaluate the behaviour of the market when new or modified norms are ap-

⁴ Available at <http://e-institutions.iia.csic.es/eide/pub/>

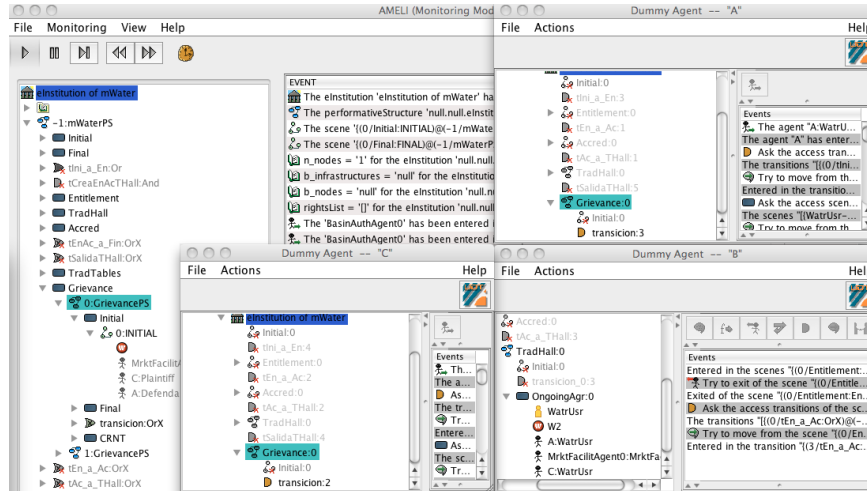


Fig. 4. A snapshot of the *mWater* electronic institution running on AMELI

plied. To this end, we are working on defining evaluation functions to measure the performance of the market. These measures include the amount of water transfer agreements signed in the market, volume of water transferred, number of conflicts generated, etc. Apart from these straightforward functions we are also working on defining "social" functions in order to assess values such as the trust and reputation levels of the market, or degree of water user satisfaction, among others.

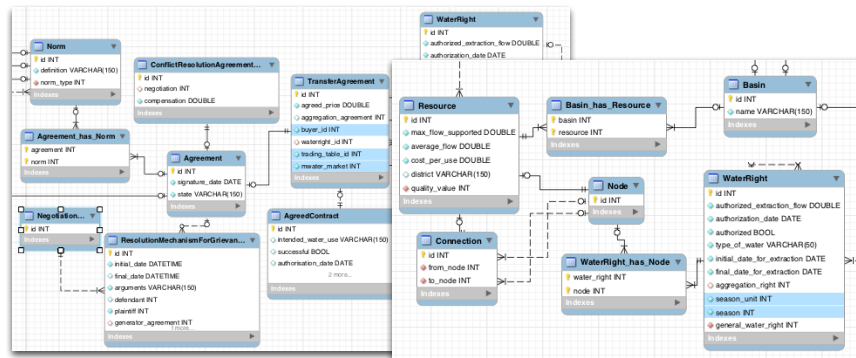


Fig. 5. A fragment of the information model of *mWater*

4 Discussion and closing remarks

In real life problems, in many occasions it is difficult or even impossible to check norm compliance, specially when the violation of the norm cannot be directly observable. In other occasions, it is not only difficult to regiment all actions, but it might be preferable to allow agents to violate norms, since they may obtain a higher personal benefit or they may intend to improve the organization functionality, despite violating or ignoring norms. It is clear that from a general thought and design perspective of an Electronic Institution, it is preferable to define a safe and trustful environment where norms cannot be violated (i.e. norms are considered as hard constraints), thus providing a highly regimented scenario that inspires confidence to their users. However, from a more flexible and realistic perspective, it is appealing to have the possibility for agents to violate norms for personal gain. Although this is a very realistic attribute that humans can have, it eventually leads to corruption and, consequently, the designer may think to rule it out. But again, from a norm enforceability standpoint it is always a good idea to allow this: it does not only make the environment more open and dynamic, but it also provides a useful tool for decision support. In such a thread, we are able to range the set of norms, from a very relaxed scenario to a very tight one, simulate the institution and the agents' behaviour, and finally analyze when the global performance—in terms of number of conflicts that appear, degree of global satisfaction or corruption, etc.—shows better, which makes it very interesting as a testbed itself [5]. In all these cases, norm enforcement methods are needed, such as second-party and third-party enforcements.

This paper has highlighted the necessity for norm enforceability in Electronic Institutions. Clearly, when the agents and their execution occur outside the boundaries of the institution it is inviable to count on a simple and efficient way to guarantee a norm-abiding behaviour, as the full observability of the whole execution and environment is rarely possible. In other words, norm violations are perfectly plausible (and unfortunately common) and are only detectable in presence of a conflict among agents.

In our *mWater* scenario, we have proposed an open mechanism that comprises two main principles: (i) the generation of a grievance when one agent detects a conflict, i.e. when an agent denounces the occurrence of a violation; and (ii) an authority entity with the role of arbiter/judge to mediate in the dispute resolution process and being able to apply sanctions. The advantage of this mechanism is twofold. First, it allows different types of grievance, either when it corresponds to the execution of a previous signed (or unsigned) agreement or, simply, when it happens as an occasional event during the habitual execution of the water scenario and its infrastructure use. Second, it provides different ways to deal with grievances, as shown in Fig. 2: (i) in a very formal and strict way by means of an arbitration procedure that relies on a traditional jury, thus applying a *third-party* enforceability mechanism (with an infrastructure enforcement); or (ii) in a more flexible way that relies on the creation of a conflict resolution negotiation table, which ranges from informal protocols (e.g., face to face) to more formal ones that may need one or more mediators. In this last case, a *second-party*

enforceability mechanism has been adopted. We have shown that this grievance procedure shows to be effective in the *mWater* scenario. But despite its origin in the water environment, it can be easily extrapolated to any other real problem modelled by using EIs, which represent the main contributions of this paper.

The underlying idea to deal with norm enforcement in generic domains follows a simple flow, but it needs some issues to be clearly defined. First of all, we require a procedure to activate or initiate a new grievance. This can be done from any type of performative structure similar to the *TradingHall* of Fig. 1. This operation requires the identification of the agents that will be involved in the grievance itself, so it is essential for all agents to be uniquely identified; that is, we cannot deal with anonymous agents, which is an important issue. Once the grievance has been initiated, we also require a mechanism for recruiting the conflicting parties. Again, this is related to the agents' identification and the necessity of (perhaps formal) communication protocols to summon all the parties. Note that this step is necessary for any type of dispute resolution, both by negotiation tables and arbitration. And, at this point we have a high flexibility for solving the conflicts, as they can be solved in many ways depending on the type of problem we are addressing at each moment. Analogously to the trading tables that we have in the *mWater* scenario, we can use general or particular tables to reach an agreement and, thus, solving the conflict, no matter the real problem we have. Finally, it is also important to note that reaching an agreement when solving the conflict does not prevent from having new conflicts that appear from such an agreement, being necessary the initiation of a new grievance procedure and repeating all the operations. Although such new grievances are possible from both the negotiation table and arbitration alternatives, it is common to have situations where the decisions/verdict taken by the arbitration judges are unappealable.

Our current work of research is focused on providing a more thorough specification of this mechanism to enforce norms in EIs, how the conflict resolution tables can be defined and to come up with specialized protocols for these tables. Our final goal is to be able to integrate this behaviour in a decision support system to simulate different agents' behaviour and norm reasoning to be applied to the *mWater* and other scenarios of execution.

Acknowledgements

This paper was partially funded by the Consolider programme of the Spanish Ministry of Science and Innovation through project AT (CSD2007-0022, INGENIO 2010), MICINN project TIN2008-06701-C03-03 and by the FPU grant AP-2007-01256 awarded to N. Criado. This research has also been partially funded by the Generalitat de Catalunya under the grant 2009-SGR-1434 and Valencian Prometeo project 2008/051.

References

1. Josep Arcos, Marc Esteva, Pablo Noriega, Juan Rodriguez-Aguilar, and Carles Sierra. Engineering open environments with electronic institutions. *Engineering Applications of Artificial Intelligence*, (18):191–204, 2005.
2. Tina Balke. A taxonomy for ensuring institutional compliance in utility computing. In Guido Boella, Pablo Noriega, Gabriella Pigozzi, and Harko Verhagen, editors, *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
3. G. Boella, L. van der Torre, and H. Verhagen. Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 17(1):1–10, 2008.
4. Guido Boella and Leendert van der Torre. Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic*, 6(2):152–171, 2008.
5. V. Botti, A. Garrido, A. Giret, F. Igual, and P. Noriega. On the design of mWater: a case study for Agreement Technologies. In *7th European Workshop on Multi-Agent Systems - EUMAS 2009*, pages 1–15, 2009.
6. Castelfranchi C. Formalising the informal? *Journal of Applied Logic*, N 1, 2004.
7. N. Criado, V. Julian, V. Botti, and E. Argente. A Norm-based Organization Management System. In *AAMAS Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN)*, pages 1–16, 2009.
8. M. Esteva. Electronic Institutions: from specification to development. *IIIA PhD Monography*, 19, 2003.
9. M. Esteva, J.A. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. Arcos. On the formal specification of electronic institutions. *Agent mediated electronic commerce*, pages 126–147, 1991.
10. M. Esteva, B. Rosell, J.A. Rodriguez-Aguilar, and J.L. Arcos. Ameli: An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, page 243. IEEE Computer Society, 2004.
11. N. Fornara and M. Colombetti. Specifying and enforcing norms in artificial institutions (short paper). In *Proc. 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 1481–1484, 2008.
12. D. Gaertner, A. Garcia-Camino, P. Noriega, J.A. Rodriguez-Aguilar, and W. Vasconcelos. Distributed norm management in regulated multiagent systems. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 90. ACM, 2007.
13. Andrés García-Camino, Juan A. Rodríguez-Aguilar, Carles Sierra, and Wamberto Weber Vasconcelos. Norm-oriented programming of electronic institutions. In *Proc. International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 670–672. ACM, 2006.
14. D. Grossi, H. Aldewereld, and F. Dignum. Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems II*, volume 4386, pages 101–114. Springer, 2007.
15. J.F. Hübner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, 20(3):369–400, 2010.
16. N.H. Minsky and V. Ungureanu. A mechanism for establishing policies for electronic commerce. In *International Conference on Distributed Computing Systems*, volume 18, pages 322–331. Citeseer, 1998.

17. N.H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(3):273–305, 2000.
18. Sanjay Modgil, Noura Faci, Felipe Rech Meneguzzi, Nir Oren, Simon Miles, and Michael Luck. A framework for monitoring agent-based normative systems. In Carles Sierra, Cristiano Castelfranchi, Keith S. Decker, and Jaime Simão Sichman, editors, *AAMAS*, pages 153–160. IFAAMAS, 2009.
19. P. Noriega. Agent-mediated auctions: The fishmarket metaphor. *IIIA Phd Monography*, 8, 1997.
20. D.C. North. *Institutions, institutional change, and economic performance*. Cambridge Univ Pr, 1990.
21. A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, 2008.
22. J.A. Rodriguez-Aguilar. On the design and construction of agent-mediated electronic institutions. *IIIA Phd Monography*, 14, 2001.
23. T. Schultz, G. Kaufmann-Kohler, D. Langer, and V. Bonnet. Online dispute resolution: The state of the art and the issues. In *Available at SSRN: <http://ssrn.com/abstract=899079>*.
24. WK Slate. Online dispute resolution: Click here to settle your dispute. *Dispute Resolution Journal*, 56(4):8–14, 2002.

Towards a Normative BDI Architecture for Norm Compliance

N. Criado¹, E. Argente¹, P. Noriega², and V. Botti¹

¹ Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

Camino de Vera s/n. 46022 Valencia (Spain)

Email: {ncriado,eargente,vbotti}@dsic.upv.es

² Institut d'Investigació en Intel·ligència Artificial

Consejo Superior de Investigaciones Científicas

Campus de la UAB, Bellaterra, Catalonia (Spain)

Email: pablo@iia.csic.es

Abstract. Multi-Agent Systems require coordination mechanisms in order to assemble the behaviour of autonomous and heterogeneous agents and achieve the desired performance of the whole system. Norms are deontic statements employed by these coordination mechanisms which define constraints to the potential excesses of agents' autonomous behaviour. However, norms are only effective if agents are capable of understanding and managing them pragmatically. In this paper, we propose an extension of the BDI proposal in order to allow agents to take pragmatic autonomous decisions considering the existence of norms. In particular, coherence and consistency theory will be employed as a criterion for determining norm compliance.

1 Introduction

The development of network technologies and Internet has made it possible to evolve from monolithic and centralized applications, in which problems are solved by a single component, to distributed applications, in which problems are solved by means of the interaction among autonomous agents. In these systems, the autonomy and heterogeneity of agents make mandatory the definition of *coordination* mechanisms for ensuring the whole performance of the system. With this aim, social notions, such as *organizations*, *institutions* and *norms*, have been introduced in the design and implementation of distributed systems.

Norms have been defined in distributed systems as regulations or patterns of behaviour established in order to constrain the potential excesses of autonomous agents. The definition of norms for controlling distributed systems requires the development of normative agents. Normative agents [8] must be endowed with capabilities for considering norms and deciding which norms to comply with and how to comply with them. The multi-context Graded BDI architecture [7] allows agents to reason in uncertain and dynamic environments.

The work presented in [9] is a first effort on the extension of the Graded BDI architecture [7] in order to allow agents to accept norms autonomously. This work focuses on the description of the architecture as a whole but it provides few details about how agents acquire new norms and face with the norm compliance dilemma. In addition, it lacks an elaborated definition of norm and norm dynamics. According to these criticisms, in this paper we propose to revise this architecture in order to allow agents to take norms into account in a more sophisticated way. In particular, here we focus on the application of both *Cognitive Coherence Theory* [26] and *Consistency Theory* [2] for reasoning about norm compliance. Coherence is a cognitive theory whose main purpose is the study of how pieces of information influence each other by imposing a positive or negative constraint over the rest of information. Consistency is a logic property which analyses the relationship among a formula and its negation. Our proposal consists on applying deliberative coherence theory for determining which norms are more coherent with respect to the agent’s mental state. In addition, consistency criterion is considered when determining how to comply with norms. Therefore, this paper tries to overlap some of the main drawbacks of the original proposal by means of adding coherence and consistency constraints to the architecture.

This paper is structured as follows: next section describes the background of our proposal; Section 3 provides norm definitions; in Section 4 the normative BDI architecture is explained; the two components in charge of norm management are explained in Sections 5 and 6; Section 7 describes the norm internalization process; and, Section 8 remarks the contributions and future work.

2 Background

Along this section all approaches considered for this work are explained. In particular, the normative multi-context Graded BDI architecture (n-BDI for short) refined in this paper is explained first. Next subsections introduce the basis of consistency and coherence theories.

2.1 BDI architectures for normative agents

Usually, proposals on agent architectures which support normative reasoning do not consider norms as dynamic objects which may be acquired and recognised by agents. On the contrary, these proposals consider norms as static constraints that are hard-wired on agent architectures. Regarding recent proposals on individual norm reasoning, the BOID architecture [4] represents obligations as mental attributes and analyses the relationship and influence of such obligations on agent beliefs, desires and intentions. However, this proposal presents some drawbacks: i) it only considers obligation norms; ii) it considers norms as static entities that are off-line programmed in agents. In relation with this last feature, the EMIL proposal [1] has developed a framework for autonomous norm recognition. Thus, agents would be able to acquire new norms by observing the behaviour of other agents which are situated in their environments. The main disadvantage of EMIL

is that agents obey all recognised norms blindly without considering their own motivations. The multi-context graded BDI agent architecture [7] does not provide an explicit representation of norms. However, it is capable of representing and reasoning with graded mental attitudes, which makes it suitable as a basis for a norm aware agent architecture.

In order to overlap these drawbacks, in [9, 10], the multi-context graded BDI agent architecture [7] has been extended with recognition and normative reasoning capabilities. According to the n-BDI proposal, an agent is defined by a set of interconnected contexts, where each context has its own logic (i.e. its own language, axioms and inference rules). In addition, bridge rules are inference rules whose premises and conclusions belong to different contexts. In particular, an n-BDI agent [9, 10] is formed by:

- *Mental* contexts [6] to characterize beliefs (BC), intentions (IC) and desires (DC). These contexts contain logic propositions such as $(\Psi\gamma, \delta)$; where Ψ is a modal operator in $\{B, D^+, D^-, I\}$ which express beliefs, positive and negative desires and intentions, respectively; $\gamma \in \mathcal{L}_{\mathcal{DL}}$ is a dynamic logic [19] proposition; and $\delta \in [0, 1]$ represents the certainty degree associated to this mental proposition. For example $(B\gamma, \delta)$ represents a belief about proposition γ of an agent and δ represents the certainty degree associated to this belief.
- *Functional* contexts [6] for planning (PC) and communication (CC).
- *Normative* contexts [9] for allowing agents to recognise new norms (RC) and to consider norms in their decision making processes (NC).
- *Bridge Rules* for connecting mental, functional and normative contexts. A detailed description of these bridge rules can be found in [7]. For a more detailed description of normative bridge rules see [9].

Regarding the normative extension of the BDI architecture, the norm decision process consists of the following steps:

1. It starts when the RC derives a new norm through analysing its environment.
2. These norms are translated into a set of inference rules which are included into the NC. The NC is responsible for deriving new beliefs and desires according to the current agent mental state and the inference rules which have been obtained from norms.
3. After performing the inference process for creating new beliefs and desires derived from norm application, the normative context must update the mental contexts.

The original proposal [9, 10] is a preliminary work towards the definition of autonomous norm aware agents capable of making a decision about norm compliance. In this sense, this approach presents several problems and deficiencies. Firstly, the notion of norm is vague and imprecise, in this sense there is not a clear definition of what an abstract norm and a norm instance mean. Regarding the norm recognition process, no details about how the set of abstract norms is updated and maintained are provided. Thus, the RC is seen as a black box that

gives no analysis of how it deals with different types of norms (e.g. social norms, explicit norms created by the institution). Finally, it lacks a more concrete description of how a BDI agent may decide about obeying or not a norm. In this sense, the derivation of positive and negative desires from obligations and prohibitions is too simple. In particular, norms that guide agent behaviours might be in conflict, since they are aimed at defining the ideal behaviour of different roles which may be played by one agent. Besides that, norm compliance decisions should be consistent with the mental state of agents. Therefore, how agents make consistent decisions about norm compliance is the main contribution of the current paper with respect to the original proposal [9].

As a solution to this problem we will employ works on formalisms for ensuring *consistency* [2] and *coherence* [26]. In particular, this paper describes how these works are applied for reasoning about norm compliance. Next subsections briefly describe both the proposal of Casali et al. [6] on consistency among graded bipolar desires and the work of Joseph et al. [17] on the formalization of deductive coherence for multi-context graded BDI agents.

2.2 Consistency for Graded BDI Agents

In [2] Benferhat et al. made a study of consistency among bipolar graded preferences. Taking this definition of consistency, Casali et al. in [6] proposed several schemas for ensuring consistency among mental graded propositions. In particular, the maintenance of consistency among desires is achieved by means of three different schemas (i.e. DC_1 , DC_2 and DC_3) which impose some constraints between the positive and negative desires of a formula and its negation. Thus, DC_2 schema (which will be employed in this paper) imposes a restriction over positive and negative desires for the same goal ($(D^+ \gamma, \delta_\gamma^+)$ and $(D^- \gamma, \delta_\gamma^-)$, respectively). In particular, it claims that an agent cannot desire to be in world more than it is tolerated (i.e. not rejected). Therefore, it determines that:

$$\delta_\gamma^+ + \delta_\gamma^- \leq 1$$

where δ_γ^+ and δ_γ^- are the desirability and undesirability degrees (i.e. the certainty of the positive and negative desire) of proposition γ , respectively.

2.3 Coherence for Graded BDI Agents

In [26] Thagard claims that coherence is a cognitive theory whose main purpose is the study of associations; i.e. how pieces of information influence each other by imposing a positive or negative constraint over the rest of information. According to Thagard's formalization, a coherence problem is modelled by a graph $g = \langle V, E, \zeta \rangle$; where V is a finite set of nodes representing pieces of information, E are the edges representing the positive or negative constraints among information; each constraint has a weight ($\zeta : E \rightarrow [-1, 1] \setminus \{0\}$) expressing the constraint strength. Maximizing the coherence [26] is the problem of partitioning nodes

into two sets (accepted A and rejected $V \setminus A$) which maximizes the strength of the partition, which is the sum of the weights of the satisfied constraints.

Taking a proof-theoretic approach, Joseph et al. [17] provide a formalization of deductive coherence for multi-context graded BDI agents. Thus, this work proposes a formalization together with mechanisms for calculating the coherence of a set of graded mental attitudes. The main idea beyond this formalism is to consider the inference relationships among propositions belonging to the same context for calculating the weight of coherence and incoherence relationships. Similarly, bridge rules are employed for setting the coherence degree among propositions belonging to different contexts. Details concerning building the coherence graph can be found in [17].

Regarding the relation of coherence with normative decision processes, in [18] Joseph et al. employed coherence as a criterion for rejecting or accepting norms. However, this work is based on a very simple notion of norm as an unconditional obligation. Moreover, this proposal only considers coherence as the one rational criterion for norm acceptance. In addition, the problem of norm conflict has not been faced. Finally the process by which agents' desires are updated according to norms have also been defined in a simple way without considering the effect of these normative desires on the previous existing desires.

3 Norm Notion

Norms have been studied from different fields such as philosophy, psychology, law, etc. MAS research has given different meanings to the norm concept, been employed as a synonym of obligation and authorization [14], social law [20], social commitment [24] and other kinds of rules imposed by societies or authorities.

In this work, we take as a basis the formalization of norms made in [21]. In this proposal a distinction among *abstract norms* and *norm instances* is made. An *abstract norm* is a conditional rule that defines under which conditions obligations, permissions and prohibitions should be created. In particular, the activation condition of an abstract norm defines when an obligation, permission or prohibition must be instantiated. The *norm instances* that are created out of the *abstract norms* are a set of active unconditional expressions that bind a particular agent to an obligation, permission or prohibition. Moreover, a norm instance is accompanied by an expiration condition which defines the validity period or deadline of the norm instance.

Following this proposal our definition of both abstract norms and norm instances is provided.

Definition 1 (Abstract Norm). *An abstract norm is defined as a tuple $n_a = \langle D, A, E, C, S, R \rangle$ where:*

- $D \in \{\mathcal{F}, \mathcal{P}, \mathcal{O}\}$ is the deontic type of the norm. In this work obligations (\mathcal{O}) and prohibitions (\mathcal{F}) impose constraints on agent behaviours; whereas permissions (\mathcal{P}) are operators that define exceptions to the activation of obligations or prohibitions;

- A is the norm activation condition. It defines under which circumstances the abstract norm is active and must be instantiated.
- E is the norm expiration condition, which determines when the norm no longer affects agents.
- C is a logic formula that represents the state of affairs or actions that must be carried out in case of obligations, or that must be avoided in case of prohibition norms.
- S, R are expressions which describe the actions (sanctions S and rewards R) that will be carried out in case of norm violation or fulfilment, respectively.

Since this work is focused on the norm compliance problem, only those norms addressed to the agent will be taken into account.

Definition 2 (Norm Instance). Given belief theory Γ_{BC} an abstract norm $n_a = \langle D, A, E, C, S, R \rangle$ is instantiated into a norm instance $n_i = \langle D, C' \rangle$ where:

- $\Gamma_{BC} \vdash \sigma(A)$, where σ is a substitution of variables in A such that $A' = \sigma(A)$ and $\sigma(S)$, $\sigma(R)$ and $\sigma(E)$ are fully grounded.
- $C' = \sigma(C)$.

Once the activation conditions of an abstract norm hold it becomes active and several norm instances, according to the possible groundings of the activation condition, must be created. For simplicity, we assume that once a norm is being instantiated then it is fully grounded. In our proposal, the instantiation of activation and expiration conditions are considered by the *Norm Instantiation* bridge rule (which will be explained in Section 6). Similarly, sanctions and rewards are also considered by this bridge rule in order to decide about convenience of norm compliance. Thus, for simplicity we omit the instantiation of the norm expiration and activation conditions ($\sigma(A)$ and $\sigma(E)$) and the sanction and reward ($\sigma(S)$ and $\sigma(R)$) in the representation of a norm instance.

4 Normative BDI Architecture (n-BDI)

As previously mentioned, the main contribution of this paper is to refine the n-BDI architecture, which was originally proposed in [9, 10], with a more elaborated notion of norm and norm reasoning. In order to design this second version of the n-BDI, the work of Sripada et al. [25] has been considered as a reference. It analyses the psychological architecture subserving norms. In particular, this architecture is formed by two closely linked innate mechanisms: one responsible for norm acquisition, which is responsible for identifying norm implicating behaviour and inferring the content of that norm; and the other in charge of norm implementation, which maintains a database of norms, detects norm violations and generates motivations to comply with norms and to punish rule violators.

The evolution of n-BDI is focused on reasoning about norm compliance and acceptance, so issues related to the detection and reaction to norm violation are beyond the scope of this paper. In this sense, norms affect n-BDI agents in

two ways: i) when a norm is recognised and accepted then it is considered to define new plans; and ii) when accepted norms are active then their instances are used for selecting the most suitable plan which complies with norms. This paper tackles with this last effect of norms. In particular, this paper describes how *Deductive Coherence* (described in Section 2.3) and *Consistency Theory* (described in Section 2.2) are applied for reasoning about norm compliance.

The n-BDI refines the normative contexts (described in Section 2.1) according to the norm notions introduced in Section 3. Therefore, Figure 1 shows a scheme of the n-BDI proposed in this paper. In particular, the RC has been redefined as the *Norm Acquisition Context* (NAC), whereas the NC has been redefined into the *Norm Compliance Context* (NCC).

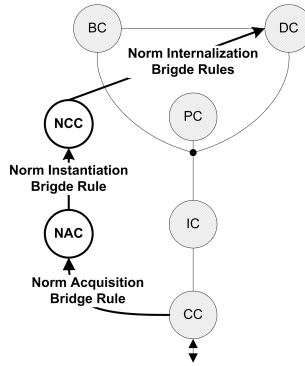


Fig. 1. Normative Extension of the Multi-Context BDI Architecture. Grey contexts and dashed lines (bridge rules) correspond to the basic definition of a BDI agent. The normative extensions are the white contexts and bold lines.

In this new version of the agent architecture not only the normative contexts have been improved by considering more elaborated normative definitions, but also the norm reasoning process has been extended with consistency and coherence notions. Thus, the norm reasoning process can be described as follows:

1. It starts when the NAC receives information cues for inferring new abstract norms through the *Norm Acquisition* bridge rule. The NAC carries out an inference process for maintaining the set of abstract norms in force in a specific moment.
2. Once the norm activation conditions hold, abstract norms are instantiated and included into the NCC by means of the *Norm Instantiation* bridge rule. Then, the NCC carries out an internal process for determining compliance with which of the norm instances. In this sense, not all active norms should be considered when updating the mental state. In this sense, our proposal consists in employing coherence theory as a criterion for determining *which* norms comply with. Therefore, the coherence maximization process is cal-

culated in order to determine which norm instances are consistent and must be taken into account when updating the desire theory.

3. Then, *Norm Internalization* bridge rules derive new desires according to the current agent mental state and the set of complied norms, also taking into account consistency considerations. These new desires may help the agent to select the most suitable plan to be intended and, as a consequence, normative actions might be carried out by the agent.

Thus, the norm reasoning process is formed by four different phases: acquisition of norms in force, decision about norm compliance and internalization of norms. Next sections describe each one of these phases in detail.

5 Norm Acquisition (NAC)

The NAC context allows agents to maintain a norm base that contains those norms which are *in force* in a specific moment (i.e. all norms which are currently applicable). Thus it is responsible for acquiring new norms and deleting obsolete norms; and updating the set of in force norms accordingly. This process can be defined as objective since no motivation or goal is considered in the acquisition process. Thus, agents only take into account their knowledge of the world in order to determine the set of norms which is more likely to be in force.

NAC Language. The NAC is formed by expressions such as (n, ρ) where n is an abstract norm according to Definition 1; and $\rho \in [0, 1]$ is a real value which assigns a degree to this abstract norm. This parameter ρ can have different interpretations. It can be defined as the reputation of the informer agent in case of leadership-based norm spreading. If norms are inferred by imitation, ρ might represent the acceptance degree of the norm. In case of utility maximizing approaches, as learning algorithms, it can be defined as the expected utility of the norm.

Abstract Norm Recognition. Regarding how new and obsolete norms are recognised, the NAC consists of a computational model of autonomous norm recognition which receives the agent perceptions, both observed and communicated facts, and identifies the set of norms which control the agent environment. Perceptions which are relevant to the norm recognition may be classified into:

- *Explicit normative perceptions.* They correspond to those messages exchanged by agents in which norms are explicitly communicated. Following this approach, several works have focused on analysing the role of leaders in the norm spreading. In particular, these leaders provide normative advices to follower agents when deciding about a norm [27, 22].
- *Implicit normative perceptions.* This type of perceptions includes the observation of actions performed by agents as a way of detecting norms. Since norms are usually supported by enforcing mechanisms such as sanctions and

rewards, the detection of them has been considered as an alternative for acquiring new norms [15]. Other works have proposed imitation mechanisms as a criterion for acquiring new norms. These models are characterized by agents mimicking the behaviour of what the majority of the agents do in a given agent society [28, 5]. Moreover, in [23] researchers have experimented with learning algorithms to identify a norm that maximizes an agent's utility.

- *Mixed normative perceptions*. There are proposals which consider both explicit and implicit normative perceptions as cues for inferring norms [1].

Abstract Norm Dynamics. The set of norms which are in force may change both explicitly, by means of the addition, deletion or modification of the existing norms; and implicitly by introducing new norms which are not specifically meant to modify previous norms, but which change in fact the system because they are incompatible with such existing norms and prevail over them [16]. However, this is a complex issue which is out of the scope of this paper. Works presented at the *Formal Models of Norm Change*³ are good examples of proposals which provide a formal analysis of all kinds of dynamic aspects involved in systems of norms.

This paper does not focus on the norm acquisition problem and the dynamics of abstract norms. In the following, the NAC will be considered as a *black box* that receives cues for detecting norms as input and generates abstract norms as output.

6 Norm Compliance (NCC)

The NCC is the component responsible for reasoning about the set of norms which hold in a specific moment. In this sense the NAC recognises all norms that are in force, whereas the NCC only contains those norms which are active according to the current situation. The NCC should determine which and how norms will be obeyed and support agents when facing with norm violations. In this sense, the NCC detects norm violations and fulfilments and generates punishing and rewarding reactions. This last issue is over the scope of this paper and will be analysed in future works.

The functionalities carried out by the NCC which are covered by this work are related to three main issues: the NCC is in charge of maintaining the set of instantiated norms which are active; then it considers convenience of norm compliance and determines which norms comply with; and, finally, it derives new desires for fulfilling these norms.

NCC Language. The NCC is formed by expressions such as: (n, ρ) where n is a norm instance according to Definition 2. $\rho \in [0, 1]$ is a real value which assigns a degree to this norm instance. This parameter can be interpreted as the salience of the norm instance. Its value can be determined according to different

³ <http://www.cs.uu.nl/events/normchange2/>

criteria such as utility of norm compliance, emotional considerations, intrinsic motivations, etc. In this paper, it is defined with regard to the certainty of norm activation as well as the convenience of norm compliance.

Instantiated norms are inferred by applying instantiation bridge rules to norms when their activation is detected. Next, these normative bridge rules are described in detail.

Norm Instantiation Bridge Rule.

$$\frac{NAC : (\langle D, A, E, C, S, R \rangle, \rho), BC : (B A, \beta_A), BC : (B \neg E, \beta_E)}{NCC : (\langle D, C \rangle, f_{instantiation}(\theta_{activation}, \theta_{compliance}))} \quad (1)$$

If an agent considers that an abstract norm $n_a = \langle D, A, E, C, S, R \rangle$ is currently active $((B A, \beta_A) \wedge (B \neg E, \beta_E))$ then a new norm instance $n_i = \langle D, C \rangle$ is generated. The degree assigned to the norm instance is defined by the $f_{instantiation}$ function which combines the values obtained by the $\theta_{activation}$ and $\theta_{compliance}$ functions.

On the one hand, $\theta_{activation}$ combines the evidence about norm activation (i.e. the certainty degrees β_A, β_E and ρ). It can be given a sophisticated definition depending on the concrete application. In this work, it has been defined as the weighed average among these three values, as follows:

$$\theta_{activation} = \frac{w_A \times \beta_A + w_E \times \beta_E + w_\rho \times \rho}{w_A + w_E + w_\rho}$$

If all values are equally weighed, then we obtain that $\theta_{activation} = \frac{\beta_A + \beta_E + \rho}{3}$

On the other hand, $\theta_{compliance}$ considers both intrinsic and instrumental motivations for norm compliance. In [11] different strategies for norm compliance from an instrumental perspective over this architecture are described. In particular, they consider the influence of norm compliance and violation on agent's goals for determining whether the agent accepts the norm. For example, an *egoist* agent will accept only those norms which benefit its goals (i.e. whose condition is positively desired). In this case:

$$\theta_{compliance} = \begin{cases} 1 & \text{if } \delta_C^+ > 0, \text{ where } (D^+ C, \delta_C^+) \in \Gamma_{DC}; \\ 0 & \text{otherwise} \end{cases}$$

Finally, values obtained by the $\theta_{activation}$ and the $\theta_{compliance}$ functions are combined by the $f_{instantiation}$:

$$f_{instantiation}(\theta_{activation}, \theta_{compliance}) = \frac{w_{activation} \times \theta_{activation} + w_{compliance} \times \theta_{compliance}}{w_{activation} + w_{compliance}}$$

Again, if these two parameters are equally weighed, then we obtain that

$$f_{instantiation}(\theta_{activation}, \theta_{compliance}) = \frac{\theta_{activation} + \theta_{compliance}}{2}$$

This approach relies upon various values such as $w_{compliance}$, w_A and $w_{activation}$. The definition of these values is beyond the scope of this paper. In previous works [9, 11, 10], it has been considered that they are defined off-line by the agent designer. However, this solution is static and it does not allow agents to adapt these values according to a changing environment. Thus, this issue will be considered in future works.

6.1 Coherence For Norm Instances

Once *Norm Instantiation* bridge rule has been applied, it is possible that there is an incoherence between mental propositions. Because of this, a maximizing coherency process is needed in order to determine which propositions are consistent and must be taken into account; and which propositions belonging to the rejection set will be ignored when deriving normative desires.

Since our proposal of agent architecture employs graded logics for representing mental propositions, this work takes as a basis the work described in Section 2.3. As argued before, this work proposes a formalization together with mechanisms for calculating the deductive coherence of a set of graded mental attitudes. Our proposal adapts this work by applying the coherence maximization algorithm to the norm compliance problem. Figure 2 illustrates an overview of the employment of coherence as a criterion for resolving the norm compliance dilemma. As shown by this figure, the normative coherence process considers propositions belonging to the BC, the NCC and NAC. Basically this process takes into account: i) the beliefs that sustain the activation of norms and their relationships among them and other beliefs of the BC; ii) the norm instances and conflict relationships among them; and iii) the abstract norms which have triggered the norm activation. Relationships among propositions belonging to each context are defined by means of inference rules and axioms, whereas coherence connections among propositions of different contexts are defined by means of norm instantiation bridge rules.

By considering coherence we will address three different problems: i) determining norm deactivation; ii) determining active norms in incoherent states and iii) normative conflict resolution. In order to formalize normative incoherence the original proposal of [17] must be extended with extra constraints. Moreover, since we apply the coherence calculation algorithms for improving the normative reasoning, then only those propositions which are relevant to the norm compliance process are taken into account. Next, both the definition of normative coherence constraints for facing with each one of these three questions as well as the determination of relevant propositions is detailed.

Detecting Norm Activation in Incomplete and Inconsistent States. As illustrated in Figure 2, norm instantiation bridge rule (see equation 6) allows norm instances (from NCC) to be connected to beliefs from BC related to their activation and expiration conditions. Norm instantiation bridge rule has as pre-conditions the belief about the occurrence of the activation condition A and the

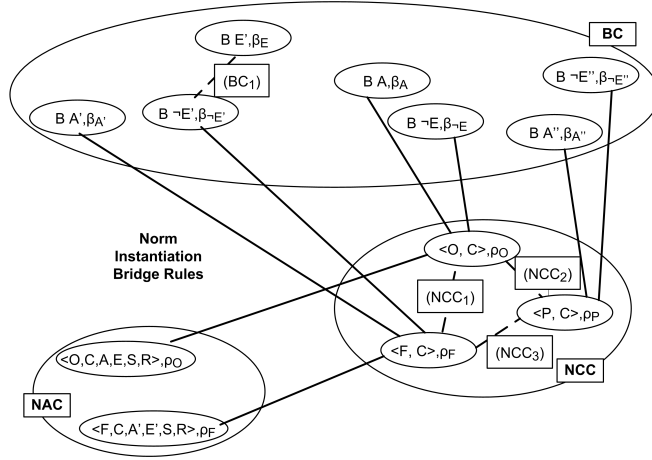


Fig. 2. Usage of coherence as a criterion for resolving the norm compliance dilemma.

negation of the expiration condition E . Usually agents do not have an explicit knowledge about the negation of E . However, it is possible to infer a certainty degree about $\neg E$ from the certainty degree of E . Following this idea, the first step for computing coherence is to calculate the closure under negation of beliefs as follows:

Definition 3 (Closure of Beliefs under Negation). *Let Γ be a finite belief theory presentation using graded formulas. We define the closure of Γ under negation as:*

$$\Gamma^\neg = \Gamma \cup \{(B\neg\varphi, (1 - \delta)) : (B\neg\varphi, \beta) \notin \Gamma \text{ and } (B\varphi, \delta) \in \Gamma\}$$

Therefore, the closure of a set of beliefs under negation consists on extending this theory by inferring new information from what is actually believed. In particular, if an agent believes that proposition E is true with a certainty degree δ but it does not have any belief concerning its negation, it is logic to assume that the certainty degree assigned to $\neg E$ should be lower than $(1 - \delta)$. We need to calculate the closure of beliefs under negation in order to detect norm deactivation. In this sense, when the certainty about the expiration condition E increases it can be inferred that the certainty of $\neg E$ decreases even if the agent does not have explicit evidence of it.

In addition we want to define an incoherence relationship among a belief related to a general proposition and its negation. This relationship is defined by means of the addition of an inference rule in the belief context:

$$(BC_1) (B\gamma, \beta_\gamma), (B\neg\gamma, \beta_{\neg\gamma}) \vdash (\bar{0}, 1 - (\beta_\gamma + \beta_{\neg\gamma}))$$

Basically this scheme means that to belief γ and $\neg\gamma$ simultaneously is a contradiction ($\bar{0}$). The certainty degree of this contradiction is defined in [18] as $1 - (\beta_\gamma + \beta_{\neg\gamma})$.

One of the main problems of the multi-context BDI architecture is the fact that it does not allow the definition of bridge rules for deleting propositions. In this sense, there is a bridge rule for inferring a new instance of a norm when its activation condition holds. However, it is not possible to create a bridge rule which deletes this instance when the expiration condition holds. In response to this problem, coherence will be used as a criterion for detecting norm deactivation. Moreover, an agent may have beliefs related to the occurrence of both the norm activation and expiration conditions. Thus it should consider all those evidences that sustain the occurrence of the expiration and activation conditions in order to determine the set of norms which are active. In particular, coherence will be used as a criterion for detecting norm activation/deactivation according to the certainty of both the expiration and the activation conditions.

Resolving Normative Conflicts. As previously argued, the above process of normative coherence is useful not only to determine which norms are active but even to resolve a norm conflict. Usually, a norm conflict has been defined in other works as a situation in which something is considered as forbidden and obliged or forbidden and permitted. In our proposal, we define permissions as a normative operator which allows defining an exception to the application of a more general obligation or prohibition norm. Thus, we also consider that norms which define something as forbidden and permitted are also in conflict. However, there is no constraint that represents this type of incoherence. In order to represent incoherence inferred from norm conflicts we add the next inference rules to the NCC:

$$\begin{aligned} (NCC_1) \quad & (\langle O, C \rangle, \rho_O), (\langle F, C \rangle, \rho_F) \vdash (\bar{0}, -\min(\rho_O, \rho_F)) \\ (NCC_2) \quad & (\langle O, C \rangle, \rho_O), (\langle P, \neg C \rangle, \rho_P) \vdash (\bar{0}, 1 - (\rho_O + \rho_P)) \\ (NCC_3) \quad & (\langle F, C \rangle, \rho_F), (\langle P, C \rangle, \rho_P) \vdash (\bar{0}, 1 - (\rho_F + \rho_P)) \end{aligned}$$

In case of a conflict between a permission and an obligation or a prohibition, the degree of the falsity constant ($\bar{0}$) is assigned value $1 - (\rho_O + \rho_P)$ or $1 - (\rho_F + \rho_P)$, respectively, in a similar way as in BC_1 . In case of a conflict among a prohibition and an obligation we define a stronger incoherence by defining the degree of the falsity constant as $-\min(\rho_O, \rho_F)$.

Selecting Relevant Propositions. Once the coherence graph has been defined and a maximising partition $(A, V \setminus A)$ over this graph has been found following [17], the set of propositions belonging to the NCC (i.e. Γ_{NCC}) is revised in order to consider only the accepted norms:

$$\Gamma'_{NCC} = \Gamma_{NCC} \cap A$$

where A is the accepted set of norm instances according to the maximizing coherence process [17], i.e. "the most coherent norm instances".

7 Norm Internalization

Regarding works on norm *internalization* in the MAS community, maybe the most relevant proposal are the works of Conte et al. [8]. According to them, a characteristic feature of norm internalization is that norms become part of the agent's identity. The concept of identity implies that norms become part of the cognitions of the individual agent. In particular, Conte et al. define norm internalization as a multi-step process, leading from externally enforced norms to norm-corresponding goals, intentions and actions with no more external enforcement. Thus they account for different types and levels of internalization.

In this paper a simplistic approximation to the norm internalization process has been considered. However, it will be object of future work extensions. In particular, we have only considered the internalization of norms as goals. In this sense, the process of norm internalization has been described by the self-determination theory [13] as a dynamic relation between norms and desires. This shift would represent the assumption that internalised norms become part of the agent's sense of identity. Thus, after performing the coherence process for creating new norm instances, the NCC must update the DC (Figure 1 Norm Internalization Bridge Rules) with the new normative desires. The addition of these propositions into this mental context may cause an inconsistency with the current mental state. As explained in Section 2.2, in [6] three schemas for ensuring consistency among mental graded propositions have been proposed. According to schema DC_2 , which imposes a restriction over positive and negative desires for a same goal, we have implemented the following inference rule:

$$(DC_2) (D^+ \gamma, \delta_\gamma^+), (D^- \gamma, \delta_\gamma^-) \vdash (\bar{0}, 1 - (\delta_\gamma^+ + \delta_\gamma^-))$$

Our proposal needs bipolar representation of desires since it is useful when selecting plans to be intended for achieving the desires. In this sense, both negative and positive effects of actions will be taken into account when selecting a plan to be intended. For example, the fact that a plan involves a forbidden action may be considered as a negative effect. Therefore, obligation norms are internalized as positive desires whereas prohibition norms are translated into negative ones. Because of this, DC_2 has been considered as a basis for the definition of bridge rules responsible for updating the DC in a consistent way. Next these norm internalization bridge rules are described.

Norm Internalization Bridge Rules.

- *Obligation Norm.* According to DC_2 schema, bridge rule for updating the DC with the positive desires derived from obligation norms is defined as follows:

$$\frac{NCC : (\langle O, C \rangle, \rho), DC : (D^- C, \delta^-), DC : (D^+ C, \delta^+)}{DC : (D^+ C, \max(\rho, \delta^+)), DC : (D^- C, \min(\delta^-, 1 - \max(\rho, \delta^+)))} \quad (2)$$

If an agent considers that the obligation is currently active then a new positive desire will be inferred corresponding to the new norm condition. Thus, the desire degree assigned to the new proposition C is defined as the maximum between the new desirability and the previous value ($\max(\rho, \delta^+)$). Moreover, the undesirability assigned to C is updated as the minimum between the previous value of undesirability assigned to γ (δ^-) and its maximum coherent undesirability, which is defined as $1 - \max(\rho, \delta^+)$.

- *Prohibition Norm.* Bridge rule for updating the DC with negative desires is defined as follows:

$$\frac{NCC : (\langle F, C \rangle, \rho), DC : (D^- C, \delta^-), DC : (D^+ C, \delta^+)}{DC : (D^- C, \max(\rho, \delta^-)), DC : (D^+ C, \min(\delta^+, 1 - \max(\rho, \delta^+))} \quad (3)$$

Similarly to obligation norms, a prohibition related to a condition C is transformed into a negative desire related to the norm condition $(D^- C, \max(\rho, \delta^-))$.

- *Permission Norm.* Finally, permission norms do not infer a positive or negative desire about the norm condition. Permission norms define exceptions to the application of a more general obligation or prohibition norm. As a consequence, they only are defined for creating an incoherence with these more general norms.

8 Conclusion

In this work a previous proposal [9, 10] of a normative BDI architecture has been revised. The first contribution of the current paper is the usage of coherence theory in order to determine what means to follow or violate a norm according to the agent's cognitions and making a decision about norm compliance. The second contribution of this paper is the employment of consistency notions for updating agent cognitions in response to these normative decisions.

The impact of normative decisions on agent cognitions will be object of future work. In this paper, the norm internalization problem has been faced in a simplistic way by considering only the impact of obeyed norms on agent's desires. Therefore, in future works the role of both *deliberative coherence* [26] and *emotions* on the norm compliance will be analysed. In particular, *deliberative coherence*, which deals with goal adoption in the context of decision making, will be considered when building plans for obeying or violating norms. In addition, we will work on extending our agent architecture with an explicit representation of emotions which will allow agents to consider phenomena such as shame, honour, gratitude, etc. in their decision making processes.

Due to lack of space, no evaluation or case study has been included here that might provide a more understanding perspective of our proposal. However, works describing the original proposal [10, 11] (neither consistency nor coherence are considered here) provide examples belonging to the m-Water case study. The m-Water [3] is a water right market which is implemented as a regulated open

multi-agent system. It is a challenging problem, specially in countries like Spain, since scarcity of water is a matter of public interest. The m-Water framework [12] is a somewhat idealized version of current water-use regulations that articulate the interactions of those individual and collective entities that are involved in the use of water in a closed basin. This is a regulated environment which includes the expression and use of regulations of different sorts: from actual laws and regulations issued by governments, to policies and local regulations issued by basin managers, and to social norms that prevail in a given community of users. For these reasons, we consider the m-Water problem as a suitable case study for evaluating performance of the n-BDI agent architecture, since agents' behaviour is affected by different sorts of norms which are controlled by different mechanisms such as regimentation, enforcement and grievance and arbitration processes.

Finally, we are working on the implementation of a prototype of the n-BDI architecture. Our aim is to evaluate empirically our proposal through the design and implementation of scenarios belonging to the m-Water case study. In future works, we will make some experiments concerning the flexibility and performance of the n-BDI agent model with respect to simple BDI agents, using the m-Water case study. However, preliminary results of the experimental evaluation of the n-BDI original proposal can be found in [10].

9 Acknowledgments

This work was partially supported by the Spanish government under grants CONSOLIDER-INGENIO 2010 CSD2007-00022, TIN2009-13839-C03-01 and TIN2008-04446 and by the FPU grant AP-2007-01256 awarded to N. Criado.

References

1. G. Andrighetto, M. Campenní, F. Cecconi, and R. Conte. How agents find out norms: A simulation based model of norm innovation. In *NORMAS*, pages 16–30, 2008.
2. S. Benferhat, D. Dubois, S. Kaci, and H. Prade. Bipolar representation and fusion of preferences on the possibilistic logic framework. In *KR*, pages 421–434. Morgan Kaufmann Publishers; 1998, 2002.
3. V. Botti, A. Garrido, A. Giret, and P. Noriega. Managing water demand as a regulated open mas. In *MALLOW Workshop on COIN*, page In Press., 2009.
4. J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. van der Torre. The boid architecture – conflicts between beliefs, obligations, intentions and desires. In *AAMAS*, pages 9–16. ACM Press, 2001.
5. M. Campenní, G. Andrighetto, F. Cecconi, and R. Conte. Normal= Normative? The role of intelligent agents in norm innovation. *Mind & Society*, 8(2):153–172, 2009.
6. A. Casali, L. Godo, and C. Sierra. A logical framework to represent and reason about graded preferences and intentions. In *KR*, pages 27–37. AAAI Press, 2008.

7. A. Casali, L. Godo, and C. Sierra. *On Intentional and Social Agents with Graded Attitudes*. PhD thesis, Universitat de Girona, 2008.
8. R. Conte, G. Andrighetto, and M. Campenni. On norm internalization. a position paper. In *EUMAS*, 2009.
9. N. Criado, E. Argente, and V. Botti. A BDI Architecture for Normative Decision Making (Extended Abstract). In *9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1383–1384, 2010.
10. N. Criado, E. Argente, and V. Botti. Normative deliberation in graded bdi agents. In *MATES*, page In Press, 2010.
11. N. Criado, E. Argente, and V. Botti. Rational strategies for autonomous norm adoption. In *9th International Workshop on Coordination, Organization, Institutions and Norms in Multi-Agent Systems (COIN@AAMAS2010)*, pages 9–16, 2010.
12. N. Criado, E. Argente, A. Garrido, J. A. Gimeno, F. Igual, V. Botti, P. Noriega, and A. Giret. Norm enforceability in electronic institutions? In *11th International Workshop on Coordination, Organization, Institutions and Norms in Multi-Agent Systems (COIN@MALLOW2010)*, page In Press, 2010.
13. E. Deci and R. Ryan. The” what” and” why” of goal pursuits: Human needs and the self-determination of behavior. *Psychological Inquiry*, 11(4):227–268, 2000.
14. F. Dignum. Autonomous agents with norms. *Artif. Intell. Law*, 7(1):69–79, 1999.
15. F. Flentge, D. Polani, and T. Uthmann. Modelling the emergence of possession norms using memes. *Journal of Artificial Societies and Social Simulation*, 4(4), 2001.
16. G. Governatori and A. Rotolo. Changing legal systems: Abrogation and annulment part i: Revision of defeasible theories. In *DEON*, pages 3–18, 2008.
17. S. Joseph, C. Sierra, M. Schorlemmer, and P. Dellunde. Formalising deductive coherence: An application to norm evaluation. Technical report, IIIA-CSIC, 2009.
18. S. Joseph, C. Sierra, M. Schorlemmer, and P. Dellunde. Deductive coherence and norm adoption. *Logic Journal of the IGPL*, page In Press, 2010.
19. J. Meyer. Dynamic logic for reasoning about actions and agents. In *Logic-Based Artificial Intelligence*, pages 281–311. Kluwer Academic Publishers, 2000.
20. Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 14(6), 1995.
21. N. Oren, S. Panagiotidi, J. Vázquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. In *COIN IV*, pages 156–171, Berlin, Heidelberg, 2009. Springer-Verlag.
22. B. T. R. Savarimuthu, M. Purvis, and M. K. Purvis. Social norm emergence in virtual agent societies. In *AAMAS*, pages 1521–1524. IFAAMAS, 2008.
23. S. Sen and S. Airiau. Emergence of norms through social learning. In *IJCAI*, pages 1507–1512, 2007.
24. M. P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
25. C. Stripada and S. Stich. A framework for the psychology of norms. *The Innate Mind: Culture and Cognition*, pages 280–301, 2006.
26. P. Thagard. *Coherence in Thought and Action*. The MIT Press, Cambridge, Massachusetts, 2000.
27. H. Verhagen. *Norm Autonomous Agents*. PhD thesis, Stockholm University, 2000.
28. F. y Lopez. *Social Power and Norms*. PhD thesis, Citeseer, 2003.

Generating Executable Multi-Agent System Prototypes from SONAR Specifications

Michael Köhler-Bußmeier, Matthias Wester-Ebbinghaus, Daniel Moldt

University of Hamburg, Department for Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
(koehler,wester,moldt)@informatik.uni-hamburg.de

Abstract. This contribution presents the MULAN4SONAR middleware and its prototypical implementation for a comprehensive support of organisational teamwork, including aspects like team formation, negotiation, team planning, coordination, and transformation. Organisations are modelled in SONAR, a Petri net-based specification formalism for multi-agent organisations. SONAR models are rich and elaborated enough to automatically generate all necessary configuration information for the MULAN4SONAR middleware.

1 Introduction

Organisation-oriented software engineering is a discipline which incorporates research trends from distributed artificial intelligence, agent-oriented software engineering, and business information systems (cf. [1, 2] for an overview). The basic metaphors are built around the interplay of the macro level (i.e. the organisation or institution) and the micro level (i.e. the agent). Organisation-oriented software models are particularly interesting for self- and re-organising systems since the system's organizing principles (structural as well as behavioral) are taken into account explicitly by representing (in terms of reifying) them at run-time.

The following work is based on the organisation model SONAR (Self-Organising Net Architecture) which we have presented in [3, 4]. In this paper we turn to a middleware concept and its prototypical implementation for the complete organisational teamwork that is induced by SONAR.

First of all we aim at a rapid development of our middleware prototype. Therefore we need a specification language that inherently supports powerful high-level features like pattern matching and synchronisation patterns. The second requirement is a narrow gap between the specification and implementation of the middleware prototype. Ideally, middleware specifications are directly executable. As a third requirement, we are interested in well established analysis techniques to study the prototype's behaviour. As a fourth requirement we want the middleware specifications to be as close as possible to the supported SONAR-model of an organization. Related to this, the fifth requirement results as the possibility to be able to directly generate the middleware specifications from the SONAR-model automatically. The sixth requirement is that we want an easy translation of the prototype into an agent programming language.

Since SONAR-models are based on Petri nets we have chosen high-level Petri nets [5] as the specification language for our middleware prototype. This choice meets the requirements stated above: We can reuse SONAR-models by enriching them with high-level features, like data types, arc inscription functions etc. Petri nets are well known for their precise and intuitive semantics and their well established analysis techniques, including model checking or linear algebraic techniques. We particularly choose the formalism of *reference nets*, a dialect of high-level nets which supports the nets-in-nets concept [6] and thus allows to immediately incorporate (“program”) micro-macro dynamics into our middleware. Reference nets receive tool support with respect to editing and simulation by the RENEW tool [7]. Additionally, RENEW has been extended by the agent-oriented development framework MULAN [8, 9], which allows to program multi-agent systems in a language that is a hybridisation of reference nets and Java. We make use of MULAN and provide a middleware for SONAR-models. Consequently, our middleware is called MULAN4SONAR and we present a fully-functional prototype in this paper.

The paper is structured as follows: Section 2 briefly sketches our formal specification language for organisational models, called SONAR. Section 3 addresses our MULAN4SONAR middleware approach on a rather abstract and conceptual level. It illustrates the structure of our target system: SONAR-models are compiled into a multi-agent system consisting of so called position agents, i.e. agents that are responsible for the organisational constraints. Section 4 describes our implemented middleware prototype in detail. It is generated from SONAR-models. The middleware serves integration and control of all organisational activities, like team formation, negotiation, team planning, coordination, and transformation. We consider related work in Section 5 before we close the paper with a conclusion in Section 6.

2 The Underlying Theoretical Model: SONAR

In the following we give a short introduction into our modelling formalism, called SONAR. A SONAR-model encompasses (i) a data ontology, (ii) a set of interaction models (called *distributed workflow nets, DWFs*), (iii) a model, that describes the team-based delegation of tasks (called *role/delegation nets*), (iv) a network of organisational positions, and (v) a set of transformation rules. A detailed discussion of the formalism can be found in [3], its theoretical properties are studied in [4].

In SONAR a formal organisation is characterised as a delegation network of sub-systems, called *positions*. Each position is responsible for the execution or delegation of several tasks. Figure 1 illustrates the relationship between the SONAR interaction model, the delegation model and the position network – i.e. the aspects (ii) to (iv).¹ The left side of the figure describes the relationship between the positions (here: *broker, virtual firm, requester, etc.*) in terms of their respective roles (here: *Producer, Consumer etc.*) and their associated delegation links. In

¹ To keep the model small we we have omitted all data-related aspects and transformation rules – i.e. the aspects (i) and (v) – in this figure.

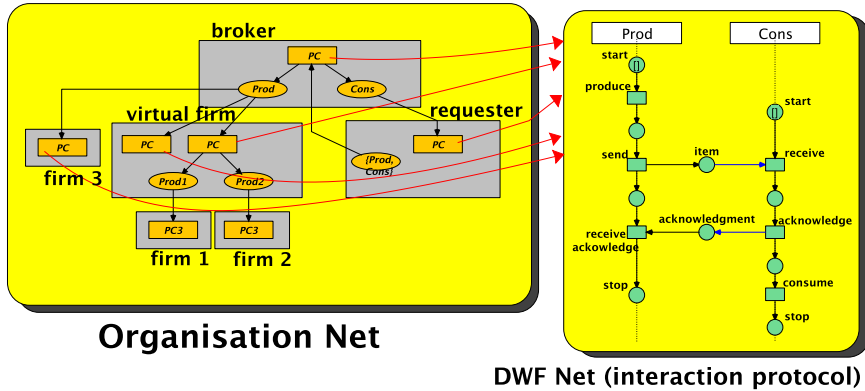


Fig. 1. A simplified SONAR-Model

this scenario, we have a requester and two suppliers of some product. Coupling between them is provided by a broker.² From a more fine-grained perspective, the requester and one of the suppliers consist of delegation networks themselves. For example, in the case of the *virtual firm* supplier, we can identify a management level and two subcontractors: *firm 1* *firm 2*. The two subcontractors may be legally independent firms that integrate their core competencies in order to form a virtual enterprise (e.g. separating fabrication of product parts from their assembly). The coupling between the firms constituting the virtual enterprise is apt to be tighter and more persistent than between requester and supplier at the next higher system level, which provides more of a market-based and on-the-spot connection.

SONAR relies on the formalism of Petri nets. Each task is modelled by a place p and each task implementation (delegation/execution) is modelled by a transition t . Each task place is inscribed by the set of roles which are needed to implement it, e.g. the set $\{Prod, Cons\}$ for the place in the position *requester*. Each transition t is inscribed by the DWF net $D(t)$ that specifies the interaction between the roles. In the example we have two inscriptions: PC and $PC3$ where the former is shown on the right of Figure 1. Positions are the entities which are responsible for the implementation of tasks.³ Therefore, each node in $P \cup T$ is assigned to one position O .⁴

² Note that for this simplified model brokerage is an easy job, since there are only two producers and one consumer. In general, we have several instances for both groups with a broad variety of quality parameters, making brokerage a real problem.

³ The main distinction between roles and position is that positions – unlike roles – are situated in the organisational network, they implement roles and are equipped with resources.

⁴ Organisation nets can be considered as enriched organisation charts. Organisation nets encode the information about delegation structures – similar to charts – and also about the delegation/execution choices of tasks, which is not present in charts.

So far we have used only the static aspects of Petri nets, i.e. the graph structure. But SONAR also benefits from the dynamic aspects of Petri nets: Team formation can be expressed in a very elegant way. If one marks one initial place of an organisation net *Org* with a token, each firing process of the Petri net models a possible delegation process. More precisely, the *token game* is identical to the team formation process (cf. Theorem 4.2 in [4]). It generates a *team net* (the team's structure) and a *team DWF*, i.e. the team's behavior specification.

As another aspect, SONAR-models are equipped with transformation rules. Transformation rules describe which modifications of the given model are allowed. They are specified as graph rewrite rules [10]. As a minimal requirement the rules must preserve the correctness of the given organisational model. In SONAR transformations are not performed by the modeller – they are part of the model itself. Therefore we assume that a SONAR model is *stratified* by models of different levels. The main idea is that the activities of DWF nets that belong to the level n are allowed to modify those parts that belong to levels $k < n$ but not to higher ones.

3 Organisational Position Network Activities

We now elaborate on the activities of a multi-agent system behaving according to a SONAR-model.

3.1 Conceptual Overview

The basic idea is quite simple: With each position of a SONAR-model we associate one dedicated agent, called an *organisational position agent* (OPA). This is illustrated in Figure 2 where the OPAs associated with a SONAR-model together embody a middleware layer.

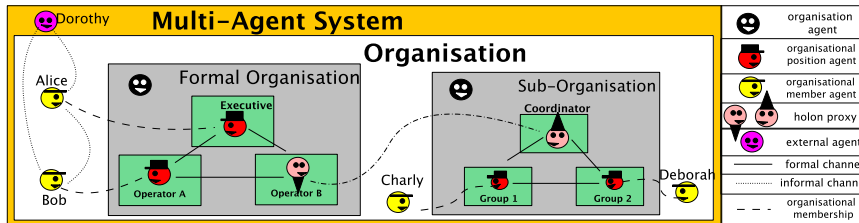


Fig. 2. An Organisation as an OPA/OMA Network

An OPA network embodies a formal organisation. An OPA represents an organisational *artifact* and not a *member/employee* of the organisation. However,

If one fuses all nodes of each position into one single node, one obtains a graph which represents the organisation's chart. Obviously, this construction removes all information about the organisational processes.

each OPA represents a conceptual connection point for an *organisational member agent* (OMA). An organisation is not complete without its OMAs. Each OMA actually interacts with its OPA to carry out organisational tasks, to make decisions where required. OMAs thus implement/occupy the formal positions.⁵ Note that an OMA can be an artificial as well as a human agent. An OPA both enables and constrains organisational behaviour of its associated OMA. Only via an OPA an OMA can effect the organisation and only in a way that is in conformance with the OPA’s specification. In addition, the OPA network as a whole relieves its associated OMAs of a considerable amount of organisational overhead by automating coordination and administration. To put it differently, an OPA offers its OMA a “behaviour corridor” for organisational membership. OMAs might of course only be partially involved in an organisation and have relationships to multiple other agents than their OPA (like Alice and Bob in Figure 2) or even to agents completely external to the organisation (like Alice and Dorothy). From the perspective of the organisation, all other ties than the OPA-OMA link are considered as informal connections.

To conclude, an OPA embodies two conceptual interfaces, the first one between micro and macro level (one OPA versus the complete network of OPAs) and the second one between formal and informal aspects of an organisation (OPA versus OMA). We can make additional use of this twofold interface. Whenever we have a system of systems setting with multiple scopes or domains of authority (e.g. virtual organisations, strategic alliances, organisational fields), we can let an OPA of a given (sub-)organisation act as a member towards another OPA of another organisation. This basically combines the middleware perspective with a holonic perspective (cf. [11]).

3.2 Organisational Teamwork

SONAR-models of organisations induce *teamwork activities*. We distinguish between organisational teamwork activities of first- and of second-order. First-order activities target at carrying out “ordinary” business processes to accomplish business tasks.

- *Team Formation*: Teams are formed in the course of an iterated delegation procedure in a top-down manner. Starting with an initial organisational task to be carried out, successive task decompositions are carried out and sub-tasks are delegated further. A team net according to Section 2 consists of the positions that were involved in the delegation procedure.
- *Team Plan Formation/Negotiation*: After a team has been formed, a compromise has to be found concerning how the corresponding team DWF net (cf. Section 2) is to be executed as it typically leaves various alternatives of going

⁵ Note that from a technical point of view, the OPA network *is* already a complete MAS. This MAS is highly non-deterministic since a SONAR-model specifies what is allowed and what is obligatory, so many choices are left open. Conceptually, the OPA network represents the *formal organisation* while the OMAs represent its *informal* part which in combination describe the whole organisation.

one way or the other. A compromise is found in a bottom-up manner with respect to the team structure. The “leaf” positions of the team net tell their preferences and the intermediary, inner team positions iteratively seek compromises between the preferences/compromise results of subordinates. The final compromise is a particular process of the team DWF net and is called the team plan.

- *Team Plan Execution*: As the team plan is a DWF net process that describes an interaction between team positions, team plan execution follows straightforward.⁶
- *Hierarchical propagation*: If a holonic approach as illustrated in Figure 2 is chosen, team activities that span multiple organisations are propagated accordingly.

Second-order activities reorganisation efforts.

- *Evaluation*: Organisational performance is monitored and evaluated in order to estimate prospects of transformations. To estimate whether an organisational transformation would improve organisational performance, we introduce *metrics* that assign a multi-dimensional assessment to a formal organisation. In addition to the Petri net-based specifications of the previous section, there may exist additional teamwork constraints and parameters that may be referred to. How to measure the quality of an organisational structure is generally a very difficult topic and highly contingent. We will not pursue it further in this paper.
- *Organisational Transformations*: As described in Section 2, transformations can either be applied to a formal organisation externally or be carried out by the positions themselves as transformation teams (cf. exogenous versus endogenous reorganisation [12]). In the latter case, transformations are typically triggered by the above mentioned evaluations. But it might also be the case that a new constraint or directive has been imposed and the organisation has to comply.

3.3 Organisation Agents

As shown in Figure 2 all the OPAs of an organisation are within the context of an *organisation agent* which represents the OPA network as a whole. The organisation agent is responsible for the management of the organisational domain data (e.g. customer databases etc.) but also for the management of the organisational *meta data* which includes the data ontology, the interaction protocols (i.e. the process ontology), and also a representation of the SONAR-model itself. This is illustrated in the top half of Figure 3.

Additionally, the organisation agent is responsible for the network wide framing of the organizational teamwork efforts, i.e. team formation, negotiation, and

⁶ For the time being, we do not address the topic that team plan execution might fail and what rescue efforts this might entail.

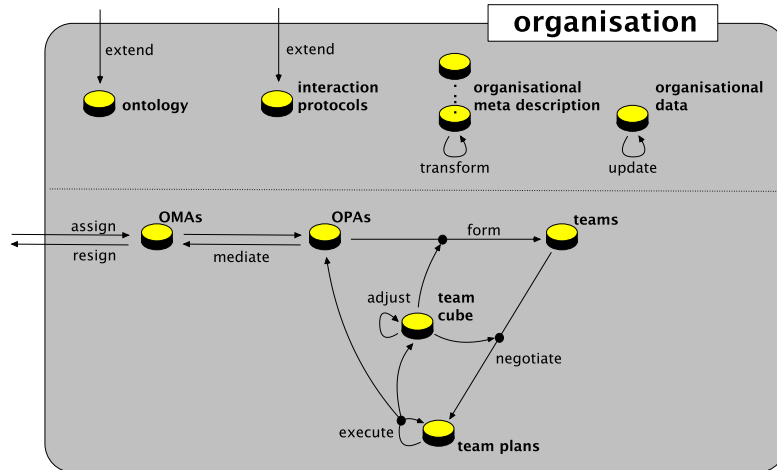


Fig. 3. The Organisation Agent

team plan execution (as illustrated in the bottom half of Figure 3). The organisation agent is responsible for monitoring the *abstract* aspects on the teamwork (i.e. the OPA network perspective), while the OPAs are responsible for the *concrete* decisions (i.e. the OPA perspective).⁷ For example, the organisation agent abstractly specifies that during the team formation the OPA O may delegate some task to another agent which must belong to a certain set of OPAs,⁸ but the concrete choice for a partner is left to the OPA O which in turn coordinates its decision with its associated OMA.

In our architecture the concrete choices of the OPAs are framed by the so called *team cube* (cf. Figure 3). The notation *cube* is due to the fact that we have three dimension of teamwork: team formation, negotiation, and team plan execution. For each dimension we can choose between several mechanisms. For example in the team formation phase the delegation of tasks to subcontractors can either be implemented by a market mechanism (i.e. choosing the cheapest contractor), by a round-robin scheduling (i.e. choosing contractors in cyclic order), or even by some kind of “affection” between OPAs/OMAs. Given a concrete situation that initiates a teamwork activities, the organisation chooses an appropriate mechanism for

⁷ Note that the existence of a single agent representing the organisation has not to be confused with a monolithic architecture. The main benefit of the existence of an organisation agent is that it allows to provide a network-wide *view* on the team activities.

The abstract aspects could as well be implemented by the OPAs themselves and thus be totally distributed. In fact the concurrency semantics of Petri nets perfectly reflects this aspect: In the mathematical sense the processes of an organisation agent are in fact distributed, even if generated from one single net.

⁸ This set of possible delegation partners is calculated from the SONAR-model.

each of the three dimensions. During the execution phase of the team plans the team cube evaluates the process to improve the assignment of mechanisms.

4 The MULAN4SONAR Middleware

Each position of a SONAR-organisation consists of a formal part (the OPA as an organisational artifact) and an informal part (the OMA as a domain member). An organisation together with the OPA network relieves its associated OMAs of a great part of the organisational overhead by automation of administrative and coordination activities. It is exactly the *generic* part of the teamwork activities from Section 3.2 that is automated by the organisation/OPA network: Team formation, team plan formation, team plan execution always follow the same mechanics and OMAs only have to enter the equation where domain actions have to be carried out or domain-dependent decisions have to be made.

4.1 Compilation of SONAR Specifications into MULAN4SONAR

In the following we demonstrate the compilation of an organisational SONAR-model into the MULAN4SONAR middleware layer for automated teamwork support. A SONAR-model is semantically rich enough to provide all necessary information to allow an automated generation/compilation. The aspects of this compilation and the resulting prototypical middleware are discussed using the organisation example introduced above in Figure 1. The prototypical middleware layer generated from this SONAR-model is specified by a high-level Petri net, namely a reference net. This is beneficial for two reasons: (1) the translation result is very close to the original specification, since the prototype directly incorporates the main Petri net structure of the SONAR-model; (2) the prototype is immediately functional as reference nets are directly executable using the open-source Petri net simulator RENEW [7] and we can easily integrate the prototype into MULAN [8, 9], our developing and simulation system for MAS based on Java and reference nets. Therefore we have chosen to implement the compiler as a RENEW-plugin.

The plugin implements a compiler that is based on graph rewriting. The compiler searches for a net fragment in the SONAR-model that matches the pattern on the left hand side of a rewrite rule and translates it into a reference net fragment which is obtained as the instantiation of the rule's right hand side. An example rule with the parameter n is given in Figure 4: The rule attaches a place for the OPA a to the transition. In the final model this place contains the OPA that represents the position "position name". The rule also adds inscriptions that describe that OPA a is willing to implement the task t (denoted by the inscription `a:askImpl("t")`) and a list of inscriptions `a:askPartner("pi", Oi)` (one for each $p_i, 1 \leq i \leq n$) describing that a delegates the subtask p_i to the OPA O_i . The variable x denotes the identifier of the teamwork process.

We consider teamwork in six phases. For each phase, the original SONAR-model (in our case the one from Figure 1) is taken and transformation rules generate

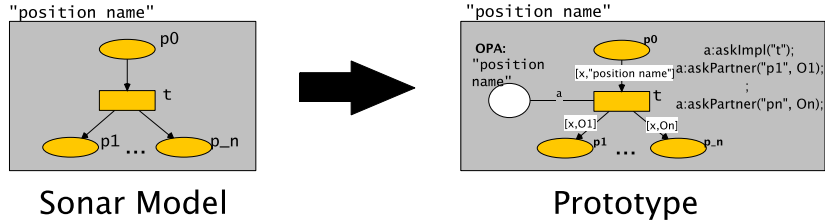


Fig. 4. A transformation rule for Phase 1

an executable reference net fragment. For example, the transformation rule from Figure 4 is used for the first phase, *selection of team members* (see below). Finally, the fragments for the phases 1 to 6 are linked sequentially and the resulting overall net represents the main (organisation-specific) middleware component that is used in the (generic) MULAN4SONAR middleware layer to coordinate the organizational teamwork. The six teamwork phases are the following:

1. *Selection of team members*: By agents receiving tasks, refining them and delegating sub-tasks, the organisation is explored to select the team agents. This way, a team tree is iteratively constructed but the overall tree is not globally known at the end of this phase.
2. *Team assembly*: The overall team tree is assembled by iteratively putting sub-teams together. At the end of this phase, only the root agent of the team tree knows the overall team.
3. *Team announcement*: The overall team is announced among all team member agents.
4. *Team plan formation*: The executing team agents (i.e. the leaves of the team tree) construct partial local plans related to the team DWF net. These partial plans are iteratively processed by the ancestors in the team tree. They seek compromises concerning the (possibly conflicting) partial plans until the root of the team tree has build a global plan with a global compromise.
5. *Team plan announcement and plan localisation*: The global team plan is announced among all team member agents. The executing team agents have to localise the global plan according to their respective share of the plan.
6. *Team plan execution*: The team generates an instance of the team DWF net, assigns all the local plans to it, and starts the execution.

Here, we will only discuss first-order organisational teamwork. However, our MULAN4SONAR middleware approach features a recursive system architecture in order to support reorganization, including second-order activities (a presentation of the whole model can be found in the technical report [13]).

Before the six phases are discussed in more detail, we illustrate how a MULAN multi-agent system that incorporates our MULAN4SONAR middleware layer looks like.

4.2 Multi-Agent System with MULAN4SONAR Middleware Layer

In Section 3, we have described our general vision of a multi-agent system that incorporates SONAR organisations: The formal part of each SONAR organization is explicitly represented by a distributed middleware layer consisting of OPAs for each position and one organisation agent as an additional meta-level entity. In our current prototypical implementation of the MULAN4SONAR middleware layer, the organisation agent is actually not yet fully included, at least not as an *agent*. Instead, the organisation agent of a SONAR organisation manifests itself in terms of the generated six-phase reference net explained in the previous subsection (together with possible DWF nets). This concept is illustrated in Figure 5.

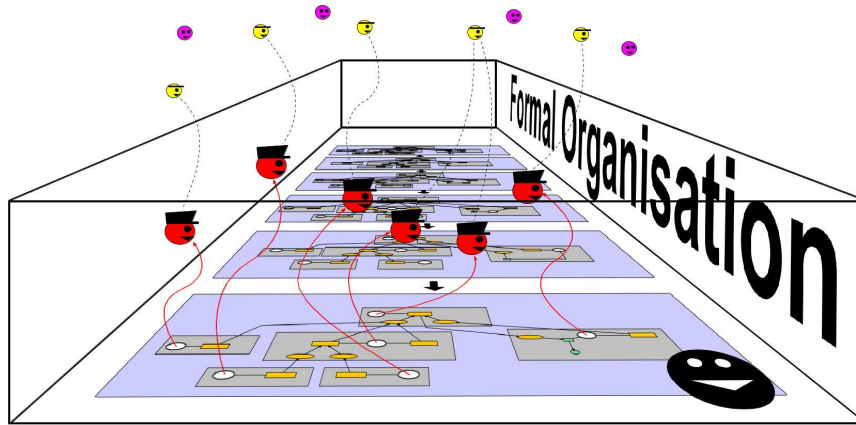


Fig. 5. MULAN4SONAR middleware layer in the current prototype

It is shown that the formal part of a SONAR organisation is embodied by the generated middleware net and the position agents that are hosted on the *agent places* of the net. Here we do not elaborate on the internal structure of the agents as we would have to go into the details of multi-agent system programming with MULAN which is out of the scope of the paper. All OPAs share the same generic OPA architecture (GOPA) that we have presented in [14]. Note that in the current prototype, the OPAs are *directly embedded* on the agent places of the middleware net. This is justified as they are actually reified *parts* of the formal organisation and we assume that the whole middleware (and thus the formal organization) is executed on the same MULAN platform. The OMAs however are external agents that have chosen to act as members of the organization. Consequently, they can be hosted on remote platforms and communicate with their respective OPAs via message passing.

For future developments of our MULAN4SONAR middleware we plan to have the organisation agent to be actually realized as a MULAN agent (see Subsection 4.4).

4.3 Explanation of the Six Teamwork Phases

As explained in Subsection 4.1, a SONAR-model of an organisation is compiled into executable reference nets for each of the six teamwork phases. Afterwards, the reference nets for the phases 1 to 6 are combined in one reference net and linked sequentially. This linkage is achieved via synchronisation inscriptions. Thus, the end of a phase is synchronised with the start of the succeeding phase.

The reference nets for the six phases share the same net structure but have different inscriptions. This reflects the fact that all teamwork is generated from the same organisational SONAR-specification, but in different phases different information is needed. Figure 6 shows the generated reference net for the first phase, *selection of team members*.⁹

Before any teamwork can occur, the system setup has to be carried out. Six position agents (OPAs) – one for each position – are initialized and registered. The position agents are hosted on the `agent` places of the generated middleware net. After this step the initialisation is finished and teamwork may ensue.

For our given SONAR-model we have only one position that is able to start a team, namely O_4 since it is the only position having a place with an empty preset (i.e. the place p_0). Whenever the position agent O_4 decides to begin teamwork, it starts the first phase, *team member selection*. The only possibility for task p_0 is to delegate it to O_1 . Here, O_1 has only one implementation possibility for this task, namely t_1 . This entails to generate the two subtasks p_1 and p_2 . O_1 selects the agents these subtasks are delegated to. For p_2 there is the only possibility O_4 but for p_1 there is the choice between O_2 and O_3 . Partner choices occur via the synchronisation `a:askPartner(p, O)` between the middleware net and the position agents: Agent a provides a binding for the partner O when the task p has to be delegated. Assume that the agent O_1 decides in favour of O_3 , then the control is handed over to O_3 which has a choice how to implement the task: either by t_2 or by t_5 . This decision is transferred between position agents and middleware net via the synchronisation `a:askImpl(t)` which is activated by the agent a only if t has to be used for delegation/implementation.

After this iterated delegation has come to an end – which is guaranteed for well-formed SONAR-models – all subtasks have been assigned to team agents and the first phase ends. At this point the agents know that they are team members, but they do not know each other yet. To establish such mutual knowledge the second phase starts.

We cannot cover every phase in detail. The general principle has been shown for phase one, namely enriching the original SONAR model of an organisation with (1) connections to position agents and (2) execution inscriptions along the purpose of the respective teamwork phase.

⁹ Note that the rule from Figure 4 has been applied several times.

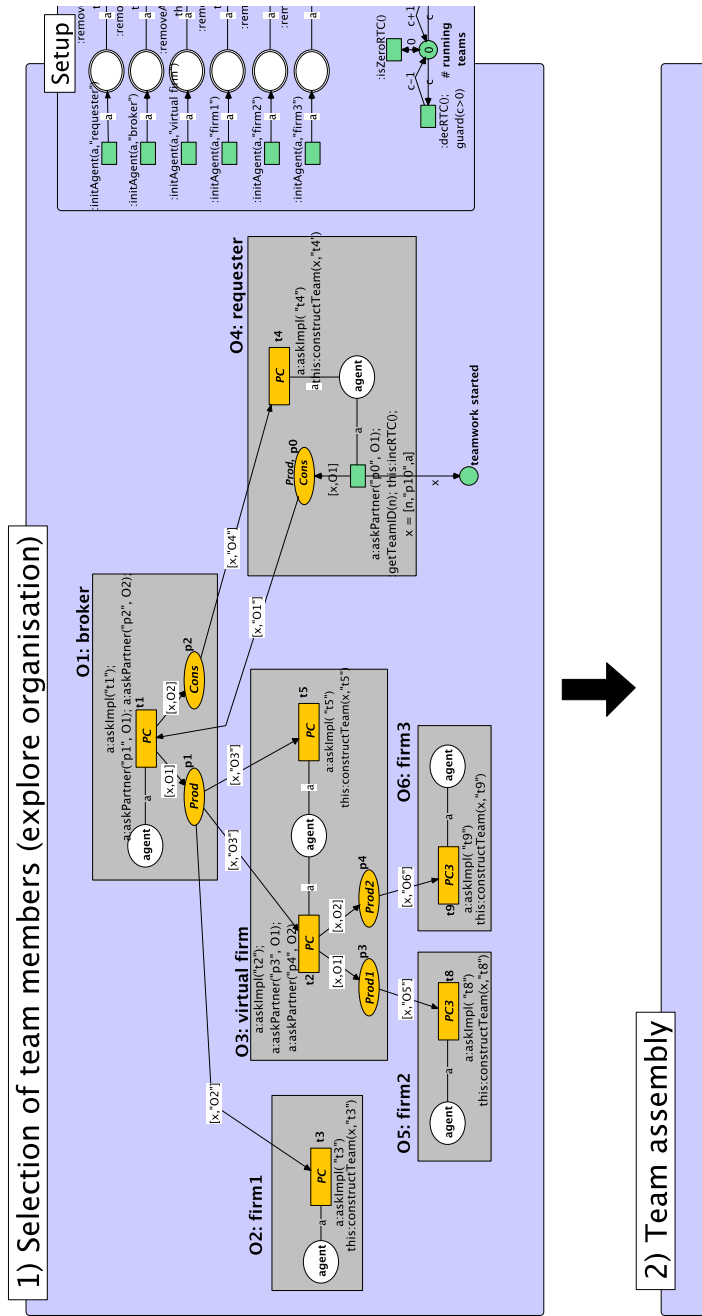


Fig. 6. Zoom: First Phase of the MULAN4SONAR-middleware

The purpose of the remaining five phases has been covered in Subsection 4.1. Here, we want to cover one technical aspect specifically. The description of the first phase has made clear that it is a top-down phase. Following the delegation relationships of the original SONAR-model, a team tree is built from the root down to the leaves. It is also clear that the second phase has to be a bottom-up phase. The overall team is not yet known to any position agent. Thus, beginning with the leaves of the team tree and the corresponding "one-man sub-teams", sub-teams are iteratively assembled until the complete team is finally known at the root node. Consequently, for the second phase, the direction of the arrows has to be reversed compared to the original SONAR model. Analogous observations hold for the remaining four phases. Phases 3 and 5 are top-down phases while phases 4 and 6 are bottom-up phases.

4.4 Strengths, Weaknesses and Future Work

In this subsection, we give a brief qualitative evaluation of the approach taken in this paper. SONAR is a formal model of organisations based on Petri nets. It is often difficult to initially come up with an approach to deploy formal specifications in a software environment. In the case of the Petri net specifications, one can take advantage of the inherent operational semantics. In this sense, Petri nets often allow for a rapid prototyping approach to go from abstract models (requiring only simple Petri net formalisms) to fully functional, executable models (requiring high-level Petri net formalism, in our case reference nets). Consequently, our first approach was to take a SONAR-specification of an organisation and derive an executable prototype by manually attaching inscriptions and add some auxiliary net elements.

While manually crafting an executable reference net for each specific SONAR-model is of course not worthwhile in the long run, it provided us with very early lessons learned and running systems from the beginning on. The work presented in this paper was the next step. Based on our experiences from the handcrafted prototypes we were able to clearly denominate and devise the transformation rules that were needed for automated generation of executable reference net fragments from SONAR-models.

Consequently, we see the conceptual as well as operational closeness between an underlying SONAR-model and its generated middleware net as a crucial advantage for our fast progress in deploying SONAR-organisations. In addition, formal properties like well-formedness (cf. [3, 4]) of a SONAR-model directly carry over to the implementation level.

However, there are also problems associated with our current approach. Firstly, organisational specifications at run-time are only available in terms of the reference net generated from the underlying SONAR-model. This format is not very suitable for being included in an agent's reasoning processes. Secondly, reorganization efforts are only achieved via a workaround. Changing only particular elements of a reference net at runtime is not inherently supported by our environment. Thus, for a reorganization of an organization, the whole middleware net has to be replaced.

Because of the mentioned problems, we are working on further improving the MULAN4SONAR middleware. Current efforts target at keeping the organizational specification as a more accessible and mutable data structure at the level of the middleware layer. Although it is no longer necessarily represented as a reference net itself, the organisational activities and dynamics allowed by the middleware layer are still directly derived from the underlying Petri net semantics of SONAR.

5 Related Work

Our work is closely related to other approaches that propagate middleware layers for organisation support in multi-agent systems like \mathcal{S} -MOISE⁺ [15], AMELI [16] or TEAMCORE/KARMA [17]. The specifics of each middleware layer depends on the specifics of the organizational model that is supported. What all approaches have in common is that domain agents are granted access to the middleware layer via *proxies* that constrain, guide and support an agent in its function as a member of the organisation, cf. *OrgBox* in \mathcal{S} -MOISE⁺, *Governor* in AMELI, *Team Wrapper* in TEAMCORE/KARMA. Our organisational position agents, the OPAs, serve a similar purpose. They are coupled with organisational member agents, the OMAs, which are responsible for domain-level actions and decisions.

However, in the case of \mathcal{S} -MOISE⁺ and AMELI, management of organisational dynamics is mainly taken care of by middleware manager agents (the *OrgManager* for \mathcal{S} -MOISE⁺ and the *institution*, *scene* and *transition managers* for AMELI). The proxies mainly route communication between the domain level agents and the middleware managers. Consequently, middleware management is to some degree centralised.¹⁰ In our case, the OPAs are both proxies and middleware managers. They manage all six phases of organisational teamwork in a completely distributed way. This is quite similar to the function of the Team Wrappers in TEAMCORE/KARMA. The KARMA middleware component can be compared to the organisational agent in our approach. It is a meta-level entity that is responsible for setting up the whole system and for monitoring performance.

In [19], we additionally study the conceptual fit between different middleware approaches (in combination with the organisational models they support) and their application on different levels of a large-scale system of systems.

6 Conclusion

In this paper, we have built upon our previous work SONAR on formalising organisational models for MAS by means of Petri nets [4, 3]. In particular, the paper is dedicated to a prototypical MULAN4SONAR middleware layer that supports the deployment of SONAR-models. As SONAR-specifications are formalised with

¹⁰ However, in the case of \mathcal{S} -MOISE⁺, the new middleware approach ORA4MAS [18] (*organizational artifacts for MAS*) has been devised, resulting in a more decentralised approach.

Petri nets, they inherently have an operational semantics and thus already lend themselves towards immediate implementation. We have taken advantage of this possibility and have chosen the reference net formalism as an implementation means. Reference nets implement the nets-in-nets concept [6] and thus allow us to deploy SONAR-organisations as nested Petri net systems. The reference net tool RENEW [7] offers comprehensive support, allowing us to refine/extend the SONAR specifications into fully executable prototypes.

This leaves us with a close link between a SONAR specification of an organisation and its accompanying MULAN4SONAR middleware support. The structure and behaviour of the resulting software system is directly derived and compiled from the underlying formal model. For example, we have explicitly shown how the organisation net of a formal SONAR-specification can be utilised for the middleware support of six different phases of teamwork. In each phase, the original net is used differently (with different inscriptions and arrow directions). This approach of deploying SONAR-models does not only relieve the developer of much otherwise tedious programming. It also allows to preserve desirable properties that can be proven for the formal model and that now carry over to the software technical implementation.

Finally, although we have introduced the idea of SONAR-organizations acting in the context of other SONAR-organizations, we have not addressed the topic in detail here. We study this subject in [20, 21], but on a more abstract/generic level than SONAR offers. Nevertheless, we have already begun to transfer the results to SONAR.

References

1. Carley, K.M., Gasser, L.: Computational organisation theory. In Weiß, G., ed.: *Multiagent Systems*. MIT Press (1999) 229–330
2. Dignum, V., ed.: *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global (2009)
3. Köhler-Bußmeier, M., Wester-Ebbinghaus, M., Moldt, D.: A formal model for organisational structures behind process-aware information systems. *Transactions on Petri Nets and Other Models of Concurrency. Special Issue on Concurrency in Process-Aware Information Systems* **5460** (2009) 98–114
4. Köhler, M.: A formal model of multi-agent organisations. *Fundamenta Informaticae* **79** (2007) 415 – 430
5. Girault, C., Valk, R., eds.: *Petri Nets for System Engineering – A Guide to Modeling, Verification, and Applications*. Springer (2003)
6. Valk, R.: Object Petri nets: Using the nets-within-nets paradigm. In Desel, J., Reisig, W., Rozenberg, G., eds.: *Advanced Course on Petri Nets 2003*. Volume 3098 of LNCS, Springer (2003) 819–848
7. Kummer, O., Wienberg, F., Duvaligneau, M., Schumacher, J., Köhler, M., Moldt, D., Rölke, H., Valk, R.: An extensible editor and simulation engine for Petri nets: Renew. In Cortadella, J., Reisig, W., eds.: *International Conference on Application and Theory of Petri Nets 2004*. Volume 3099 of LCNS, Springer (2004) 484 – 493
8. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In Colom, J.M., Koutny, M., eds.: *International Conference on Application and Theory of Petri Nets*. Volume 2075 of LNCS, Springer (2001) 224–241

9. Cabac, L., Döriges, T., Duvigneau, M., Moldt, D., Reese, C., Wester-Ebbinghaus, M.: Agent models for concurrent software systems. In Bergmann, R., Lindemann, G., eds.: Proceedings of the Sixth German Conference on Multiagent System Technologies, MATES'08. Volume 5244 of LNAI, Springer (2008) 37–48
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of algebraic graph transformation. Springer (2006)
11. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: A foundation for the organization of multiagent systems. In: Holonic and Multi-Agent Systems for Manufacturing, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS). Volume 2744 of LNCS, Springer (2003) 71–80
12. Boissier, O., Hübner, J., Sichman, J.S.: Organization oriented programming: From closed to open organizations. In O'Hare, G., Ricci, A., O'Grady, M., Dikenelli, O., eds.: Engineering Societies in the Agents World VII. Volume 4457 of LNCS, Springer (2007) 86–105
13. Köhler-Bußmeier, M., Wester-Ebbinghaus, M.: A Petri net based prototype for MAS organisation middleware. In Moldt, D., ed.: Workshop on Modelling, object, components, and agents (MOCA'09), University of Hamburg, Department for Computer Science (2009) 29–44
14. Köhler-Bußmeier, M., Wester-Ebbinghaus, M.: Sonar: A multi-agent infrastructure for active application architectures and inter-organisational information systems. In Braubach, L., van der Hoek, W., Petta, P., Pokahr, A., eds.: Conference on Multi-Agent System Technologies, MATES 2009. Volume 5774 of LNAI, Springer (2009) 248–257
15. Hübner, J.F., Sichman, J.S., Boissier, O.: S-moise: A middleware for developing organised multi-agent systems. In: International Workshop on Organizations in Multi-Agent Systems: From Organizations to Organization-Oriented Programming (OOP 2005). (2005) 107–120
16. Esteva, M., Rodriguez-Aguilar, J., Rosell, B., Arcos, J.: Ameli: An agent-based middleware for electronic institutions. In Sierra, C., Sonenberg, L., Tambe, M., eds.: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004). (2004) 236–243
17. Pynadath, D., Tambe, M.: An Automated Teamwork Infrastructure for Heterogeneous Software Agents and Humans. In: Autonomous Agents and Multi-Agent Systems, 7(1–2). (2003) 71–100
18. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents: Giving the organisational power back to the agents. In: Autonomous Agents and Multi-Agent Systems, 20(3). (2010) 369–400
19. Wester-Ebbinghaus, M., Köhler-Bußmeier, M., Moldt, D.: From Multi-Agent to Multi-Organization Systems: Utilizing Middleware Approaches. In: Alexander, A., Picard, G., Vercouter, L., eds.: Engineering Societies in the Agents World IX. Volume 5485 of LNCS, Springer (2008) 46–65
20. Wester-Ebbinghaus, M., Moldt, D.: Modelling an open and controlled system unit as a modular component of systems of systems. In Köhler-Bußmeier, M., Moldt, D., Boissier, O., eds.: International Workshop on Organizational Modelling (OrgMod'09), University of Paris (2009) 81–100
21. Wester-Ebbinghaus, M., Moldt, D.: Structure in threes: Modelling organization-oriented software architectures built upon multi-agent systems. In: Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2008). (2008) 1307–1311

Embodied Organizations: a unifying perspective in programming Agents, Organizations and Environments

Michele Piunti¹, Olivier Boissier², Jomi F. Hübner³, and Alessandro Ricci¹

¹ Università di Bologna, Italy - {michele.piunti,a.ricci}@unibo.it

² Ecole des Mines St-Etienne, France - boissier@emse.fr

³ University of Santa Catarina, Florianópolis, Brazil - jomi@inf.furb.br

Abstract. MAS research pushes the notion of openness related to systems combining heterogeneous computational entities. Typically, those entities answer to different purposes and functions and their integration is a crucial issue. Starting from a comprehensive approach in developing agents, organizations and environments, this paper devises an integrated approach and describes a unifying programming model. It introduces the notion of *embodied organization*, which is described first focusing on the main entities as separate concerns; and, second, establishing different interaction styles aimed to seamlessly integrate the various entities in a coherent system. An integration framework, built on top of Jason, CArAgO and Moise (as programming platforms for agents, environments and organizations resp.) is described as a suitable technology to build embodied organizations in practice.

1 Introduction

Agent based approaches consider *agents* as autonomous entities encapsulating their control, characterized (and specified) by epistemic states (beliefs) and motivational states (goals) which result in a goal oriented behavior. Recently, organization oriented computing in Multi Agent Systems (MAS) has been advocated as a suitable computation model coping with the complex requirements of socio-technical applications. As indicated by many authors [8, 2, 6], *organizations* are a powerful tool to build complex systems where computational agents can autonomously pursue their activities exhibiting social attitudes. The organizational dimension is conceived in terms of functionalities to be exploited by agents, while it is assumed to control social activities by monitoring and changing those functionalities at runtime. Being conceived in terms of human organizations, i.e., being structured in terms of norms, roles and global objectives, this perspective assumes an organizational layer aimed at promoting desired coordination, improving control and equilibrium of social dynamics. Besides, the need for openness and interoperability requires to cope with computational *environments* populated by several entities, not modellable as agents or organizations, which are supposed to be concurrently exploited by providing functionalities

supporting agents objectives. These aspects are even more recognized in current ICT, characterized by a massive interplay of self-interested entities (humans therein) developed according to different models, technologies and programming styles. Not surprisingly, recent approaches introduced environment as pivotal dimension in MAS development [22, 14]. Such a multifaceted perspective risks to turn systems into a scattered aggregation of heterogenous elements, while their interplay, as well as their interaction, is reduced to a problem of technological interoperability. To prevent this, besides the different mechanisms and abstractions that must be considered, there is a strong need of binding these elements together in a flexible and clear way.

Providing a seamless integration of the above aspects places the challenge to conceive the proper integration pattern between several entities and constructs. A main concern is agent awareness, namely the need for agents to exhibit special abilities and knowledge in order to bring about organizational and environmental notions—which typically are not native constructs of their architectures [21, 15]. Once the environment dimension is introduced as an additional dimension, a second concern is how to connect in a meaningful way the organizational entities and the environmental ones, thereby (i) how the organization can ground normative measures as regimentation and obligations in environments, and (ii) how certain events occurring in environments may affect the global organizational configuration. These aspects enlighten a series of drawbacks on existing approaches, either on the conceptual model and on the programming constructs to be adopted to build systems in practice.

Taking a programming perspective, this work describes an infrastructural support allowing to seamlessly integrate various aspects characterizing an open MAS. In doing so, the notion of *Embodied Organization* is detailed, aimed at introducing each element in the MAS as an integral part of a structured infrastructure. In order to reconcile organizations, agents and environments, *Embodied organization* allows developers to focus on the main entities as separate concerns, and then to establish different interaction styles aimed to seamlessly integrate the various entities in a coherent system. In particular, the proposed approach defines a series of basic mechanisms related to the interaction model:

- i. How the agents could profitably interact with both organizational and other environmental entities in order to attain their design objectives;
- ii. How the organizational entities could control agent activities and regiment environmental resources in order to promote desired equilibrium;
- iii. How environmental changes could affect both organizational dynamics and agents activities;

The rest of the paper is organized as follows: Section 2 provides a survey of situated organization as proposed by existing works. Starting from the description of the basic entities characterizing an integrated perspective, Section 3 presents a unified programming model including agents, organizations and environments. The notion of Embodied Organization is detailed in Section 4, while Section 5 discusses a concrete programming model to implement it in practice.

Finally, Section 6 concludes the paper discussing the proposed approach and future directions.

2 Organizations situated in MAS Environments

Although early approaches in organization programming have not been addressed at modeling environments explicitly, recent trends are investigating the challenge to situate organizations in concrete computational environments. In what follows, a survey on related works is discussed, enlightening strengths and drawbacks of existing proposals.

2.1 Current Approaches

Several agent based approaches allow to implement situated organizations instrumenting computational environments where social interactions are of concern. A remarkable example of situated organization is due to Okuyama et al. [12], who proposed the use of “normative objects” as reactive entities inspectable by agents working in “normative places”. Normative objects can be exploited by the organization to make available information about norms that regulate the behavior of agents within the place where such objects can be perceived by agents. Indeed, they are supposed to indicate obligations, prohibitions, rights and are readable pieces of information that agents can get and exploit in computational environments. The approach envisages a distributed normative infrastructure which is assumed to control emergent dynamics and to allow agents to implicitly interact with a normative institution. The mechanism is based on the intuition that the reification of a particular state in a normative place may constitute the realization of a particular institutional fact (e.g., “being on a car driver seat makes an agent to play the role driver”). This basic idea is borrowed from John Searle’s work on speech acts and social reality [16, 17] Searle envisaged an institutional dimension rising out of collective agreements through special kind of rules, that he refers as *constitutive rules*. Those rules constitute (and also regulate) an activity the existence of which is logically dependent on the rules themselves, thus forming a kind of tautology for what a constitutive rule also defines the notion that it regulates. In this view, “being on a car driver seat makes an agent to play the role driver” strongly situate the institutional dimension on the environmental one, both regulating the concept of role adoption and, at the same time, defining it.

Constitutive rules in the form $X \text{ counts as } Y \text{ in } C$ are also at the basis of the formal work proposed by Dastani et al. [5]. Here a normative infrastructure (which is referred as “normative artifact”) is conceived as a centralized environment that is explicitly conceived as a container of *institutional facts*, i.e., facts related to the normative/institutional states, and *brute facts*, i.e. related to the concrete/ “physical” workplace where agents work. To shift facts from the brute dimension to the normative one the system is assumed to handle constitutive rules defined on the basis of “count-as” and “sanctioning” constructs,

which allows the infrastructure to recast brute facts to institutional ones. The mechanism regulating the application of “count-as” and “sanctioning” rules is then based on a monitoring process which is established as an infrastructural functionality embedded inside the normative system. Thanks to this mechanism, agents behavior can be automatically regulated through enforcing mechanisms, i.e. without the intervention of organizational agents.

A similar approach is proposed in the work by Tinnemeier et al. [20], where a normative programming language based on conditional obligations and prohibitions is proposed. Thanks to the inclusion of the environment dimension in the normative system, this work explicitly grounds norms either on institutional states either on specific environmental states. In this case indeed the normative system is also in charge of monitoring the outcomes of agent activities as performed in the work environment, in so doing providing a twofold support to the organizational dimension and to the environmental one.

With the aim to reconcile physical reality with institutional dimensions, an integral approach has been proposed with the MASQ approach, which introduces a meta-model promoting an analysis and design of a global systems along several conceptual dimensions [19]. The MASQ approach relies on the less recent AGR model, extended with an explicit support to environment as envisaged by the AGRE and AGREEN [1]. Four dimensions are introduced, ranging from endogenous aspects (related to agent’s mental attitudes) to exogenous aspects (related to environments, society and cultures where agents are immersed). In this case, the same infrastructure used to deploy organizational entities is also regulated by precise rules for interactions between agents and environment entities. The resulting interaction model relies on the theory of influences and reactions [9], in the context of which several interaction styles can be established among the heterogenous entities dwelling the system.

Besides conceptual and formal integration, few approaches have accounted a programming approach for situated organizations. By relating situated activities in the workplace, the *Brahms* platform endows human work practices and allows to represent the relations of people, locations, agent systems, communication and information content [18]. Based on existing theories of situated action, activity theory and distributed cognition, the *Brahms* language promotes the interplay of intelligent software agents with humans their organizations. A similar idea is provided by Situated Electronic Institutions (SEI) [4], recently proposed as an extension of Electronic Institutions (EI) [7]. Besides providing a runtime management of the normative specification of dialogic interactions between agents, the notion of observability of environment states is at the basis of SEI. They are aimed at interceding between real environments and EI. In this case, special governors, namely modelers, allow to bridge environmental structures to the institution by instrumenting environments with “embodied” devices controlled by the institutional apparatus. Participating agents can, in this case, perform individual actions and interactions (either non message based) while operating upon concrete devices inside the environment. Besides, SEI introduces the notion of staff agents, namely organization aware agents which role is to monitor ongoing

activities performed by agents which are not under the direct control of the institution. Staff agents are then assumed to bridge the gap between participating agents and the institutional dimensions: they typically react to norm violations, possibly ascribing sanctioning and enforcements to disobeying agents. Institutional control is also introduced by the mean of feedback mechanisms aimed at comparing observed properties with certain expected values. On the basis of possible not standard properties detected, an autonomic mechanism specifies how reconfigure the institution in order to re-establish equilibrium.

The ORA4MAS approach [11] proposed a programming model for concretely building systems integrating organizational functionalities in instrumented work environment. In ORA4MAS organizational entities are viewed as artifact based infrastructures. Specialized organizational artifacts (OAs) are assumed to encapsulate organizational functions, which can be exploited by agents to fulfill their organizational purposes. Using artifacts as basic building blocks of organizations, allows agents to natively interact with the organizational entity at a proper abstraction level, namely without being constrained to shape external actions as mechanism-level primitives needed to work with middleware objects. The consequence is that the infrastructure does not rely on a sort of hidden components, but the organizational layer is placed beside the agents as a suitable set of services and functionalities to be dynamically exploited (and created) as an integral part of the MAS work environment. On the other side, ORA4MAS does not provide an explicit support to environmental resources which are not included in the organizational specification. Two types of agents are assumed to evolve in ORA4MAS systems: *(i)* participating agents, assumed to join the organization in order to exploit its functions (i.e., adopting roles, committing missions etc.), while *(ii)* organization aware agents, assumed to manage the organization by making changes to its functional and structural aspects (i.e., creating and updating functional schemes or groups) or to make decisions about the deontic events (i.e. norm violations).

2.2 Open Issues and Challenges

Despite the richness of the models proposed for organizations of agents situated in computational environments, many aspects are still under discussion and have still to converge in a shared perspective between the different research lines. In the literature, this variety of approaches have been dealt with separately, each forming a different piece of a global view, with few consideration for how they could fit all together. On these basis, we here enlighten a series of current issues and challenges which our approach, described later on, is going to face with.

Agents/Organisations/Environments Interactions Typically interactions are based on a sub-agentive level, and are founded on protocols and mechanisms, instead on being based on the effective capabilities and functionalities exhibited by the entities involved in the whole system. Different approaches are provided for the interaction model between environment, agents and their organizations. Besides, there is not a clear vision on how environment and organizational en-

tities should support agents in their native capabilities, as for instance the ones related to action and perception.

Grounding Goals The computational treatments of goals clashes different approaches once they are referred to agents and their subjective goals, and when they are related to organizations and their global goals. For instance, approaches as MASQ, ORA4MAS describe in a rather abstract terms (i) *how* the subjective and global goals should be fulfilled in practice; (ii) *which* brute state has to be reached in order to consider a goal as achieved. By considering environments explicitly, either agents and organizations should be able to ground goals to actual environment configurations, thus recognizing the fulfillment of their objectives once the pursued goals have been reached in practice (this approach is adopted, for instance, in [5]). Other approaches, as for instance ORA4MAS [11], do not assume organizations able to automatically detect the fulfillment of global goals in terms of environment configurations.

Grounding Norms As for goals, a weak support is provided for grounding norms in concrete application domains, thus allowing to establish how and when a norm has been fulfilled or violated. Furthermore few studies have been addressed at managing norm lifecycle with respect to distributed and (highly) dynamic environments. No agreement is then established on which kind of monitoring and sanctioning mechanisms must be adopted. Some approaches envisage the role of organizational/staff agents [4], other approaches propose the sole automatic regulation provided by a programmable infrastructure [5, 20].

Agent Awareness It is not clear which kind of capability, and which grade of awareness, is required for agents to exploit the functionalities provided by the (situated) entities embedding organizational and environmental resources. Related to organizations, some approaches propose agents able to automatically internalize organizational specifications (i.e. MASQ, “normative objects”), other approaches, as (ORA4MAS and SEI) assume agents’ awareness to be *encoded* at a programming level.

Openness Concerns about interoperability and openness cross each of the above mentioned aspects. Few approaches account technological integration, for instance with respect to varying agent architectures, protocols and data types. Besides, the described proposals typically focus on a restricted set of interaction styles (i.e. dialogical interactions supported by an institutional infrastructure in SEI, environment mediated interactions in normative objects, an hybrid approach in ORA4MAS).

With the aim to respond the above mentioned challenges, the next sections describe an integrated approach aimed at devising a unified programming model seamlessly integrating agents, organizations and environments.

3 Unifying Agents, Organizations and Environments Programming

This section figures out the main elements characterizing an Embodied Organization. It envisages an integrated MAS in terms of societies of agents, envi-

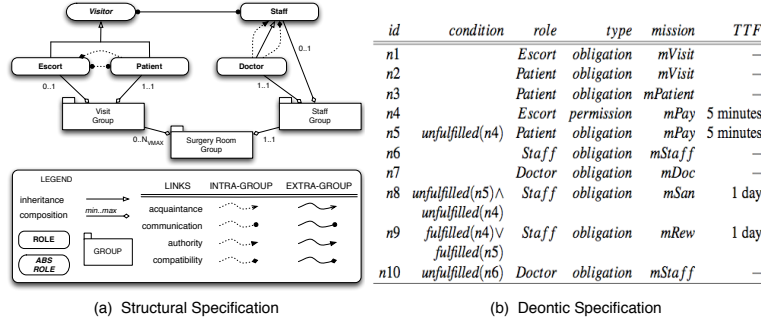


Fig. 1. Structural (a) and Normative (b) specifications for the hospital scenario, represented using the *Moise* graphical notation.

ronmental and organizational entities. In doing this, we refer to the consistent body of work already addressed at specifying existing computational models, while only the aspects which are relevant for the purposes of this work will be detailed. In particular, we refer to Jason [3] as agent development framework, CArTAgO [14] for environments and *Moise* [10] for organizations.

In order to ease the description, the approach will be sketched in the context of an hospital scenario. It summarizes the dynamics of an ambulatory room, and can be seen as an open system, where heterogenous agents can enter and leave in order to fulfill their purposes. In particular, two types of agents are modeled as organization participants. *Staff agents* (namely physicians and medical nurses) are assumed to cooperate with each other in order to provide medical assistance to visitors. Accordingly, *visitor agents* (namely patients and escorts) are assumed to interact themselves in order to book and exploit the medical examinations provided by the staff.

3.1 Organizations

The first considered dimension concerns the organization. We do adopt the *Moise* model, which allows to specify an organization based on three different dimensions referred as (i) structural, (ii) functional, and (iii) normative⁴. The Structural Specification (SS) provides the organizational structure in terms of groups of agents, roles and functional relations between roles (links). A role defines the behavioral scope of agents actually playing it, thus providing a standardized pattern of behavior for the autonomous part of the system. An inheritance relation can be specified, indicating roles that extend and inherit properties from parent roles. As showed in Fig. 1 (left), visitor agents can adopt two roles, patient and escort, both inheriting from a visitor abstract role. The doctor role

⁴ We here provide a synthesis of the *Moise* approach showing the specification of the hospital scenario. For a more detailed description, see [10].

is assumed to be played by a physician. It extends the properties of a more generic staff role, which is assigned in support and administration activities inside the group. Relationships can be specified between roles to define authorities, communication channels and acquaintance links. Groups consist in a set of roles and related properties and links. In the hospital scenario escorts and patients form visit groups, while staff and doctor form staff groups. The specification allows taxonomies of groups (i.e., escorts and patients forming visit group), and intra-group links, stating that an agent playing the source role is linked to all agents playing the target role. Notice that the cardinalities for roles inside a group are specified, indicating the maximum amount of agents allowed to play that role. The constraints imposed by the SS allow to establish global properties on groups, e.g. the well-formedness property means to complain role cardinality, compatibility, and so on.

The Functional Specification (FS) gives a set of functional schemes specifying how, according with the SS, various groups of agents are expected to achieve their global (organizational) goals. The related schemes can be seen as goal decomposition trees, where the root is a goal to be achieved by the overall group and the leafs are goals that can be achieved by the single agents. A mission defines all the goals an agent commits to when participating in the execution of a scheme and, accordingly, groups together coherent goals which are assigned to a role in a group. The FS for the hospital scenario (Fig. 2) presents three rehearsed schemes. The visitor scheme (*visitorSch*) describes the goal tree related to the visitor group. It specifies three missions, namely *mVisit* as the mission to which each agent joining the visit group has to commit, *mPatient* as the mission to be committed by the patient who has to undergo the medical visit, and *mPay* as the mission to be committed by at least one agent in the visit group. Notice that the goals “do the visit” (which is related to the mission *mPatient*) and “pay visit” (which is related to the mission *mPay*) can be fulfilled in parallel. The *monitorSch* describes the activities performed by a staff agent. These plans are aimed at verifying if the activities performed by the visitors follow an expected outcome, namely if the visitors fulfill the payment committing the *mPay* mission (which includes the “pay visit” goal). Finally, the *docSch* specifies the activities to which a doctor has to commit, namely to perform the visit to every patient. Notice that each mission has a further property specifying the maximum amount of time than an agent has to commit to the mission (“time to fulfill”, or *ttf* value). The FS also defines the expected cardinality for every mission in the scheme, namely the number of agents inside the group who may commit a given mission without violating the scheme constraints.

The Normative Specification (NS) relates roles (as they are specified in the SS) to missions (as they are specified in the FS) by specifying a set of norms. *Moise* norms result in terms of *permissions* or *obligations* to commit to a mission. This allows goals to be indirectly related to roles and groups, i.e. through the policies specified for mission commitment. Fig. 1 (right) shows the declarative specification of the norms regulating the hospital scenario, and refers to the missions described in Fig. 2. “Time to fulfill” (*ttf*) values refer to the maximum

amount of time the organization expects for the agent to fulfill a norm. For instance, norms $n1$ and $n2$ define an obligation for agents playing either patient and escort roles to commit to the $mVisit$ mission. A patient is further obliged to commit to $mPatient$ mission ($n3$). The norm $n10$ is activated only when the norm $n6$ is not fulfilled: It specifies an obligation for a doctor to commit the $mStaff$ mission, if no other staff agent is committing to it inside the group. Based on the constraints specified within the SS and FS, the NS is assumed to include an additional set of norms which are automatically generated in order to control role cardinality, goal compliance, deadline of commitments, etc.

The concrete computational entities based on the above detailed specification have been developed based on an extended version of ORA4MAS [11]. This programming approach envisages organizational artifacts (OA) are those non-autonomous computational entities adopted to reify organizations at runtime, thereby implementing the *institutional* dimension within the MAS. In particular, ORA4MAS adopts two types of artifacts, referred as *scheme* and *group* artifacts, which manage the organizational aspects as specified in *Moise's* functional, structural and normative dimensions. The resulting system has been referred as Organizational Management Infrastructure (OMI), where the term infrastructure can be understood from an agent perspective: it embeds those organizational functionalities exploitable by agents to participate the organizational activities and to access organization resources possibly exploiting, creating and modifying OAs on the need. Of course, in order to suitably exploit the OMI functionalities, agents need to be equipped with special capabilities and knowledge about the organizational structures, that is what in Subsection 2.2 we refer as agent awareness.

3.2 Environments

As said in Subsection 2.1, the ORA4MAS approach does not support environments besides organizational functionalities. To this end, dually to the OMI, an Environment Management Infrastructure (EMI) is introduced to embed the set of environmental entities aimed at supporting pragmatic functionalities. While artifacts are adopted as basic building blocks to implement the EMI, environments also make use of workspaces (e.g., an *Hospital* workspace is assumed to contain the hospital infrastructures). Artifacts are adopted in this case to provide a concrete (*brute*) dimension – at the environment level – to the global system. Workspace are adopted in order to model a notion of locality in terms of an application domain.

As Fig. 2 shows, it is quite straightforward to find a basic set of Environment Artifacts (EA) building the EMI. Taking an agent perspective, the developer here simply imagines which kind of service may be required for the fulfillment of the various missions/goals, thus mapping artifact functionalities to the functional specification given by the *Moise* FS.

Designing an EMI is thus not dissimilar to instrumenting a real workplace in the human case: (*i*) to model the hospital room it will be used a specialized *hospital* workspace, (*ii*) to automate bookings it will be provided a *Desk* artifact,

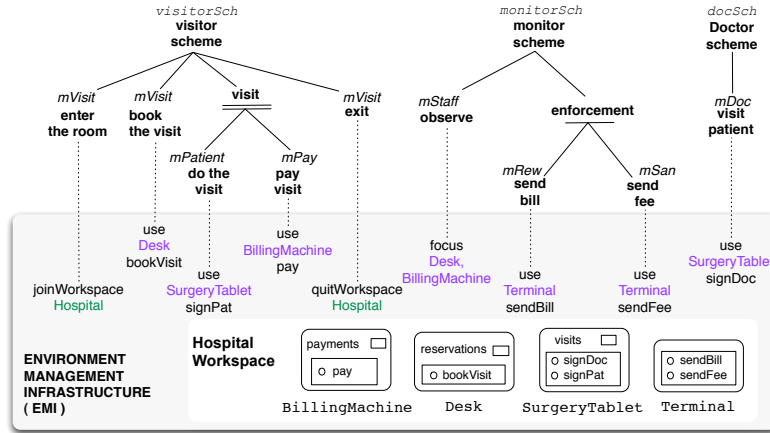


Fig. 2. (Above) *Moise* Functional Specification (FS) for the hospital scenario. Schemes are used to coordinate the behavior of autonomous agents. (Below) FS is used to find a set of environmental artifacts, and to map their functionalities in the EMI.

(iii) to finalize visits it will be provided a (program running on an) *Surgery Tablet* artifact, (iv) to automate payments it will be provided a *Billing Machine* artifact, and (v) to send fees and bills it will be provided a *Terminal* artifact.

3.3 Agents

Besides the abstract indication of the different artifacts exploitable at the environment level, the Fig. 2 also shows the actions to be performed by agents for achieving their goals. Thanks to the *CARTAgO* integration technology, several agent platforms are actually enabled to play in environments: seamless interoperability is provided by implementing a basic set of actions, and related perception mechanisms, allowing agents to interact with artifacts and workspaces [14, 15]. Those actions are directly mapped into artifact operations (functions), or addressed to the workspace: in the case of the EMI, a *Jason* agent has to perform a `joinWorkspace("Hospital")` action to enter the room (which is related to the *mVisit* mission); to book the visit (related to the *mVisit* mission) the action `bookvisit() [artifact_name("Desk")]` has to be performed on the desk artifact, and so on (see Fig. 2, below).

The same semantic mapping agents' actions into artifact operations is adopted to describe interactions between agents and OMI: e.g., `commitMission` is an operation that can be used by agents upon the *scheme* artifact to notify mission commitments; `adoptRole` (or `leaveRole`) can be used by an agent upon the *group* artifact in order to adopt (leave) a given role inside the group, etc.

Fig. 3 (left) shows a global picture of the resulting system. As showed, agents fulfill their goals and coordinate themselves by interacting with EMI artifacts,

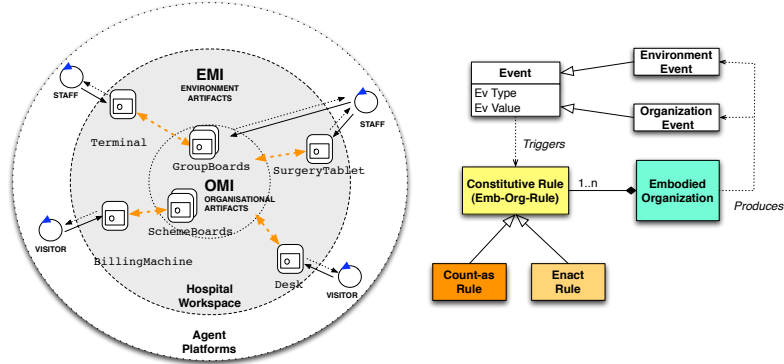


Fig. 3. (Left) Global view of the system presents an open set of agents at work with infrastructures managing Environment and Organization. Functional relationships between EMI and OMI are established by count-as and enact rules. (Right) Meta-model for Organizational Embodied Rules, used to implement count-as and enact rules.

while staff agents, which we assume as special agents aware of organizational functionalities, can directly interact with the OMI. Both these dimensions are an integral part of the global infrastructure and, most important, can be dynamically exploited by agents to serve their purposes. From an agent perspective, the whole system can be understood as a set of *facts* and *functions*, which are exploited, from time to time, to the organizational and environmental dimensions. Through artifacts, the global infrastructure provides observable states, namely information readable by agents for improving their knowledge. Artifacts also provide operations, namely process based functionalities, aimed at being exploited by agents for externalizing activities in terms of external actions. Thus, the epistemic nature of observable properties can be addressed to the informational dimension of the whole infrastructure, while the pragmatic nature of artifact operations is assumed to cover the functional dimension.

4 Embodied Organizations

As far as the global system is conceived, EMI and OMI are situated side by side inside the same work environment, but they are conceived as separated systems. They are assumed to face distinct application domains, the former being related to concrete environment functionalities and the latter dealing specifically with organizational ones. The notion of Embodied Organization provides a more strict integration: it further identifies and implements additional mechanisms and conceives a unified infrastructure enabling functional relationships between EMI and OMI. As some of the approaches discussed in Section 2, we theoretically found this relationship on Searle's notion of constitutive rules. Differently from other approaches, we ground the notion of Embodied Organization

on a concrete programming model, as the one who lead us to the implementation of EMI and OMI. As explained below, Embodied Organizations rely on a revised management of events in `CARTAgO`, and can be specified by special programming constructs referred as `Emb-Org-Rules`.

4.1 Events

A crucial element characterizing Embodied Organizations is given by the renewed workspace kernel based on events. Events are records of significant changes in the application domain, handled at a platform level inside `CARTAgO`. They are referred to both state and processes to represent the transitions of configurations inside workspaces. Each event is represented by a type,value pair $(\langle ev_t, ev_v \rangle)$: Event *type* indicates the type of the event (i.e., `join_req` indicating agents joining workspace, `op_completed` indicating the completion of an artifact operation, `signal` indicating events signalled within artifact operation execution, and so on); Event *value* gives additional information about the event (i.e., the source of the event, its informational content, and so on). Due to the lack of space, the complete list of events, together with the description of the mechanism underlying event processing, can not be described here. The interested reader can find the complete model, including the formal transition system, in [13]. We here emphasize the relevance of events, which have the twofold role (*i*) to be perceived or triggered by agents (i.e. focusing/using artifacts) and (*ii*) to be collected and ranked within the workspace in order to trace the global dynamic of the system.

4.2 Embodied Organization Rules

While the former role played by events refers to the interaction between agents and artifacts, the second role is exploited to identify, and possibly govern, intra-workspace dynamics. On such a basis, the notion of *Embodied Organization* refers to the particular class of situated organization structured in terms of artifact based infrastructures and governed by constitutive rules based on workspace events. Events are originated within the infrastructure, being produced by environmental and organizational entities. Computing constitutive rules is realized by `Emb-Org-Rule`, which consist of a programmable constructs “gluing” together organizational and environmental dimensions. An abstract model of this process is shown by the dotted arrows between EMI and OMI in Fig. 3 (right). Structures defining `Emb-Org-Rule` refer to *count-as* and *enact* relations.

Count-as rules state which are the consequences, at the organizational level, for an event generated inside the overall infrastructure. They indicate how, since the actions performed by the agents, the system automatically detects relevant events, thus transforming them to the application of a set of operators aimed at changing the configuration of the Embodied Organization. In so doing, either relevant events occurring inside the EMI (possibly triggered by agents actions), either events occurring in the context of the organization itself (OMI) can be vehicled to the institutional dimension: these events can be further translated in

```

+join_req(Ag)
-> make("visitorGroupBoard",
"OMI.GroupBoard",
["moise/hospital.xml","visitGroup"]);
    make("visitorSchBoard",
"OMI.SchemeBoard",
["moise/hospital.xml","visitorSch"]);
    apply("visitorGroupBoard",
adoptRole(Ag, "patient"));
    include(Ag).

+op_completed("visitorGroupBoard", _,
adoptRole(Ag, "patient"))
-> apply("visitorSchBoard",
commitMission(Ag, "mPat")).

+ws_leaved(Ag)
-> apply("visitorGroupBoard",
leaveRole(Ag, "patient")).

+op_completed("BillingMachine",
Ag, pay)
-> apply("visitorSchBoard",
setGoalAchieved(Ag, pay_visit)).

+op_completed("Terminal",
Ag, sendFee)
-> apply("monitorSchBoard",
setGoalAchieved(Ag, send_fee)).

```

Table 1. Example of Emb-Org-Rule (count-as) in the hospital scenario.

```

+signal("visitorGroupBoard",
role_cardinality, visitor)
-> disable("Desk", bookVisit).

+signal("monitorSchBoard",
goal_non_compliance,
obligation(Ag,
ngoal(monitorSch,mRew,send_bill),
achieved(monitorSch,send_bill,Ag), TTF)
-> exclude(Ag).

```

Table 2. Example of Emb-Org-Rule (enact) in the hospital scenario.

the opportune institutional changes inside the OMI, that is assumed to update accordingly.

Enact rules state, for each institutional event, which is the control feedback at the environmental level. Hence, *enact* rules express how the organizational entities automatically control the environmental ones. The use of enact rules allows to exploit organizational events (i.e. role adoption, mission commitment) in order to elicit changes in the environment.

5 Programming Embodied Organizations

Embodied Organizations enable a unified perspective on agents, organizations and environments by conceiving an interaction space based on a twofold infrastructure governed by events and constitutive rules (Emb-Org-Rules). In this section examples of programming such rules are discussed.

Programming Count-as Rules According to the *Moise* FS previously defined, the organization expects that an agent va_{id} joining the hospital workspace is assumed to play the role visitor, which purpose is to book a medical visit and possibly achieve it. Thus, an event $join_req, \langle va_{id}, t \rangle$, dispatched once an agent va_{id} tries to enter the workspace, from the point of view of the organization “count-as” creating a new position related to the visit group. Making the event $join_req$ to “count as” va_{id} adopting the role visitor, is specified by the first rule in Table 1 (left): it states that since an event signalling that an agent Ag is joining the workspace, an Emb-Org-Rule must be applied to the system. The body of the rule specifies that two new instances of organizational artifacts related to the visit group will be created using the make operator. In this case the new

artifacts will be identified by `visitorGroupBoard` and `visitorSchBoard`. The following operator constitutes the new role inside the group: `apply` acts on the `visitorGroupBoard` artifact just created by automatically making the agent *Ag* to adopt the role `patient`. Finally, once the adopt role operator succeeds, the last operator includes the agent *Ag* in the workspace.

In the above described scenario, the effect of the application of the rule provides an institutional outcome to the `joinWorkspace` actions. Besides joining the workspace, a sequence of operators is applied establishing what this event means in organizational terms. When the effects of the role-adoption are committed, as previously described, a new event is generated by the group board: $\langle \text{op_completed}, \langle \text{"visitorGroupBoard"}, va_{id}, \text{adoptRole}, \text{patient} \rangle \rangle$. For the organization, such an event may “count-as” committing to mission *mPat* on the `visitorSchBoard`. This relation is specified by the second rule in Table 1, where a `commitMission` is applied to the `visitorSchBoard` for the mission *mPat*. Similarly, an event $\langle \text{ws_leaved}, \langle va_{id}, t \rangle \rangle$, signalling that the visitor agent has left the workspace, from an organizational perspective “count-as” leaving the role `patient`. This relation is specified by the first rule in Table 1 (right), where a `leaveRole` is applied to the `visitorGroupBoard` for the role `patient`. At the same time, an event like $\langle \text{op_completed}, \langle \text{BillingMachine}, va_{id}, \text{pay}, t \rangle \rangle$ signals that a visitor agent has successfully finalized the pay operation upon the billing machine. Such an event “count-as” having achieved the goal `pay visit` on the `visitorSchBoard` (second rule in Table 1, right). Finally, an event $\langle \text{op_completed}, \langle \text{Terminal}, sa_{id}, \text{sendFee}, t \rangle \rangle$, signalling that a staff agent has successfully used the terminal to send the fee to a given patient, “count-as” having achieved the goal `send fee` (third rule in Table 1, right).

Programming Enact Rules *Enact* effects are defined to indicate how, from the events occurring at the institutional level, some control feedback can be applied to the environmental infrastructure. As far as the execution of the operations is conceived in *CARTAgO*, the OMI automatically dispatches events signalling ongoing violations. Violations are thus organizational events which may suddenly elicit the application of some *enact* rule used to regiment the environment.

In Table 2, a regimentation is installed by the organization thanks to the enact rule stating that an event $\langle \text{signal}, \langle \text{visitorGroupBoard}, \text{role_cardinality}, \emptyset, t \rangle \rangle$ signalled by the `visitorGroupBoard` indicates the violation for the norm `role_cardinality`. The related enact rule is given in Table 2 (left), where the reaction to this event is specified in order to disable the book operation on the desk artifact, for all the agents inside the workspace. The absence of any parameter related to agent identifier in the `disable("Desk", bookVisit)` operator makes the disabling to affect the overall set of agents inside the workspace. Similarly, violating the obligation imposed to the staff agent to fulfill sanctioning and rewarding missions elicits the scheme board assigned to the *monitorSch* to signal the event $\langle \text{signal}, \langle \text{monitorSchBoard}, \text{goal_non_compliance}, \text{obligation}(\text{Ag}, \text{ngoa}(\text{monitorSch}, \text{mRew}, \text{send_bill}), \text{achieved}(\text{monitorSch}, \text{send_bill}, \text{Ag}), \text{TTF}), t \rangle \rangle$. This event is generated thanks to a special norm (called `goal_non_compliance`) which is automatically generated since the *Moise* specification and stored in-

side the OMI. Due to the enact rule specified in Table 2 (right), this causes the exclusion for the `Ag` agent from the hospital workspace.

6 Conclusion and Perspectives

In this paper Embodied Organizations have been introduced as a unified programming model promoting a seamless integration of environmental and organizational dimensions of a MAS. A series of responses to the challenges envisaged in Subsection 2.2 could be listed: **Infrastructures.** Either environmental and organizational entities are implemented in concrete infrastructures instrumenting workspaces, decentralized in specialized artifacts which serve informational and operational functions. **Interaction.** The approach establishes a coherent semantic for agent - infrastructure interactions, Embodied Organizations define functional relationships between the heterogenous entities at the basis of organizations and environments. These are placed in terms of programmable constructs (**Emb-Org-Rules**), governed by workspace events and inspired by Searle's notion of constitutive rules. **Goals and Norms.** Implementing organizations in concrete environments allows to deal explicitly with goals and norms, which fulfillment can be structurally monitored and promoted at the organizational level through the use of artifacts. **Awareness.** Embodied Organizations are aimed to fit the work of agents and accordingly to allow them to externalize pragmatic and organizational activities. The use of **Emb-Org-Rule** automates and promotes specific organizational patterns, to which agents may effortlessly participate simply by exploiting environmental resources. Artifacts can be used in goal oriented activities, and, most important, without the need to be aware of organizational notions like roles, norms, etc. **Openness.** Technological interoperability is ensured at a system level, by providing mechanisms for agent-artifact interactions which are based on a coherent semantic. Besides, several interaction styles can be established at an application level, being agents mediated by infrastructures which can be modified, replaced and created on the need.

Future work will be addressed at covering missing aspects, such as the dialogical dimension of interactions, and the inclusion of real embodied entities in the system (i.e., humans, robots, etc.). An important objective is the definition of a general purpose approach, towards the full adoption of the proposed model in the context of concrete application domains and mainstream agent oriented programming.

References

1. José-Antonio Báez-Barranco, Tiberiu Stratulat, and Jacques Ferber. A unified model for physical and social environments. In *Environments for Multi-Agent Systems III, Third International Workshop (E4MAS 2006)*, volume 4389 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2006.
2. Olivier Boissier, Jomi Fred Hübner, and Jaime Simão Sichman. Organization Oriented Programming: From Closed to Open Organizations. In *Engineering Societies*

- for Agent Worlds (ESAW-2006). *Extended and Revised version in Lecture Notes in Computer Science LNCS series*, Springer, pages 86–105, 2006.
3. Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldrige. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons, 2007.
 4. Jordi Campos, Maite Lòpez-Sánchez, Juan A. Rodríguez-Aguilar, and Marc Esteva. Formalising Situatedness and Adaptation in Electronic Institutions. In *COIN-08, Proc.*, 2008.
 5. Mehdi Dastani, Nick Tinnemeier, and John-Jules CH. Meyer. A programming language for normative multi-agent systems. In *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI-Global, 2009.
 6. Virginia Dignum, editor. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI-Global, 2009.
 7. Marc Esteva, Juan A. Rodríguez-Aguilar, Bruno Rosell, and Josep L. AMELI: An agent-based middleware for electronic institutions. In *Proceedings of International conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, pages 236–243, New York, 2004. ACM.
 8. Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From Agents to Organizations: An Organizational View of Multi-agent Systems. In *Proceedings of (AOSE-03)*, volume 2935 of *Lecture Notes Computer Science (LNCS)*. Springer, 2003.
 9. Jacques Ferber and Jean-Pierre Müller. Influences and Reaction: a Model of Situated Multi-Agent Systems. In *Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96)*. AAAI, 1996.
 10. Jomi F. Hübner, , Jaime S. Sichman, and Olivier Boissier. Developing organised multi-agent systems using the MOISE+ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3/4):370–395, 2007.
 11. Jomi F. Hübner, Olivier Boissier, Rosine Kitio, and Alessandro Ricci. Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents. *Journal of Autonomous Agents and Multi-Agent Systems*, April 2009.
 12. Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa. A Distributed Normative Infrastructure for Situated Multi-Agent Organisations. In *Decl. Agent Lang. & Techn. (DALT-VI)*, volume 5397 of *LNCS*. Springer, 2009.
 13. Michele Piunti. *Designing and Programming Organizational Infrastructures for Agents situated in Artifact-based Environments*. PhD thesis, ALMA MATER STUDIORUM Università di Bologna, April 2010.
 14. Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: An artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 2010. Springer, ISSN 1387-2532 (Print) 1573-7454 (Online).
 15. Alessandro Ricci, Andrea Santi, and Michele Piunti. Action and Perception in Multi-Agent Programming Languages: From Exogenous to Endogenous Environments. In *Proceedings Programming Multiagent Systems (PROMAS-10)*, 2010.
 16. John R. Searle. *Speech Acts*, chapter What is a Speech Act? Cambridge University Press, 1964.
 17. John R. Searle. *The Construction of Social Reality*. Free Press, 1997.
 18. M. Sierhuis. *Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design*. PhD thesis, University of Amsterdam, SIKS Dissertation Series, 2001.
 19. Tiberiu Stratulat, Jacques Ferber, and John Tranier. MASQ: Towards an Integral Approach of Agent-Based Interaction. In *Proc. of 8th Conf. on Agents and Multi Agent Systems (AAMAS-09)*, 2009.

20. Nick Tinnemeier, Mehdi Dastani, J.-J.Ch. Meyer, and L. van der Torre. Programming normative artifacts with declarative obligations and prohibitions. In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009)*, 2009.
21. M. Birna van Riemsdijk, Koen Hindriks, and Catholijn Jonker. Programming organisation-aware agents: a research agenda. In *In 10th Engineering Societies for Agents Worlds (ESAW 09)*, 2009.
22. Danny Weyns, Andrea Omicini, and James J. Odell. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5-30, February 2007. Special Issue on Environments for Multi-agent Systems.

Group Intention is Social Choice with Commitment

Guido Boella¹, Gabriella Pigozzi², Marija Slavkovic², and Leendert van der Torre²

¹ `guido@di.unito.it`

² `{gabriella.pigozzi, marija.slavkovic, leon.vandertorre}@uni.lu`

Abstract. A collaborative group is commonly defined as a set of agents, which share information and coordinate activities while working towards a common goal. How do groups decide which are their common goals and what to intend? According to the much cited theory of Cohen and Levesque, an agent intends g if it has chosen to pursue goal g and it has committed itself to making g happen. Following the same line of reasoning, a group intention should be a collectively chosen goal with commitment. The literature often considers a collective goal to be one of those individual goals that are shared by all members. This approach presumes that a group goal is also an individual one and that the agents can act as a group if they share the beliefs relevant to this goal. This is not necessarily the case. We construct an abstract framework for groups in which common goals are determined by social choice. Our framework uses judgment aggregation to choose a group goal and a multi-modal multi-agent logic to define commitment and revision strategies for the group intentions.

1 Introduction

An agent acts according to its beliefs and intentions. According to what does a group act? We would expect that in order for groups to act, jointly or by coordinating their activities, they need to establish what to believe i.e., form epistemic attitudes, and what to aim for, i.e., to form motivational attitudes. In existing frameworks for collaborative activities [9, 12, 16] and group decision-making protocols [10], the formation of group attitudes is defined only for a specific type of groups. These groups consist of agents that engage in pursuing a group goal only when the members have the same beliefs regarding this goal, or when they are successful in reaching an agreement on a given set of beliefs.

Consider a group of robots in charge of office building maintenance. One candidate group goal for them is to clean the meeting room. The decision rule to adopt the goal is that the room needs to be cleaned when the following conditions (reasons) are met: the floors are dirty or the garbage bin is full, and people do not occupy the room. To decide whether to pursue this goal, the robots need to decide if the reasons to adopt the goal are true. The robots cannot check whether the room is occupied or what state it is in. Hence, to estimate the state of affairs, the robots need to rely on their individual beliefs, which may diverge.

Assume there are three robots in the group. One robot believes that the room is occupied and thus, according to it, the group should not adopt the goal. According to the other two robots, the group should adopt the goal. One robot believes that the garbage bin is full and the floors clean and the other that the floors are dirty. The question is how to should one aggregate the beliefs of the robots in this case? The majority of the robots would estimate that the goal should be adopted. However the group is not univocal

and as a result, the goal would not be chosen for a group goal when the robots reason according to [9, 12, 16]. A general method for forming group attitudes needs to specify how group attitudes are formed also when agents have disagreeing beliefs or limited persuasive abilities. A framework that allows for such general methods is still lacking.

Consider now that the group adopts the goal, but before pursuing it, the robots learn that there is a seminar scheduled in the meeting room. The group has to de-commit from their intention to clean the room. However, to be able to do so, the group has to have a commitment strategy that allows de-committing upon change in the reasons for the goal adoption. Furthermore, the group needs to be able to reconsider its reasons for goals, and goals, after de-committing. If they simply drop the goal, without “remembering” why, they would not be able to re-deliberate and commit again once the seminar is over.

An intelligent agent reacts to the changes in its environment and a group should be capable of doing the same. When new information becomes available the group faces a choice: to remain committed to its group attitudes or to reconsider them. The most sensitive commitment strategy proposed by Rao and Georgeff [21] allows for a group reaction only when the goal is accomplished or impossible. However, the existence of a goal is intertwined with the existence of some beliefs [2]. Consequently, there is a need for a commitment strategy that reacts to new information regarding those beliefs on which the goal hinges. Furthermore, groups need to know not only when to de-commit from their goals but also how to reconsider their goal-related beliefs.

The research question we address in this paper is:

How can groups choose and reconsider their goals?

A good methodology for collectively choosing and reconsidering goals is one that can be used by any group of agents regardless of the homogeneity of the individual beliefs of its members and their persuasion abilities. The relation between individual goals and beliefs can be specified and analyzed in modal agent logics like BDI_{LTL} [22]. The challenge in group goal generation is to incorporate the aggregation of individual attitudes into collective ones, as studied in merging, judgment aggregation and social choice [3, 11, 13, 14, 17]. However, using this approach is not straightforward. The main difficulty lies in the inability to use judgment aggregation directly in a BDI_{LTL} framework. Properties of judgment aggregation and modal agent logic are of different kinds, as they were initially developed for different purposes.

Our research question thus breaks down into the following sub-questions:

1. *How to aggregate individual (epistemic and motivational) attitudes?*
2. *What are the desirable properties for this aggregation?*

A good methodology for collectively choosing and reconsidering goals is also a methodology that is dynamic enough to allow for the group to change its epistemic and motivational attitudes in light of new information. Having such a methodology increases the autonomy and reactivity of groups. Hence we need to answer the following sub-question as well:

3. *Which commitment and reconsideration strategies should be available for groups?*

We thus focus on finding the following solutions:

Formal framework. We extend a multi-agent modal language to be able to represent judgment aggregation in it.

Choice of aggregation. Judgment aggregation is an abstract framework that allows for various desirable properties to be specified for the aggregation procedure. The task is to determine which aggregation properties are necessary and desirable for aggregating individual beliefs and goals.

Commitment and reconsideration strategies. Within our formal framework, we define *when* to de-commit from intentions and *how* to change them.

Multiple goals. Since a group can have more than one goal, we need to model the effect that the commitment to (and reconsideration of) one goal has over the commitment to (and reconsideration of) other goals.

We make the following assumptions. The group has a fixed membership – agents do not join or depart from the group. The group has a set of candidate group goals and an order of priority over these goals. The decision rules for each candidate goal are available to the group. How the decision rules are learned is outside of the scope of this paper. Here we do not consider how plans are generated, executed and revised once the group goals are selected or reconsidered, nor we consider whether the group goals are executed via individual or joint actions.

The layout of the paper is as follows. In Section 2 we introduce judgment aggregation. In Section 3 we extend a multi-modal agent formalism to capture the aggregation of individual attitudes. Sections 4 and 5 respectively study the commitment and reconsideration strategies. Related work, conclusions and outlines for future work are in Section 6.

2 From individual attitudes to group goals

Let us consider again the example of the robot cleaning crew from Section 1.

Example 1. Let $C = \{w_1, w_2, w_3\}$ be a crew of cleaning robots. We denote the group goal to clean the meeting room with g_1 , and the reasons to adopt this goal with: there are no people in the room (p_1), the room is dirty (p_2), the garbage bin in it is full (p_3). The individual beliefs of the robots on whether g_1 should be the group goal are justified by individual beliefs on p_1, p_2, p_3 using the decision rule $(p_1 \wedge (p_2 \vee p_3)) \leftrightarrow g_1$.

A group of agents could collectively decide to adopt or reject a goal by voting. However, the goals of the agents are not independent from their beliefs [2], which we express using decision rules. When the decision is whether to adopt a goal or not, we also need to explain why this goal should (not) be adopted. Having reasons for (not) adopting a goal enables the agents to re-considerer this goal in light of new information. Judgment aggregation deals with the problem of reaching decisions for a set of logically dependent issues, by aggregating individual opinions on these issues.

2.1 Judgment aggregation preliminaries

A general overview of judgment aggregation is given in [17]. Here we present the terminology and definitions of judgment aggregation we use in our framework.

Consider a logic \mathcal{L} with entailment operator \models . In a judgment aggregation framework, an agenda $\mathcal{A} \subseteq \mathcal{L}$ is the pre-defined set of issues, on which every agent casts her judgments. E.g., the agenda of Example 1 is $\mathcal{A} = \{p_1, p_2, p_3, g_1\}$. Consider a valuation function v such that a judgment “yes” on issue a is a valuation $v(a) = 1$, while a judgment “no” on the same issue is a valuation $v(a) = 0$. A profile is the set of all judgments assigned, on the agenda issues, by the decision-making agents.

Definition 1 (Profile). Let $\mathcal{N} = \{1, 2, \dots\}$ be a set of agent names, $\mathcal{A} \subseteq \mathcal{L}$ an agenda and $\bar{\mathcal{A}} = \mathcal{A} \cup \{\neg a \mid a \in \mathcal{A}\}$. A profile π is the set of judgments for the agenda items, submitted by the agents in the group: $\pi \subseteq \mathcal{N} \times \bar{\mathcal{A}}$. We define two operators: The judgment set for agent i is $\pi \Rightarrow i = \{a \mid (i, a) \in \pi\}$. The set of all the agents who judged “yes” to $a \in \bar{\mathcal{A}}$ is $\pi \Downarrow a = \{i \mid (i, a) \in \pi\}$.

Definition 1 extends the definition of a profile given [17] with the operators \Rightarrow and \Downarrow . We introduce these operators to ease the explanation of the various aggregation properties we present later on. To get a better intuitive grasp on these operators, the reader should envision the profile as a two-dimensional object with the agenda items identifying the columns and the agents identifying the rows:

$$\pi = \begin{matrix} & p_1 & p_2 & p_3 & g_1 \\ w_1 & \left\{ \begin{matrix} 1 & 1 & 0 & 1 \end{matrix} \right\} \\ w_2 & \left\{ \begin{matrix} 0 & 1 & 1 & 1 \end{matrix} \right\} \\ w_3 & \left\{ \begin{matrix} 1 & 0 & 0 & 0 \end{matrix} \right\} \end{matrix}$$

π is a possible profile for Example 1. We identify $\pi \Rightarrow w_2$ as the row labeled w_2 and $\pi \Downarrow p_2$ as the 1 entries in the column labeled p_2 , which identify the agents who casted judgement “yes” on p_2 .

In judgment aggregation, the judgments over \mathcal{A} , both individual and aggregates, are constrained by decision rules $\mathcal{R} \subseteq \mathcal{L}$. The set \mathcal{R} contains only formulas r such that all the non-logical symbols of \mathcal{A} occur in $r \in \mathcal{R}$. For instance, in Example 1, the decision rule is: $(p_1 \wedge (p_2 \vee p_3)) \leftrightarrow g_1$ and all agents respect it. In general, each agent could follow a different decision rule and yet another decision rule can be imposed for the group. In judgment aggregation, the agents are allowed to submit only those judgment sets which are consistent with $r \in \mathcal{R}$. Often, the agents are also required to cast judgments on all the agenda issues. We construct Definition 2 to formalize what it means for a judgment set to be admissible.

Definition 2 (Admissible profiles). A judgment set $\pi \Rightarrow i$ is complete if and only if for every $a \in \mathcal{A}$, either $(i, a) \in \pi$ or $(i, \neg a) \in \pi$. A judgment set $\pi \Rightarrow i$ is consistent with $r \in \mathcal{R}$ if and only if $\{r\} \cup (\pi \Rightarrow i) \not\models \perp$. A judgment set $\pi \Rightarrow i$ is admissible if it is consistent with the given $r \in \mathcal{R}$ and complete for \mathcal{A} . The set of all admissible judgment sets for a given \mathcal{A} and r is denoted by \mathcal{W} . A profile π is admissible if $\pi \Rightarrow i$ is admissible for all $i \in \mathcal{N}$. The set of all admissible profiles for agents \mathcal{N} is denoted by Π .

We denote $con(r, \varphi) = 1$ if φ is consistent with r and $con(r, \varphi) = 0$ otherwise. For the profile in Figure 1, $con((p_1 \wedge (p_2 \vee p_3)) \leftrightarrow g_1, \pi \Rightarrow i) = 1$ for every $i \in \mathcal{C}$.

We now present the definition of a judgment aggregation function, as given in [17].

Definition 3 (Judgment aggregation function). Given an agenda \mathcal{A} and agents \mathcal{N} , a judgment aggregation (JA) function is $f : 2^{\mathcal{N} \times \overline{\mathcal{A}}} \mapsto 2^{\overline{\mathcal{A}}}$.

We refer to $f(\pi)$ as the collective judgment set for \mathcal{N} . We denote the result of $f(\pi)$ with \perp when the JA function produces an inadmissible judgment set. If $f(\pi) \neq \perp$ then we call $f(\pi)$ a decision and denote it by \mathcal{D}_π . Figure 1 illustrates an example of a judgment aggregation function and profile, for which the collective judgment set is \perp because $\text{con}((p_1 \wedge (p_2 \vee p_3)) \leftrightarrow g_1, \{p_1, p_2, p_3, \neg g_1\}) = 0$.

The JA function is a useful abstraction, because many properties of judgment aggregation can be defined in terms of it. It then can be studied which properties can be accepted together (avoiding impossibility results). Given a JA function f , we describe the most common properties in the JA literature.

Universal domain. f satisfies universal domain if and only if its domain is Π .

Anonymity. Given a profile $\pi \in \Pi$, let $\hat{\pi} = \{\pi \Rightarrow 1, \dots, \pi \Rightarrow n\}$, be the multiset of all the individual judgment sets in π . Two profiles $\pi, \pi' \in \Pi$ are permutations of each other if and only if $\hat{\pi} = \hat{\pi}'$. f satisfies anonymity if and only if $f(\pi) = f(\pi')$ for all permutation π and π' .

Unanimity on $a \in \overline{\mathcal{A}}$. f satisfies unanimity on $a \in \overline{\mathcal{A}}$ if and only if for every profile $\pi \in \Pi$ it holds: if for all $i \in \mathcal{N}$, $(i, a) \in \pi$, then $a \in f(\pi)$.

Collective rationality. f satisfies collective rationality if and only if for all $\pi \in \Pi$, $\text{con}(r, f(\pi)) = 1$ for a given $r \in \mathcal{R}$, and either $a \in f(\pi)$ or $\neg a \in f(\pi)$ for every $a \in \mathcal{A}$.

Constant. f is constant when there exists $\varphi \in 2^{\overline{\mathcal{A}}}$ such that for every $\pi \in \Pi$, $f(\pi) = \varphi$.

Independence. Let $\Phi = \{\pi \Downarrow a \mid a \in \mathcal{A}\}$ for every $\pi \in \Pi$. Let f_1, \dots, f_m be functions defined as $f_j : \mathcal{A} \times \Phi \mapsto \{0, 1\}$. The JA function f satisfies independence if and only if for all $\pi \in \Pi$, there exist functions f_i such that for all $\varphi \in f(\pi)$ it holds that $\varphi = \{a \in \mathcal{A}, f_j(a, \pi \Downarrow a) = 1\} \cup \{\neg a \mid a \in \mathcal{A}, f_j(a, \pi \Downarrow a) = 0\}$.

The best known example of $f_j : \mathcal{A} \times \Phi \mapsto \{0, 1\}$ is the simple majority voting f_m which counts how many agents expressed judgment “yes” on agenda item a . The function f_m returns a if that number of agents is greater than $\lceil \frac{n}{2} \rceil$ and $\neg a$ otherwise.

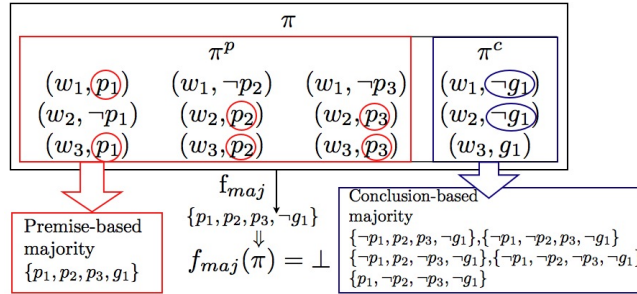


Fig. 1. A profile, issue-wise majority aggregation, premise-based and conclusion-based majority.

The issue-wise majority function f_{maj} is defined as

$$f_{maj} = \left\{ \begin{array}{l} \varphi = \{f_m(\pi \Downarrow a) \mid a \in \mathcal{A}\} \text{ if } \varphi \text{ is complete and consistent} \\ \perp \text{ otherwise} \end{array} \right\}$$

The JA function f_{maj} satisfies *universal domain*, *anonymity*, *unanimity*, *completeness*, and *independence* but it does not satisfy *collective rationality*, as it can be seen on Figure 1. All judgement aggregation functions which satisfy *universal domain*, *anonymity*, *independence* and *collective rationality* are *constant* [17]. The most debated [3, 17] is *independence*. The reason why it is convenient to have independence is because it is a necessary condition to guarantee the non-manipulability of f [8]. An aggregation function is non-manipulable, if no agent can obtain his sincere judgment set φ selected as the collective judgment set by submitting another judgment set φ' .

Two aggregation procedures that violate *independence* but guarantee *universal domain*, *anonymity* and *collective rationality* have been proposed in the literature: *premise-based* and *conclusion-based* procedures. A distinguishing feature of judgment aggregation with respect to social choice theory is the distinction between premises and conclusions. The agenda is a union of two disjoint sets: the premise set (\mathcal{A}^p), and the conclusion set (\mathcal{A}^c). $\mathcal{A} = \mathcal{A}^p \cup \mathcal{A}^c$, $\mathcal{A}^p \cap \mathcal{A}^c = \emptyset$. In Example 1, $\mathcal{A}^p = \{p_1, p_2, p_3\}$ and $\mathcal{A}^c = \{g_2\}$. We give a general definition on when a JA function is premise- or conclusion-based in Definition 4. The literature on judgment aggregation [17] defines premise- and conclusion-based procedures only in terms of issue-wise majority.

Definition 4. Let $\pi^p = \{(i, a) \mid (i, a) \in \pi, a \in \mathcal{A}^p\}$, $\pi^c = \{(i, a) \mid (i, a) \in \pi, a \in \mathcal{A}^c\}$ and let the *premise and conclusion aggregation functions* be defined as $f^p : \Pi \mapsto 2^{\mathcal{A}^p} / \emptyset$ and $f^c : \Pi \mapsto 2^{\mathcal{A}^c} / \emptyset$ correspondingly. The JA function f is *premise-based* if and only if there exists a f^p such that $f^p(\pi^p) \subseteq f(\pi)$ and *conclusion-based* if and only if there exists a f^c such that $f^c(\pi^c) \subseteq f(\pi)$, for all $\pi \in \Pi$.

An example of premise and conclusion-based aggregations is given in Figure 1. Intuitively, a JA function is premise-based when the collective judgment set on the premises is obtained as a result of aggregating only the premise profile π^p . The decisions $f(\pi)$ are those $\varphi \in \mathcal{W}$ for which $f^p(\pi^p) \subset \varphi$ (or alternatively $f^c(\pi^c) \subset \varphi$ for conclusion-based aggregations). However, it is possible that, depending on the decision rule, there are more than one $\varphi \in \mathcal{W}$ that satisfy the condition $f^p(\pi^p) \subset \varphi$ (or alternatively $f^c(\pi^c) \subset \varphi$). This is what happens for the conclusion-based procedure we illustrate in Figure 1.

2.2 Judgment aggregation for BDI agents

In this section first we set the problem of finding collective decisions for group goals in the context of judgment aggregation. We then argue that the aggregation function used for this problem should satisfy: *collective rationality*, *universal domain*, *unanimity* and select a unique $\varphi \in \mathcal{W}$. For a democratic group, in which all agents have equal say on what the group attitudes should be, *anonymity* should be satisfied. For a group in which the agents have different levels of expertise, *anonymity* can be omitted.

We use Gg to denote that “ g is a group goal”. A set of the candidate group goals is the set $\mathcal{G} = \{Gg \mid g \in \mathcal{L}\}$ which contains all the goals which the group considers to adopt. For each goal $Gg \in \mathcal{G}$, the group has at its disposal decision rules $\mathcal{R}_g \subseteq \mathcal{R}$ and an agenda \mathcal{A}_g composed of the goal g (conclusion), and all the reasons (premises) relevant for g , which were given by the decision rule. Each agent submits her judgments on the agenda thus generating a profile π_g , such that $con(\mathcal{R}_g, \pi_g \Rightarrow i) = 1$ for all $i \in \mathcal{N}$.

The decision rules contain all the constraints which the individual and collective judgment sets should satisfy. These constraints contain three types of information: rules describing how the goal depends on the reasons (justification rules \mathcal{R}_g^{just}), rules describing the constraints of the world inhabited by the agents (domain knowledge \mathcal{R}_g^{DK}) and rules that describe how g interacts with other candidate goals of the group (coordination rules \mathcal{R}_g^{coord}). Hence, the decision rule for a group goal g is $\mathcal{R}_g = \mathcal{R}_g^{just} \cup \mathcal{R}_g^{DK} \cup \mathcal{R}_g^{coord}$.

Example 2 (Example 1 revisited). Consider the cleaning crew from Example 1. $\mathcal{R}_{g_1}^{just}$ is $(p_1 \wedge (p_2 \vee p_3)) \leftrightarrow Gg_1$ and $\mathcal{A}_{g_1} = \{p_1, p_2, p_3, Gg_1\}$. Suppose that the crew has the following candidate group goals as well: *place the furniture in its designated location* (g_2) and *collect recyclables from garbage bin* (g_3). The agendas are $\mathcal{A}_{g_2} = \{p_4, p_5, p_6, p_7, Gg_2\}$, $\mathcal{A}_{g_3} = \{p_3, p_8, p_9, Gg_3\}$. The justification rules are $\mathcal{R}_{g_2}^{just} \equiv (p_4 \wedge p_5 \wedge (p_6 \vee p_7)) \leftrightarrow Gg_2$ and $\mathcal{R}_{g_3}^{just} \equiv (p_8 \wedge p_9 \wedge p_3) \leftrightarrow Gg_3$. The formulas $p_4 - p_9$ are: the furniture is out of place (p_4), the designated location for the furniture is empty (p_5), the furniture has wheels (p_6), the furniture has handles (p_7), the agents can get revenue for recyclables (p_8), there is a container for the recyclables (p_9). An example of a domain knowledge could be $\mathcal{R}_{g_2}^{DK} \equiv \neg p_4 \rightarrow \neg p_5$, since it can not happen that the designated location for the furniture is empty while the furniture is not out of place. Group goal Gg_3 can be pursued at the same time as Gg_1 , however, Gg_2 can only be pursued alone. Thus the coordination rule for all three goals is $\mathcal{R}_{g_1}^{coord} = \mathcal{R}_{g_2}^{coord} = \mathcal{R}_{g_3}^{coord} \equiv ((Gg_2 \wedge \neg(Gg_1 \vee Gg_3)) \vee \neg Gg_2)$.

We want the justifications for a goal to explain when a goal should be adopted/refuted. Having collective justifications for a goal enables the agents to, when the world changes, consider adopting a goal that has been rejected previously. To this end, we take into consideration justification rules which are of form $Gg \leftrightarrow \Gamma$, where $\Gamma \in \mathcal{L}$ such that all the non-logical symbols of Γ occur in \mathcal{A}_g^p and $\{Gg\} = \mathcal{A}_g^c$.

We now continue to discuss the desirable properties for the aggregation of individual beliefs and goals. We need a JA function that satisfies *universal domain* to be able to aggregate all admissible profiles. We can use only JA functions that satisfy *collective rationality*. If the collective set is not complete, for example, if it contains only a collective judgment on the goal, then we do not know why the goal was (not) adopted and consequently we would not know when to revise it. For example, the cleaning crew decides for the goal g_3 (to collect recyclables), without having the reasons like p_9 (a container where to put them). If the world changes and $\neg p_9$ holds, the robots will continue to collect recyclables.

The aggregation of all admissible profiles should produce a set consistent with the decision rule because otherwise we would not be generating the group goals and justifications for them. For the same reason we need an aggregation method that selects a unique $\varphi \in \mathcal{W}$.

To guarantee that $con(\mathcal{R}_g, f(\pi_g)) = 1$, we need to choose between conclusion-based and premise-based procedures. At first glance, the premise-based procedure seems an obvious choice since it will produce complete collective judgment sets under our decision rules. However, upon closer inspection, this procedure has notable drawbacks.

As it is observable from the profile in Figure 1; a premise-based procedure may lead

the group to adopt a conclusion that the majority (or even the unanimity) of the group does not endorse. In our case, the conclusion is the goal and a premise-based aggregation may establish a group goal which none of the agents is interested in pursuing. To avoid this we need to aggregate using a conclusion-based procedure. In particular we want a conclusion-based procedure that has the property of unanimity on Gg .

Given that our decision rules are of the form $g \leftrightarrow T$, there exist profiles for which a conclusion-based procedure will not produce complete collective set of judgments. However, the conclusion-based aggregation can be supplemented with an additional procedure that completes the produced set of judgments when necessary. Such aggregation procedure is the complete conclusion-based procedure (CCBP) developed in [20]. This CCBP satisfies *universal domain* and is *collectively rational*. However, it does not always produce a unique decision. CCBP produces a unique collective judgment for the goal, but it can generate more than one set of justifications for it. This is an undesirable, but not an unmanageable property. To deal with ties, as it is the practice with voting procedures, the group can either determine a default set of justifications for adopting/rejecting each candidate goal, or it can appoint one member of the group as tiebreaker. Tie-breaking problems in judgment aggregation are the focus of our ongoing research.

The CCBP from [20] also satisfies *anonymity*. Whether this is a desirable property for a group of artificial agents depends entirely on whether the group is democratic or the opinions of some agents are more important. CCBP can be adjusted to take into consideration different weights on different agents' judgment sets.

3 Formal framework for modeling group attitudes

In this section we introduce the language of modal multi-agent logic in which we represent individual and collective mental attitudes. We then combine the methodology of judgment aggregation with this representation language and show how collective attitudes are generated. To model commitment to a group goal and reconsideration of a group goal we use temporal logic.

3.1 Modal multi-agent logic

Just like modal agent logic is concerned mainly with the relation between the individual goals and beliefs over time, modal multi-agent logic is concerned with the relation between group goals and beliefs over time. We use modal multi-agent logic to represent: the agenda, individual judgments, collective judgments and new information that may prompt goal revision. In line with judgment aggregation proper, we do not use the formal language to represent the judgment aggregation function, but only the arguments of this function and the results from it. We assume that there is a service, available to the agents, that elicits the individual judgments, performs the aggregation and makes the aggregation results available, to the members and for plan-generation.

Agenda issues in judgment aggregation are usually represented by propositional formulas. This is not a viable option in our case. First, we want to represent the difference between a goal and the supporting reasons by means of representing the distinction between conclusions and premises explicitly in the logic. Second, the logic should represent the distinction between individual and collective judgments. We distinguish

conclusions from premises by using a single K modal operator Gg for representing that “ g is a group goal”.

The obvious choice for modeling the judgment “true” on agenda issue a , of an agent i , is the modal operator belief $B_i a$ (correspondingly $B_i \neg a$ for a judgment “false” on a). However, we find that beliefs are ill suited for modeling collective judgments of agents. While a belief $B_i a$ is an exclusively private mental state, judgments are public and contributed for the decision-making purposes of the group. A judgment is thus closer to a public commitment than to a private belief. Hence, we model judgments by using the modal operator of acceptance $A_S a$ [19]. $A_S a$ denotes: agents in S accept a . The operator $A_S a$ allows us to represent both individual judgments, $S = \{i\}$, for $i \in \mathcal{N}$ and collective judgments with $S = \mathcal{N}$. We define the group acceptance $A_{\mathcal{N}} a$ to be the result of applying judgment aggregation to the individual acceptances. We present the formal definitions of profile and judgment aggregation function in our logic in Definition 6. The modal operator A_S we use is inspired by the modal operator of the *acceptance logic* of [19]. The details on the relation between acceptance logic and our acceptance operator are given in the Appendix.

We represent the new information that becomes available to the agents with a normal modal K operator E . Ea denotes: “it is observed that a is true”.

Lastly, to model how the group attitudes evolve with reconsideration we need a temporal logic. By using LTL we do not need to distinguish between path formulas and state formulas, but we are able to quantify over traces. Just as in BDI_{LTL} , where $B\Box a$ denotes that a is believed to be necessary, we use $E\Box \neg a$ to denote that a is observed to be impossible.

We now give the syntax of our modal multi-agent logic AGE_{LTL} .

Definition 5 (Syntax). *Let Agt be a non-empty set of agents, with $S \subseteq Agt$, and L_P be a set of atomic propositions. The admissible formulae of AGE_{LTL} are formulae ψ_0, ψ_1 and ψ_2 of languages \mathcal{L}_{prop} , \mathcal{L}_G and $\mathcal{L}_{AE_{LTL}}$ correspondingly:*

$$\psi_0 ::= p \mid (\psi_0 \wedge \psi_0) \mid \neg \psi_0$$

$$\psi_1 ::= \psi_0 \mid G\psi_0$$

$$\psi_2 ::= \psi_0 \mid A_S \psi_1 \mid E\psi_2 \mid X\psi_2 \mid (\psi_2 U \psi_2)$$

where p ranges over L_P and S over 2^{Agt} . Moreover, $\Diamond\phi \equiv \top U \phi$, $\Box\phi \equiv \neg \Diamond \neg \phi$, and $\phi R \phi' \equiv \neg(\neg \phi U \neg \phi')$.

We can now adjust the definition for a judgment aggregation function. We represent individual judgments with $A_{\{i\}} a$ with intuitive reading “agent i judges a as true” and $A_{\{i\}} \neg a$ with reading “agent i judges a as false”.

Definition 6 (JA in AGE_{LTL}). *Let $\mathcal{N} = \{1, 2, \dots\}$ be a set of agent names and $\mathcal{G} \subseteq \mathcal{L}_G / \mathcal{L}_{prop}$ a set of candidate goals.*

An agenda $\mathcal{A}_g \subseteq \mathcal{L}_G$ for goal $Gg \in \mathcal{G}$ is a set of formulas such that $\mathcal{A}_g = \overline{\mathcal{A}}_g^p \cup \overline{\mathcal{A}}_g^c$,

with $\overline{\mathcal{A}}_g^p = \mathcal{A}_g^p \cup \{\neg a \mid a \in \mathcal{A}_g^p\}$, $\mathcal{A}_g^p \subseteq \mathcal{L}_{prop}$ and $\overline{\mathcal{A}}_g^c = \{Gg, \neg Gg\}$.

A profile of judgments is the set $\overline{\pi} = \{A_{\{i\}} a \mid i \in \mathcal{N}, a \in \mathcal{A}_g\}$.

$\overline{\pi} \Rightarrow i = \{a \mid A_{\{i\}} a \in \overline{\pi}\}$ is the judgment set of agent i .

$\overline{\pi} \Downarrow a = \{i \mid A_{\{i\}} a \in \overline{\pi}\}$ is the set of all the agents that accepted a .

Given a set of decision rules $\mathcal{R}_g \subset \mathcal{L}_{prop}$, a profile is acceptable if and only if, for all $i \in \mathcal{N}$ and all $a \in \mathcal{A}$, $con(\mathcal{R}_g, \overline{\pi} \Rightarrow i) = 1$ and either $A_{\{i\}} a$ or $A_{\{i\}} \neg a$. The set of all

acceptable profiles for \mathcal{N} and \mathcal{A}_g , given \mathcal{R}_g is $\overline{\Pi}$.

The decision for a profile $\overline{\pi}$, $D_{\overline{\pi}} = \{A_{\mathcal{N}}a \mid a \in f^a(\overline{\pi}), f^a : \overline{\Pi} \mapsto 2^{\mathcal{A}_g}\}$.

For instance, the profile in Figure 1, is $\overline{\pi}_{g_1} = \{A_{\{w_1\}}p_1, \neg A_{\{w_1\}}p_2, \neg A_{\{w_1\}}p_3, A_{\{w_1\}}\neg Gg_1, A_{\{w_2\}}\neg p_1, \dots, A_{\{w_3\}}Gg_1\}$. The decision using premise-based majority would be $\{A_{CP_1}, A_{CP_2}, A_{CP_3}, A_{CG_1}\}$.

AGE_{LTL} can be used to model that an agent does not have a judgment on an agenda issue, but we will work with the assumption that either $A_{\{i\}}$ or $A_{\{i\}}\neg a$ for all agents and agenda issues.

AGE_{LTL} has Kripke semantics. As in Schild [22], a Kripke structure is defined as a tuple $\mathcal{M} = \langle W, \mathcal{R}, \mathcal{G}, \mathcal{E}, \mathcal{A}, L \rangle$. W is a set of possible situations, and \mathcal{R} is the temporal relation over situations $\mathcal{R} \subseteq W \times W$. \mathcal{G} is the goal relation over situations $\mathcal{G} \subseteq W \times W$, while \mathcal{E} is the observation relation over situations $\mathcal{E} \subseteq W \times W$. Let $\Delta = 2^{\mathcal{N}} \times Inst$. $\mathcal{A} : \Delta \mapsto W \times W$ maps every $S \in \Delta$ to a relation \mathcal{A}_S between possible situations. L is a truth assignment to the primitive propositions of L_P for each situation $w \in W$, i.e., $L(w) : Prop \mapsto \{true, false\}$.

Temporal formulas are validated in the standard manner [15]. Normal modal formulas $G\psi$ and $E\psi$ have standard semantics, see for example [4]. Acceptance formulas $A_S\psi$ are validated according to the semantics of acceptance logic presented in [19]. The axiomatization of AGE_{LTL} is given in the Appendix. Note that AGE_{LTL} is a fusion of the decidable logics: LTL , acceptance logics and two K modal logics.

3.2 Generation of group goals

The mental state of the group is determined by the mental states of the members and the choice of judgment aggregation function. We represent the mental state of the group by a set χ of AGE_{LTL} formulas. The set χ contains: the set of all candidate goals for the group $\mathcal{G} \subseteq \mathcal{L}_G/\mathcal{L}_{prop}$ and, for each $Gg \in \mathcal{G}$, the corresponding decision rules \mathcal{R}_g , as well as the individual and collective acceptances made in the group regarding agenda \mathcal{A}_g . The set χ is common knowledge for the group members. An agent uses χ when it acts as a group member and its own beliefs and goals when it acts as an individual.

To deal with multiple, possibly mutually inconsistent goals, the group has a priority order \succeq_x over the group goals $\mathcal{G} \subset \chi$. To avoid overburdening the language with a \succeq_x operator, we incorporate the priority order within the decision rules $\mathcal{R}_{g_i}^{just} \equiv \Gamma_i \leftrightarrow Gg_i$. We want the decision rules to capture that if Gg_i is not consistent (according to the coordination rules) with some higher priority goals Gg_1, \dots, Gg_m , then the group can accept Gg_i if and only if none of Gg_1, \dots, Gg_m is accepted. Hence, we replace the justification rule $\mathcal{R}_{g_i}^{just} \in \chi$ with $\mathcal{R}_{g_i}^{just} \equiv (\Gamma_i \wedge \bigwedge_j^m (A_{\mathcal{N}}\neg Gg_j)) \leftrightarrow Gg_i$, where $Gg_j \in \mathcal{G}$, $Gg_j \succeq_x Gg_i$ and $Gg_i \wedge Gg_j \wedge \mathcal{R}_{g_i}^{coord} \models \perp$.

Example 3. Consider the goals and rules of the robot crew C from Example 2. Assume the crew has been given the priority order $Gg_1 \succ_x Gg_2 \succ_x Gg_3$. χ contains: $\mathcal{G} = \{Gg_1, Gg_2, Gg_3\}$, one background knowledge rule, one coordination rule, three justification rules, out of which two are new priority modified rules:

$\{G, \neg p_4 \rightarrow \neg p_5, (Gg_2 \wedge \neg(Gg_1 \vee Gg_3)) \vee \neg Gg_2, Gg_1 \leftrightarrow (p_1 \wedge (p_2 \vee p_3)), Gg_2 \leftrightarrow (p_4 \wedge p_5 \wedge (p_6 \vee p_7) \wedge A_C\neg Gg_1), Gg_3 \leftrightarrow (p_8 \wedge p_9 \wedge p_3 \wedge (A_C\neg Gg_2))\}$.

The agents give their judgments on one agenda after another starting with the agenda for the highest priority candidate goal. Once the profile $\bar{\pi}$ and the decision $\mathcal{D}_{\bar{\pi}}$ for a goal g are obtained, they are added to χ . To avoid the situation in which the group casts judgments on an issue that has already been decided, we need to remove decided issues from \mathcal{A}_g before eliciting the profile for this agenda.

The group goals are generated by executing **GenerateGoals**(χ, \mathcal{N}).

```
function GenerateGoals( $\chi, S$ ) :
for each  $Gg_i \in \mathcal{G}$  s.t.  $[\forall Gg_j \in \mathcal{G}: (Gg_j \succ Gg_i) \Rightarrow (A_{\mathcal{N}}Gg_j \in \chi \text{ or } A_{C:\chi} \neg Gg_j \in \chi)]$ 
  {  $B := (\{a \mid A_{\mathcal{N}}a \in \chi\} \cup \{\neg a \mid A_{\mathcal{N}}\neg a \in \chi\}) \cap \mathcal{A}_{g_i}$ ;
    comment //  $B$  is the set of already collectively accepted issues from  $\mathcal{A}_{g_i}$ 
     $\mathcal{A}_{g_i}^* := \mathcal{A}_{g_i}/B$ ;
     $\bar{\pi}_{g_i} := \text{elicit}(S, \mathcal{A}_{g_i}^*, \chi)$ ;
     $\chi := \chi \cup \bar{\pi}_{g_i} \cup f^a(\bar{\pi}_{g_i});$  }
return  $\chi$ .
```

elicit requests the agents to submit complete judgment sets for $\bar{\pi}_{g_i} \subset \chi$. We require that *elicit* is such that for all returned $\bar{\pi}$ it holds $\text{con}(\chi, \bar{\pi} \Rightarrow i) = 1$ for all $i \in \mathcal{N}$ and that $\text{con}(\chi, f^a(\bar{\pi})) = 1$. When a higher priority goal Gg_i is accepted by the group, a lower priority incompatible goal Gg_j cannot be adopted regardless of the judgments on the issues in \mathcal{A}_{g_j} . Nevertheless, Although *elicit* will provide individual judgments for the agenda \mathcal{A}_{g_j} . If the acceptance of Gg_i is reconsidered, we can obtain a new decision on Gg_j because the profile for Gg_j is available.

Example 4. Consider the χ for robots given in Example 3. The following calls to *elicit* are made in the given order. First, $\bar{\pi}_{g_1} = \text{elicit}(\mathcal{N}, \mathcal{A}_{g_1}^*, \chi)$ with the $\text{GenerateGoals}(\chi) = \chi' = \chi \cup \bar{\pi}_{g_1} \cup f^a(\bar{\pi}_{g_1})$. Second, $\bar{\pi}_{g_2} = \text{elicit}(\mathcal{N}, \mathcal{A}_{g_2}^*, \chi')$, with $\text{GenerateGoals}(\chi') = \chi'' = \chi' \cup \bar{\pi}_{g_2} \cup f^a(\bar{\pi}_{g_2})$. Last, $\bar{\pi}_{g_3} = \text{elicit}(\mathcal{N}, \mathcal{A}_{g_3}^*, \chi'')$, with $\text{GenerateGoals}(\chi'') = \chi''' = \chi'' \cup \bar{\pi}_{g_3} \cup f^a(\bar{\pi}_{g_3})$. Since there is no overlapping between agendas \mathcal{A}_{g_2} and \mathcal{A}_{g_1} , $\mathcal{A}_{g_1}^* \equiv \mathcal{A}_{g_1}$ and $\mathcal{A}_{g_2}^* \equiv \mathcal{A}_{g_2}$. However, since $\mathcal{A}_{g_2} \cap \mathcal{A}_{g_3} = p_3$, then $\mathcal{A}_{g_3}^* = \{p_8, p_9, Gg_3\}$.

Proposition 1. *GenerateGoals terminates if and only if elicit terminates and does not violate the candidate goal preference order.*

The proof is straightforward.

4 Commitment strategies

The group can choose to reconsider the decision (acceptance or rejection) on a group goal in presence of new information. Whether the group chooses to reconsider depends on how committed it is to bring the goal about. According to Cohen and Levesque [5], an agent intends g if it has chosen to pursue goal g and it committed itself to making g happen. Following the same line of reasoning, we define group intention to be $I_{\mathcal{N}}g \equiv A_{\mathcal{N}}g$ and read it as “the agents \mathcal{N} intend g ”. We defined collective acceptance as resulting from judgment aggregation, which is a social choice method. Thus, in our framework, group intention is social choice with commitment. The level of commitment of a group to a collective acceptance depends on the choice of commitment strategy.

These are the three main commitment strategies (introduced by Rao and Georgeff [21]):

Blind commitment: $I_i g \rightarrow (I_i g \mathbf{U} B_i g)$

Single-minded commitment: $I_i g \rightarrow (I_i g \mathbf{U} (B_i g \vee B_i \Box \neg g))$

Open-minded commitment: $I_i g \rightarrow (I_i g \mathbf{U} (B_i g \vee \neg G_i g))$

These commitment strategies only consider the relation between beliefs regarding g and Gg . Instead, a commitment to a goal can now be reconsidered upon new information on either one of the agenda issues in \mathcal{A}_g and also upon new information on a higher priority goal.

The strength of our framework is exhibited in its ability to describe the groups' commitment not only to its decision to adopt a goal, but also to its decision to reject a goal. Namely, if the agents decided $I_{\mathcal{N}} g_i$ and $A_{\mathcal{N}} \neg g_j$ are committed to both $I_{\mathcal{N}} g_i$ and $A_{\mathcal{N}} \neg g_j$. Commitment to reject g allows for g to be reconsidered and eventually adopted if the state of the world changes.

Let \mathcal{N} be a set of agents with a set of candidate goals \mathcal{G} . Let $Gg_i, Gg_j \in \mathcal{G}$ have agendas $\mathcal{A}_{g_i}, \mathcal{A}_{g_j}$. We use $p \in \mathcal{A}_{g_i}^p$ and $q_i \in \mathcal{A}_{g_i}^c, q_j \in \mathcal{A}_{g_j}^c$. The profiles and decisions are $\bar{\pi}_{g_i}$ and $f^a(\bar{\pi}_{g_i})$; $Gg_j > Gg_i$, and Gg_j cannot be pursued at the same time as Gg_i .

We use the formulas $(\alpha_1) - (\alpha_7)$ to refine the blind, single-minded and open-minded commitment. Instead of the "until", we use the temporal operator *release*: $\psi \mathbf{R} \phi \equiv \neg(\neg\psi \mathbf{U} \neg\phi)$, meaning that ϕ has to be true until and including the point where ψ first becomes true; if ψ never becomes true, ϕ must remain true forever. Unlike the *until* operator, the *release* operator does not guarantee that right hand-side formula will ever become true, which in our case translates to the fact that an agent could be forever committed to a goal.

$(\alpha_1) E g_i \mathbf{R} I_{\mathcal{N}} g_i$

$(\alpha_2) \perp \mathbf{R} A_{\mathcal{N}} \neg G g_i$

$(\alpha_3) (E \Box \neg g_i \vee E g_i) \mathbf{R} A_{\mathcal{N}} q_i$

$(\alpha_4) A_{\mathcal{N}} \neg q_j \mathbf{R} A_{\mathcal{N}} q_i$

$(\alpha_5) A_{\mathcal{N}} p \rightarrow (E \neg p \mathbf{R} A_{\mathcal{N}} q_i)$

Blind commitment: $\alpha_1 \wedge \alpha_2$.

Only the observation that the goal is achieved ($E g_i$) can release the intention to achieve the goal $I_{\mathcal{N}} g_i$. If the goal is never achieved, the group will always be committed to it. If a goal is not accepted, then the agents will not reconsider accepting it.

Single-minded commitment: α_3 .

Only new information on the goal (either that the goal is achieved or had become impossible) can release the decision of the group to adopt /reject the goal. Hence, new information is only regarded if it concerns the conclusion, while information on the remaining agenda items is ignored.

Extended single-minded commitment: $\alpha_3 \wedge \alpha_4$.

Not only new information on g_i , but also the collective acceptance to adopt a more important incompatible goal g_j can release the intention of the group to achieve g_i . Similarly, if g_i is not accepted, the non-acceptance can be revised, not only if g_j is observed to be impossible or achieved, but also when the commitment to pursue g_j is dropped (for whatever reason).

Open-minded commitment: $\alpha_3 \wedge \alpha_5$.

A group will maintain its collective acceptances to adopt/reject a goal as long as the new information regarding all collectively accepted agenda items is consistent with $f^a(\bar{\pi}_{g_i})$.

Extended open-minded commitment: $\alpha_3 \wedge \alpha_4 \wedge \alpha_5$.

Extending on the single-minded commitment, a change in intention to pursue a higher priority goal Gg_j can also release the acceptance of the group on Gg_i .

Once an intention is dropped, a group may need to reconsider its collective acceptances. This may cause for the dropped goal to be re-affirmed, but a reconsideration process will be invoked nevertheless.

5 Reconsideration strategies

In Section 3.2 we defined the mental state of the group χ . We can now define what it means for a group to be *coherent*.

Definition 7 (Group coherence). *Given a Kripke structure \mathcal{M} and situations $s \in W$, a group of \mathcal{N} agents is coherent if the following conditions are met:*

(ρ_1): $\mathcal{M} \models \neg(A_S a \wedge A_S \neg a)$ for any $S \subseteq \mathcal{N}$ and any $a \in \mathcal{A}_g$.

(ρ_2): If $\mathcal{M}, s \models \chi$ then $\chi \not\models \perp$.

(ρ_3): $\mathcal{M}, s \models \bigwedge \mathcal{G} \rightarrow \neg \Box \neg g$ for all $Gg \in \mathcal{G}$.

(ρ_4): Let $Gg \in \mathcal{G}$ and $\mathcal{G}' = \mathcal{G} / \{Gg\}$, then $\mathcal{M} \models (\bigwedge \mathcal{G}' \wedge E \Box \neg g) \rightarrow \mathbf{X}(\neg Gg)$.

(ρ_5): Let $p \in \mathcal{A}_g^p$ and $q \in \{Gg, \neg Gg\}$. $Ep \wedge (Ep \mathbf{R} A_{\mathcal{N}} q) \rightarrow \mathbf{X} A_{\mathcal{N}} p$

The first condition ensures that no contradictory judgments are given. The second condition ensures that the mental state of the group is logically consistent in all situations. The third and fourth conditions ensure that impossible goals cannot be part of the set of candidate goals and if g is observed to be impossible in situation s , then it will be removed from \mathcal{G} in the next situation. ρ_5 enforces the acceptance of the new information on the group level, when the commitment strategy so allows – after a is observed and that lead the group to de-commit from g , the group necessarily accepts a .

A coherent group accepts the observed new information on a premise. This may cause the collective acceptances to be inconsistent with the justification rules. Consequently, the decisions and/or the profiles in χ need to be changed in to ensure that ρ_1 and ρ_2 are satisfied. If, however $\Box \neg g$ or g is observed, the group reconsiders χ by removing Gg from \mathcal{G} . In this case, the decisions and profiles are not changed.

For simplicity, at present we work with a world in which the agents' knowledge can only increase, namely the observed information is not a fluent. A few more conditions need to be added to the definition of group coherence, for our model to be able to be applicable to fluents. E.g., we need to define which observation is accepted when two subsequent contradictory observations happen.

For the group to be coherent at all situations, the acceptances regarding the group goals need to be reconsidered after de-commitment. Let $\mathcal{D}_g \subset \chi$ contain the group acceptances for a goal g , while $\bar{\pi}_g \subset \chi$ contain the profile for g . There are two basic ways in which a collective judgment set can be reconsidered. The first way is to elicit a new profile for g and apply judgment aggregation to it to obtain the reconsidered \mathcal{D}_g^* . The second is to reconsider only \mathcal{D}_g without re-eliciting individual judgments. The first approach requires communication among agents. The second approach can be done

by each agent reconsidering χ by herself. We identify three reconsideration strategies available to the agents. The strategies are ordered from the least to the most demanding in terms of agent communication.

Decision reconsideration (\mathcal{D} -r). Assume that $Ep, p \in \mathcal{A}_g^p, q \in \{Gg, \neg Gg\}$ and the group de-committed from $A_{\mathcal{N}}q$. The reconsidered decision \mathcal{D}_g^* is such that p is accepted, i.e., $A_{\mathcal{N}}p \in \mathcal{D}_g^*$, and the entire decision is consistent with the justification rules, namely $con(\mathcal{R}_g^{pjust}, \mathcal{D}_g^* \Rightarrow \mathcal{N}) = 1$. If the \mathcal{D} -r specifies an unique \mathcal{D}_g^* , for any observed information and any \mathcal{D}_g , then χ can be reconsidered without any communication among the agents. Given the form of \mathcal{R}_g^{pjust} (see Section 3.2), this will always be the case.

However \mathcal{D} -r is not always an option when the de-commitment occurred due to a change in collective acceptance of a higher priority goal g' . Let $q' \in \{Gg', \neg Gg'\}$. Let the new acceptance be $A_{\mathcal{N}}\neg q'$. \mathcal{D} -r is possible if and only if $\mathcal{D}_g^* = \mathcal{D}_g$ and $con(\mathcal{R}_g^{pjust}, \mathcal{D}_g \cup \{A_{\mathcal{N}}\neg q'\}) = 1$. Recall that $A_{\mathcal{N}}q'$ was not in \mathcal{A}_g and as such the acceptance of q' or $\neg q'$ is never in the decision for $\bar{\pi}_g$.

Partial reconsideration of the profile (Partial $\bar{\pi}$ -r). Assume that $Ea, a \in \mathcal{A}_g, Gg \in \mathcal{G}$. Not only the group, but also the individual agents need to accept a . The *Partial $\bar{\pi}$ -r* asks for new individual judgments be elicited. This is done to ensure the logical consistency of the individual judgment sets with the observations. New judgments are only elicited from the agents i which $A_{\{i\}}\neg a$.

Let $W \subseteq \mathcal{N}$ be the subset of agents i s.t. $A_{\{i\}}\neg a \in \chi$. Agents i are s.t. $A_{\{i\}}a \in \chi$ when the observation is $E\neg a$. Let $\bar{\pi}_g^W \subseteq \bar{\pi}_g$ be the set of all acceptances made by agents in W . We construct $\chi' = \chi / \bar{\pi}_g^W$. The new profile and decision are obtained by executing *GenerateGoals* (χ', W).

Example 5. Consider Example 4. For $\bar{\pi}_{g_1}, \bar{\pi}_{g_2}$ and $\bar{\pi}_{g_3}$ of the robot crew C , the decisions $\mathcal{D}_{g_1} = \{A_{Cp_1}, A_{C\neg p_2}, A_{Cp_3}, A_{CGg_1}\}, \mathcal{D}_{g_2} = \{A_{Cp_4}, A_{Cp_5}, A_{Cp_6}, A_{Cp_7}, A_{C\neg Gg_2}\}$ and $\mathcal{D}_{g_3} = \{A_{Cp_8}, A_{Cp_9}, A_{CGg_3}\}$ are made. Assume the group de-commits on Gg_1 because of $E\neg p_2$. If the group is committed to Gg_3 , the commitment on Gg_3 will not allow for $A_{\mathcal{N}}p_3$ to be modified when reconsidering Gg_1 . Since $A_{\mathcal{N}}p_3$ exists in χ' , p_3 will be excluded from the (new) agenda for g_1 , although it was originally in it. *elicit* calls only on the agents in W to complete $\bar{\pi}_{g_1} \in \chi'$ with their judgment sets.

Full profile reconsideration ($\bar{\pi}$ -r). The full profile reconsideration is the same with the partial reconsiderations in all respects except one – now $W = \mathcal{N}$. Namely, within the full profile revision strategy, each agent is asked to revise his judgment set by accepting the new information, regardless whether he had already accepted the information or not.

5.1 Combining revision and commitment strategies

Unlike the Rao and Georgeff commitment strategies [21], in our framework the commitment strategies are not axioms of the logic. We require that the commitment strategy is valid in all the models of the group and not in all the models of AGE_{LTL} . This allows the group to define different commitment strategies and different revision strategies for different goals. It might even choose to revise differently depending on which information triggered the revision. Choosing different revision strategies for each goal, or each

type of new information, should not undermine the coherence of the group record χ . The conditions of group coherence of the group ensures that after every reconsideration χ must remain consistent. However, some combinations of commitment strategies can lead to incoherence of χ .

Example 6. Consider the profiles and decisions in Example 5. Assume that initially the group chose open-minded commitment for $I_C g_1$ and blind commitment for $I_C g_3$, with goal open-minded commitment for $A_C \neg G g_2$. If $E g_1$ and thus $I_C g_1$ is dropped, then the extended open-minded commitment would allow $A_C \neg G g_2$ to be reconsidered and eventually $I_C g_2$ established. However, since the group is blindly committed to $I_C g_3$, this change will not cause reconsideration and as a result both $I_C g_2$ and $I_C g_3$ will be in χ thus making χ incoherent.

Problems arise when $sub(\mathcal{R}_{g_i}^{pjust}) \cap sub(\mathcal{R}_{g_j}^{pjust}) \neq \emptyset$, where $sub(\mathcal{R}_g^{pjust})$ denotes the set of atomic sub-formulas of g ($G g_i, G g_j \in \mathcal{G}$). Proposition 2 summarizes under which conditions these problems are avoided.

Proposition 2. *Let α' and α'' be the commitment strategies selected for g_i and g_j correspondingly. $\chi \cup \alpha' \cup \alpha'' \neg \neq \perp$ (in all situations):*

- a) *if $\phi \in sub(\mathcal{R}_{g_i}^{pjust}) \cap sub(\mathcal{R}_{g_j}^{pjust})$ and $p \in \mathcal{A}_{g_i} \cap \mathcal{A}_{g_j}$, then α_5 is either in both α' and α'' or in none;*
- b) *if $G g_i$ is more important than $G g_j$ and G_j and G_i cannot be accepted at the same time, then $\alpha_4 \in \alpha''$.*

Proof. The proof is straightforward. Namely, if the change on acceptance of $G g_i$ causes the decision on $G g_j$ to induce group incoherence, we are able to de-comit from $G g_j$. If we were not able to de-comit on $G g_j$ group coherence is blocked. If the change on collective acceptance on $G g_i$ is caused by an observation on a premise $p \in \mathcal{A}_{g_i} \cap \mathcal{A}_{g_j}$ then condition a) ensures that the commitment to the collective acceptance regarding $G g_j$ does not block group coherence. If the change on collective acceptance on $G g_j$ is caused by a change in commitment to a higher priority goal the condition b) ensures that a commitment regarding $G g_j$ does not block group coherence. Condition b) allows only “goal sensitive” commitments to be selected for lower level goals.

6 Conclusions

We constructed a group decision-making framework by combining judgment aggregation and multi-agent modal logic. We identified the desirable judgment aggregation properties for aggregation in collaborative groups. Our multi-agent modal logic AGE_{LTL} is an extension of BDI_{LTL} with modal operators for representing individual and collective acceptances and observations of new information. We extend the commitment strategies of Rao and Georgeff [21] to increase the reactivity of the group to new information. Having a group goal $G g$ in our framework does not imply that the members individually have the goal $G g$ and groups can have different levels of commitment to different goals.

Our framework is intended for groups that engage in joint activity. Our framework is applicable when it is not possible to assume that the agents persuade each other on a single position and goal, but it is necessary anyway that the group presents itself as

a single whole from the point of view of beliefs and goals, and above all as a rational entity that has goals justified by the beliefs it holds, and it is able to revise these goals under the light of new information. This requirement was held by Tuomela [24] and adopted in agent theory by Boella and van der Torre [1] and Lorini [18]. There are many situations where the proposal of the paper can be applied. For example in an opensource project, where several people have to discuss online to agree on which is their position on issues (e.g. which algorithm is better for a certain task) and which is their goal (e.g. delivering a new realize on which date).

Work on collaborative group activities [9, 12, 16] and group decision-making protocols [10] focus on how to define collective intentionality and how to distribute the collective intentions over the agents. We define group intention to be the collective acceptance of a group goal and focus on defining commitment strategies for the collective acceptances. An advantage of our framework is its ability to allow groups to commit to a decision to reject a goal, thus having the option to reconsider rejected goals. Furthermore, we do not only show when to reconsider, but also how, by defining reconsideration strategies. Table 1 summarizes our commitment and reconsideration strategies.

Commitment to	Release on			Change	How			
$A_N(\neg)Gg$	$\square \neg g$	g	Gg_j	\mathcal{A}_g^p	χ	$\otimes \mathcal{D}(g)$	$\otimes \bar{\pi}_g$	JA
Blind	✓							
Single-minded	✓	✓			$\mathcal{D}\text{-r}$	✓		
Extended	✓	✓	✓		Partial $\bar{\pi}\text{-r}$		✓	✓
Open-minded	✓	✓		✓	Full $\bar{\pi}\text{-r}$		✓	✓
Extended	✓	✓	✓	✓				

Table 1. $Gg_j > Gg$ and can not be pursued at the same time with Gg . $\otimes \mathcal{D}(g)$ denotes: collective attitudes for g are reconsidered. $\otimes \bar{\pi}_g$ denotes: the profile (all or some parts of it) is re-elicited.

Icard et al. [23] consider the joint revision of individual attitudes, with the revision of beliefs triggering intention revision. However, they do not allow for the revision of intentions to cause a revision of beliefs. We focus on joint reconsideration of group attitudes and we allow for both the change in epistemic and in motivational attitudes can be a cause for reconsideration.

In our framework, the new information is simultaneously available to the entire group. In the future we intend to explore the case when only some members of the group observe the new information. The only assumptions we make regarding the connectivity of the members is that they are able to communicate their acceptances and receive the aggregation result. The problem of elicitation and communication complexity in voting is a nontrivial one [6, 7] and we intend to study these properties within our framework.

In the work we presented, we do not consider how individual acceptances are formed. We can take that $B_i\phi \rightarrow A_{\{i\}}\phi$, but this need not be the case. We can define dishonest agents as those for which $B_i\phi \rightarrow A_{\{i\}}\phi$ does not hold. In this case, the agent might declare $A_{\{i\}}\phi$ while it does not believe ϕ . The question is whether there are scenarios in which incentives for doing so arise. Furthermore, given that the some reconsideration strategies call for re-elicitation of judgments, can an agent have the incentive to deliberately give judgments that would lead to sooner re-elicitation? We intend to devote more attention to answering these questions as well as studying the manipulability properties of our decision-making framework.

References

1. G. Boella and L. van der Torre. The ontological properties of social roles in multi-agent systems: Definitional dependence, powers and roles playing roles. *Artificial Intelligence and Law Journal (AILaw)*, 15(3):201–221, 2007.
2. C. Castelfranchi and F. Paglieri. The role of beliefs in goal dynamics: Prolegomena to a constructive theory of intentions. *Synthese*, 155:237–263, 2007.
3. B. Chapman. Rational aggregation. *Politics, Philosophy and Economics*, 1(3):337–354, 2002.
4. B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, 1980.
5. P. R. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.
6. V. Conitzer and T. Sandholm. Vote elicitation: Complexity and strategy-proofness. In *AAAI/IAAI*, pages 392–397, 2002.
7. V. Conitzer and T. Sandholm. Communication complexity of common voting rules. In *ACM Conference on Electronic Commerce*, pages 78–87, 2005.
8. F. Dietrich and C. List. Strategy-proof judgment aggregation. STICERD - Political Economy and Public Policy Paper Series 09, Suntory and Toyota International Centres for Economics and Related Disciplines, LSE, Aug 2005.
9. B. Dunin-Keplicz and R. Verbrugge. Collective intentions. *Fundam. Inf.*, 51(3):271–295, 2002.
10. B. Grosz and L. Hunsberger. The dynamics of intention in collaborative activity. *Cognitive Systems Research*, 7(2-3):259–272, 2007.
11. S. Hartmann, G. Pigozzi, and J. Sprenger. Reliable methods of judgment aggregation. *Journal of Logic and Computation*, forthcoming.
12. N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artif. Intell.*, 75(2):195–240, 1995.
13. S. Konieczny and R. Pino-Pérez. Merging with integrity constraints. *Lecture Notes in Computer Science*, 1638/1999:233–244, 1999.
14. L. Kornhauser and L. Sager. The one and the many: Adjudication in collegial courts. *California Law Review*, 81:1–51, 1993.
15. F. Kröger. *Temporal Logic of programs*. Springer, Berlin, 1987.
16. H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, 1990.
17. C. List and C. Puppe. Judgment aggregation: A survey. In P. Anand, C. Puppe, and P. Pattanaik, editors, *Oxford Handbook of Rational and Social Choice*. Oxford, 2009.
18. E. Lorini and D. Longin. A logical account of institutions: From acceptances to norms via legislators. In *KR*, pages 38–48, 2008.
19. E. Lorini, D. Longin, B. Gaudou, and A. Herzig. The logic of acceptance. *Journal of Logic and Computation*, 19(6):901–940, 2009.
20. G. Pigozzi, M. Slavkovic, and L. van der Torre. A complete conclusion-based procedure for judgment aggregation. In *First International Conference on Algorithmic Decision Theory Proceedings*, pages 1–13, 2009.
21. A. S. Rao and M. P. Georgeff. Intentions and rational commitment. In *In First Pacific Rim Conference on Artificial Intelligence (PRICAI-90) Proceedings*, pages 94–99, 1993.
22. K. Schild. On the relationship between bdi logics and standard logics of concurrency. *Autonomous Agents and Multi-Agent Systems*, 3(3):259–283, 2000.
23. T. Icard, E. Pacuit, and Y. Shoham. A dynamic logic of belief and intention. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference (KR-10)*, page forthcoming, 2010.
24. R. Tuomela and K. Miller. Groups beliefs. *Synthese*, 91:285–318, 1992.

Appendix – Relations between AGE_{LTL} and acceptance logic and axiomatization of AGE_{LTL}

Here we elaborate in more detail on the fusion logic AGE_{LTL} we use. The modal operator $A_S\phi$ we use is equivalent to the modal operator $A_{S:x}\phi$ of the *acceptance logic* of [19] with one syntactic and one semantic exception.

The operator $A_{S:x}\phi$ uses x ranging over a set of labels to describe the context under which the acceptance is made. In our case the context is that of the group and since we deal with only one group, we have no use of these labels. The context labels play no role in the semantics of the acceptance logic formulas.

On the semantic level, the axioms for $A_S\phi$ are all the axioms of $A_{S:x}\phi$ except two: the axiom inclusion (*Inc.*) and the axiom unanimity (*Un.*). Dropping (*Un.*) and (*Inc.*) does not affect the decidability of the logic of acceptance. (*Un.*) (not to be confused with unanimity in judgment aggregation) states that if $A_{\mathcal{N}:x}\phi$, then $\forall i \in \mathcal{N}, A_{\{i\}:x}\phi$. In our case, it is the aggregation of individual acceptances that determines the collective acceptance. Since we use the acceptances to model judgments, we do not want an axiom that states that the individual judgments mirror the collective judgments. The agents use the collective acceptance when functioning as a group and their private beliefs when acting as individuals. In our framework we do not model the private mental states, but only individual acceptances which are “declared” to all the agents in the group.

(*Inc.*) states that if a the group C accepts ϕ , so will any subgroup $B \subset C$. In our case, the judgment aggregation over the profile containing only the judgment sets of B can produce a different collective judgment set than the profile containing all the judgment sets of C .

The axiomatization of the AGE_{LTL} logic is thus:

- (**ProTau**) All principles of propositional calculus
- (**LTLTau**) All axioms and derivation rules of LTL
- (**K-G**) $G(\phi \rightarrow \psi) \rightarrow (G\phi \rightarrow G\psi)$
- (**K-E**) $E(\phi \rightarrow \psi) \rightarrow (E\phi \rightarrow E\psi)$
- (**K-A**) $A_S(\phi \rightarrow \psi) \rightarrow (A_S\phi \rightarrow A_S\psi)$
- (**PAccess**) $A_S\phi \rightarrow A_M A_S\phi$ if $M \subseteq S$
- (**NAccess**) $\neg A_S\phi \rightarrow A_M \neg A_S\phi$ if $M \subseteq S$
- (**Mon**) $\neg A_S\perp \rightarrow \neg A_M\perp$ if $M \subseteq S$
- (**MP**) From $\vdash \phi$ and $\vdash (\phi \rightarrow \psi)$ infer $\vdash \psi$
- (**Nec-A**) From $\vdash \phi$ infer $\vdash A_S\phi$
- (**Nec-G**) From $\vdash \phi$ infer $\vdash G\phi$
- (**Nec-E**) From $\vdash \phi$ infer $\vdash E\phi$

Given $\mathcal{M} = \langle W, \mathcal{R}, \mathcal{G}, \mathcal{E}, \mathcal{A}, L \rangle$ and $s \in W$, the truth conditions for the formulas of AGE_{LTL} (in a situation s) are:

- $\mathcal{M}, s \not\models \perp$;
- $\mathcal{M}, s \models p$ if and only if $p \in L(p)$;
- $\mathcal{M}, s \models \neg\phi$ if and only if $\mathcal{M}, s \not\models \phi$;
- $\mathcal{M}, s \models \phi \wedge \psi$ if and only if $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models A_{\mathcal{N}}\phi$ if and only if $\mathcal{M}, s' \models \phi$ for all $(s, s') \in \mathcal{A}$;

- $\mathcal{M}, s \models G\phi$ if and only if $\mathcal{M}, s' \models \phi$ for all $(s, s') \in \mathcal{G}$;
- $\mathcal{M}, s \models E\phi$ if and only if $\mathcal{M}, s' \models \phi$ for all $(s, s') \in \mathcal{E}$;
- $\mathcal{M}, s \models \mathbf{X}\phi$ if and only if $\mathcal{M}, s' \models \phi$ for the $s', (s, s') \in \mathcal{R}$.
- $\mathcal{M}, s \models \phi\mathbf{U}\psi$ if and only if $\mathcal{M}, s \models \phi$; $\mathcal{M}, s^i \models \phi$ for all $s^i, i \in \{1, 2, \dots, k\}$ such that $\{(s, s^1), (s^1, s^2), \dots, (s^{k-1}, s^k)\} \in \mathcal{R}$ and for s^{k+1} such that $(s^k, s^{k+1}) \in \mathcal{R}$ it holds $\mathcal{M}, s^{k+1} \not\models \phi$ and $\mathcal{M}, s^{k+1} \models \psi$.

A formula ϕ is true in an AGE_{LTL} model \mathcal{M} if and only if $\mathcal{M}, s \models \phi$ for every situation $s \in W$. The formula ϕ is valid (noted $\models_{AGE_{LTL}}$) if and only if ϕ is true in all AGE_{LTL} models. The formula ϕ is AGE_{LTL} -satisfiable if and only if the formula $\neg\phi$ is not AGE_{LTL} valid.

MERCURIO: An Interaction-oriented Framework for Designing, Verifying and Programming Multi-Agent Systems*

Matteo Baldoni¹, Cristina Baroglio¹, Federico Bergenti⁴, Antonio Boccalatte³,
Elisa Marengo¹, Maurizio Martelli³, Viviana Mascardi³, Luca Padovani¹,
Viviana Patti¹, Alessandro Ricci², Gianfranco Rossi⁴, and Andrea Santi²

¹ Università degli Studi di Torino
{baldoni,baroglio,emarengo,padovani,patti}@di.unito.it
² Università degli Studi di Bologna
{a.ricci,a.santi}@unibo.it
³ Università degli Studi di Genova
{martelli,mascardi}@disi.unige.it, nino@dist.unige.it
⁴ Università degli Studi di Parma
{federico.bergenti,gianfranco.rossi}@unipr.it

Abstract. This is a position paper reporting the motivations, the starting point and the guidelines that characterize the MERCURIO⁵ project proposal, submitted to MIUR PRIN 2009⁶. The aim is to develop formal models of interactions and of the related support infrastructures, that overcome the limits of the current approaches by explicitly representing not only the agents but also the computational environment in terms of rules, conventions, resources, tools, and services that are functional to the coordination and cooperation of the agents. The models will enable the verification of interaction properties of MAS from the global point of view of the system as well as from the point of view of the single agents, due to the introduction of a novel social semantic of interaction based on commitments and on an explicit account of the regulative rules.

1 Motivation

The growing pervasiveness of computer networks and of Internet is an important catalyst pushing towards the realization of *business-to-business* and *cross-business solutions*. Interaction and coordination, central issues to any distributed system, acquire in this context a special relevance since they allow the involved groups to integrate by interacting according to the agreed contracts, to share best practices and agreements, to cooperatively exploit resources and to facilitate the identification and the development of new products.

* Position paper

⁵ Italian name of Hermes, the messenger of the gods in Greek mythology.

⁶ Despite the label “2009”, it is the just closed call for Italian National Projects, <http://prin.miur.it/index.php?pag=2009>.

The issues of interaction, coordination and communication have been receiving great attention in the area of Multi-Agent Systems (MAS). MAS are, therefore, the tools that could better meet these needs by offering the proper abstractions. Particularly relevant in the outlined application context are a shared and inspectable specification of the rules of the MAS and the verification of global properties of the interaction, like the interoperability of the given roles, as well as properties like the conformance of an agent specification (or of its run-time behavior) to a protocol. In open environments, in fact, it is important to have guaranties on how interaction will take place, coping with notions like responsibility and commitment. Unfortunately, current proposals of platforms and languages for the development of MAS do not supply high level tools for directly implementing this kind of specifications. As a consequence, they do not support the necessary forms of verification, with a negative impact on the applicability of MAS to the realization of business-to-business and cross-business systems.

Let us consider, for instance, JADE [4, 18, 16, 17], which is one of the best known infrastructures, sticking out for its wide adoption also in business contexts. JADE agents communicate by exchanging messages that conform to FIPA ACL [3]. According to FIPA ACL mentalistic approach, the semantics of messages is given in terms of preconditions and effects on the mental states of the involved agents, which are assumed to share a common ontology. Agent platforms based on FIPA exclusively provide syntactic checks of message structures, entrusting the semantics issues to agent developers. This hinders the applicability to open contexts, where it is necessary to coordinate autonomous and heterogeneous agents and it is not possible to assume mutual trust among them. In these contexts it is necessary to have an unambiguous semantics allowing the verification of interaction properties before the interaction takes place [52] or during the interaction [9], preserving at the same time the privacy of the implemented policies.

The mentalistic approach does not allow to satisfy all these needs [40]; it is suitable for reasoning from the *local point of view* of a single agent, but it does not allow the verification of interaction properties of a MAS from a *global point of view*. One of the reasons is that the reference model *lacks* an abstraction for the representation, by means of a public specification, of elements like (i) resources and services that are available in the environment/context in which agents interact and (ii) the rules and protocols, defining the interaction of agents through the environment/context. All these elements belong to (and contribute to make) the environment of the interacting agents. Such an abstraction, if available, would be the natural means for encapsulating resources, services, and functionalities (like ontological mediators) that can support the communication and the coordination of agents [67, 66, 43], thus facilitating the verification of the properties [13]. It could also facilitate the interaction of agents implemented in different languages because it would be sufficient that each language implements the primitives for interacting with the environment [1]. One of the consequences of the lack of an explicit representation of the environment is that only forms of *direct commu-*

nication are possible. On the contrary, in the area of distributed systems and also in MAS alternative communication models, such as the generative communication based on tuple spaces [32], have been put forward. These forms of communication, which do not necessarily require a space-time coupling between agents, are not supported.

The issues that we mean to face have correspondences with issues concerning normative MAS [70] and Artificial Institutions [31, 65]. The current proposals in this field, however, do not supply all of the solutions that we need: either they do not account for indirect forms of communication or they lack mechanisms for allowing the a priori verification of global properties of the interaction. As [31, 65] witness, there is, instead, an emerging need of defining a more abstract notion of action, which is not limited to direct speech acts. In this case, institutional actions are performed by executing instrumental actions that are conventionally associated with them. Currently, instrumental actions are limited to speech acts but this representation is not always natural. For instance, for voting in the human world, people often raise their hands rather than saying the name corresponding to their choice. If the environment were represented explicitly it would be possible to use a wider range of instrumental actions, that can be perceived by the other agents through the environment that acts as a medium.

Our goal is, therefore, to propose an infrastructure that overcomes such limits. The key of the proposal is the adoption of a *social approach* to communication [45, 14, 13, 12], based on a model that includes an explicit representation not only of *agents* but also of their *environment*, as a collection of virtual and physical resources, tools and services, “artifacts” as intended in the Agents & Artifacts (A&A) meta-model [43], which are shared, used and adapted by the agents, according to their goals. The introduction of environments is fundamental to the adoption of an observational (social) semantics, like the one used in commitment protocols, in that it supplies primitives that allow agents to perceive and to modify the environment itself and, therefore, to interact and to coordinate with one another in a way that satisfies the rules of the environment. On the other hand, the observational semantics is the only sufficiently general semantics to allow forms of interaction and of communication that do not rely solely on direct speech acts. As a consequence we will include models where communication is mediated by an environment, that encapsulates and applies rules and constraints aimed at coordinating agents at the organization level, and integrates *ontological mediation* functionalities. The environment will provide the *contract* that agents should respect and a context into which interpreting their actions. In this way, it will be possible to *formally verify the desired properties of the interaction, a priori and at execution time*.

2 Vision

The focus of our proposal is on the definition of formal models of interactions and of the related support infrastructures, which explicitly represent not only the agents but also the environment in terms of rules of interaction, conventions,

resources, tools, and services that are functional to the coordination and cooperation of the agents. These models must allow both direct and indirect forms of communication, include ontological mediators, and enable the verification of interaction properties of MAS from the global point of view of the system as well as from the point of view of the single agents. The approach we plan to pursue in order to define a formal model of interaction is based on a revision in social terms of the interaction and of the protocols controlling it, along the lines of [14, 13, 12]. Furthermore, we will model the environment, in the sense introduced by the A&A meta-model [43]. This will lead to the study of communication forms mediated by the environment. The resulting models will be validated by the implementation of software tools and of programming languages featuring the designed abstractions. More in details, with reference to Fig. 1, the goals are:

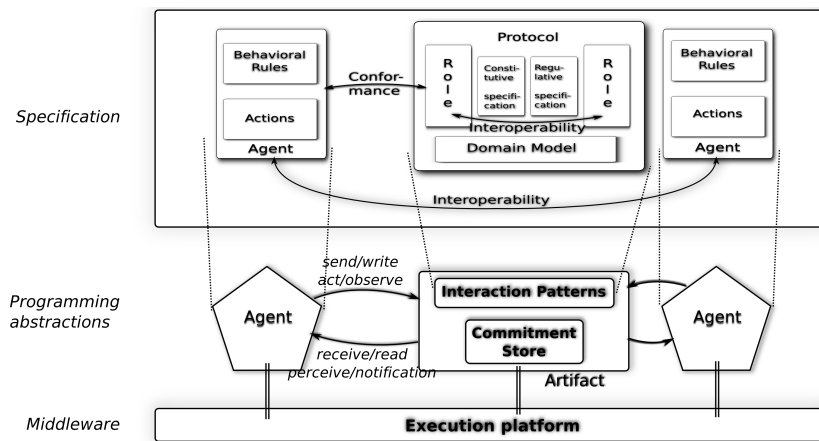


Fig. 1. The MERCURIO architecture.

To introduce a formal model for specifying and controlling the interaction. The model (top level of Fig. 1) must be equipped with an observational (commitment-based) semantics and must be able to express not only direct communicative acts but also interactions mediated by the environment. This will enable forms of verification that encompass both global interaction properties and specific agent properties such as interoperability and conformance [11]. The approach does not hinder agent autonomy, it guarantees the privacy of the policies implemented by the agents, and consequently favors the composition of heterogeneous agents. The model will be inspired by the social approach introduced in [45] and subsequently extended in [14, 13, 12].

To define high-level environment models supporting the forms of interactions and coordination between agents outlined above. These models must

support: interaction protocols based on commitments; the definition of rules on the interaction; forms of mediated communication and coordination between agents (such as stigmergic coordination). They must also enable forms of a priori and runtime verification of the interaction. To these aims, we plan to use the A&A meta-model [58, 67, 43, 56] and the corresponding notion of programmable environment [57] (programming abstractions level of Fig. 1).

To integrate ontologies and ontological mediators in the definition of the models so as to guarantee openness and heterogeneity of MAS. Mediation will occur at two distinct levels: the one related to the vocabulary and domain of discourse and the one that characterizes the social approach where it is required to bind the semantics of the agent actions with their meaning in social terms. Ontological mediators will be realized as artifacts.

To integrate the abstractions defined in the above models within programming languages and frameworks. In particular, we plan to integrate the notions of agents, of environment, of direct and mediated communication, and of ontological mediators. Possible starting points are the aforementioned FIPA ACL standard and the works that focus on the integration of agent-oriented programming languages with environments [55]. The JaCa platform [57], integrating Jason and CArtAgO, will be taken as reference. This will form the execution platform of Fig. 1 and will supply the primitives for interacting with the environments.

To develop an open-source prototype of software infrastructure for the experimentation of the defined models. The prototype will integrate and extend existing technologies such as JADE [18, 16, 17] (as a FIPA-compliant framework), CArtAgO [1] (for the programming and the execution of environments), Jason (as a programming language for BDI agents), MOISE [35] (as organizational infrastructure).

To identify applicative scenarios for the evaluation of the developed models and prototypes. In this respect we regard the domain of Web services as particularly relevant because of the need to deploy complex interactions having those characteristics of flexibility that agents are able to guarantee. Another interesting application regards the verification of adherence of bureaucratic procedures of public administration with respect to the current normative. Specific case studies will be defined in collaboration with those companies that have stated interest towards the project.

3 State of Art

These novel elements, related to the formation of and the interaction within decentralized structures, find an initial support in proposals from the literature in the area of MAS. Current proposals, however, are still incomplete in that they supply solutions to single aspects. For instance, electronic institutions [28, 10, 35, 34] regulate interaction, tackle open environments and their semantics allows the verification of properties but they only tackle direct communication protocols, based on speech acts, and do not include an explicit notion of environment. Commitment protocols [45, 69], effective in open systems and allowing more general

forms of communication, do not supply behavioral patterns, and for this reason it is impossible to verify properties of the interaction. Eventually, most of the models and architectures for environments prefigure simple/reactive agent models without defining semantics, that are comparable to the ones for ACL, and without explaining how such proposals could be integrated with direct communication models based on speech acts. We classify the relevant contributions in the literature according to the objectives and the methodological aspects that will be examined in-depth along the project.

3.1 Formal Models for Regulating the Interaction in MAS

This topic has principally been tackled by modeling interaction protocols. Most of protocol representations refer to classic models, such as Petri nets, finite state machines, process algebras, and aim at capturing the expected interaction flow. An advantage of this approach is that it supports the verification of interaction properties [52, 21, 11], such as: verifying the interoperability of the system and verifying if certain modifications of a system preserve some desired properties (a crucial issue in open domains where agents can enter/leave the system at any time). Singh and colleagues criticize the use of procedural specifications because too rigid [60, 24, 69]: agents cannot take advantage of opportunities that emerge along the interaction and that are not foreseen by their procedure. Another issue is that communication languages use a BDI semantics (FIPA ACL is an example), where each agent has goals and beliefs of its own. At the system level, however, it is impossible to perform introspection of agents, which are, for this reason, black boxes. For what concerns the verification of properties this approach allows agents to draw conclusions about their own behavior but not to verify global properties of the system [40, 64].

Both problems are solved by commitment protocols [45, 60], which rely on an observational semantics of the interaction and offer adequate flexibility to agents. Moreover, they do not require the spatio-temporal coupling of agents (as instead direct communication does). Another advantage is that, though remaining black boxes, agents agree on the meaning of the social actions of the protocol. Since interactions are observable and their semantics is shared, each agent should be able to draw conclusions concerning the system as a whole. Unfortunately, besides some preliminary studies [61], the state of art does not contain proposals on how performing the verifications in a MAS, ruled by this kind of protocols. A relevant feature seems to be the introduction, within commitment protocols, of behavioral rules which constrain the possible evolutions of the social state [13, 12].

3.2 Environment Models

The notion of environment has always played a key role in the context of MAS; recently, it started to be considered as a first-class abstraction useful for the design and the engineering of MAS [67]. A&A [43] follows this perspective, being a meta-model rooted upon Activity Theory and Computer Support Cooperative

Work that defines the main abstractions for modeling a MAS, and in particular for modeling the environment in which a MAS is situated. A&A promotes a vision of an endogenous environment, that is a sort of software/computational environment, part of the MAS, that encapsulates the set of tools and resources useful/required by agents during the execution of their activities. A&A introduces the notion of artifact as the fundamental abstraction used for modeling the resources and the tools that populates the MAS environment. The introduction of the environment as a new first-class abstraction requires new engineering approaches for programming the MAS environment. The CArtAgO framework [57] has been devised precisely for copying this new necessity. It provides the basis for the engineering of MAS environments on the base of: (i) a proper computational model and (ii) a programming model for the design and the development of the environments on the base of the A&A meta-model. In particular, it provides those features that are important from a software engineering point of view: *abstraction*, it preserves the agent abstraction level, since the main concepts used to define application environments, i.e. artifacts and workspaces, are first-class entities in the agents world, and the interaction with agents is built around the agent-based concepts of action and perception (use and observation); *modularity and encapsulation*, it provides an explicit way to modularize the environment, where artifacts are components representing units of functionality, encapsulating a partially-observable state and operations; *extensibility and adaptation*, it provides a direct support for environment extensibility and adaptation, since artifacts can be dynamically constructed (instantiated), disposed, replaced, and adapted by agents; *reusability*, it promotes the definition of types of artifact that can be reused as tools in different application contexts, such as in the case of coordination artifacts empowering agent interaction and coordination, such as blackboards and synchronizers. These features will be advantageous in the realization of the second goal of the project, w.r.t. approaches like [25], where commitment stores, communication constraints and the interaction mechanisms reside in the middleware, which shields them from the agents. This has two disadvantages: the first is that even though all these elements are accounted for in the high level specification, the lack of a corresponding programming abstraction makes it difficult to verify whether the system corresponds to the specification; the second is a lack of flexibility, in that it is not possible for the agents to dynamically change the rules of interaction or to adopt kinds of communication that are not already implemented in the middleware.

In the state of the art numerous applications of the endogenous environments, i.e. environments used as a computational support for the agents' activities, have been explored, including coordination artifacts [44], artifacts used for realizing argumentation by means of proper coordination mechanisms [42], artifacts used for realizing stigmergic coordination mechanisms [54, 48], organizational artifacts [34, 49, 50]. Even if CArtAgO can be considered a framework sufficiently mature for the concrete developing of software/computational MAS environments it can not be considered "complete" yet. Indeed at this moment the state of the art and in particular the CArtAgO framework are still lacking: (i) a reference standard

on the environment side comparable to the existing standards in the context of the agents direct communications (FIPA ACL), (ii) the definition of a rigorous and formal semantics, in particular related to the artifact abstraction, (iii) an integration with the current communication approaches (FIPA ACL, KQML, etc.), and finally (iv) the support of semantic models and ontologies.

3.3 Multi-agent Organizations and Institutions

The possibility of controlling and specifying interactions is relevant also for areas like the organizational theory [39, 70, 15, 35] and electronic institutions [28, 10] areas. Tendentiously, the focus is orthogonal to the one posed on interaction protocols, in that it concerns the modeling of the structure rather than of the interaction.

The abstract architecture of e-Institutions (e.g. Ameli [28]), places a middleware composed of governors and staff agents between participating agents and an agent communication infrastructure (e.g. JADE [18, 16, 17]). The notion of environment is dialogical: it is not something agents can sense and act upon but a conceptual one that agents, playing within the institution, can interact with by means of norms and laws, based on specific ontologies, social structures, and language conventions. Agents communicate with each other by means of speech acts and, behind the scene, the middleware mediates such communication. The extension proposed for situated e-Institutions [10] introduces the notion of “World of Interest” to model the environment, that is external to the MAS but which is relevant to the MAS application. The infrastructure of the e-Institution, in this case, mediates also the interaction of the agents in the MAS with the view of the environment that it supplies. Further along this line, but in the context of organizations, ORA4MAS [34] proposes the use of artifacts to enable the access of the agents in the MAS to the organization, providing a working environment that agents can perceive, act upon and adapt. Following the A&A perspective, they are concrete bricks used to structure the agents’ world: part of this world is represented by the organizational infrastructure, part by artifacts introduced by specific MAS applications, including entities/services belonging to the external environment.

According to [10] there are, however, two significant differences among artifacts and e-Institutions: (i) e-Institutions are tailored to a particular, though large, family of applications while artifacts are more generic; (ii) e-Institutions are a well established and proven technology that includes a formal foundation, and advanced engineering and tool support, while for artifacts, these features are still in a preliminary phase. One of the aims of MERCURIO is to give to artifacts both the formal foundation (in terms of commitments and interaction patterns) and the engineering tools that they are still missing. The introduction of interaction patterns with an observational nature, allowing the verification of global properties, that we aim at studying, will allow the realization of e-Institutions by means of artifacts. The artifact will contain all the features necessary for monitoring the on-going interactions and for detecting violations. A second step will be to consider organizations and realize them again by means of artifacts.

To this aim, it is possible to exploit open source systems like CArtaGO [1], for the programming and the execution of environments, and MOISE [35], as organizational infrastructure.

3.4 Semantic Mediation in MAS

The problem of semantic mediation at the vocabulary and domain of discourse levels was faced for the first time by the “Ontology Service Specification” [8] issued by FIPA in 2001. According to that specification, an “Ontology Agent” (OA, for short) should be integrated in the MAS in order to provide services such as translating expressions between different ontologies and/or different content languages and answering queries about relationships between terms or between ontologies. Although the FIPA Ontology Service Specification represents the first and only attempt to analyze in a systematic way the services that an OA should provide for ensuring semantic interoperability in an open MAS, it has many limitations. The main one is the assumption that each ontology integrated in the MAS adheres to the OKBC model [6]. Currently, in fact, the most widely accepted ontology language is OWL [7] which is quite different from OKBC and cannot be converted to it in an easy and automatic way. Also, agents are allowed to specify only one ontology as reference vocabulary for a given message, which is a strong limitation since an agent might use terms from different ontologies in the same message, and hence it might want to refer to more than one ontology at the same time.

Maybe due to these limitations, there have been really few attempts to design and implement OAs. The first dates back to 2001 [62] and realizes an OA for the COMTEC platform that implements a subset of the services of a generic FIPA-compliant OA. In 2007 [46] integrated an OA into AgentService, a FIPA compliant framework based on .NET [63]. Ontologies in AgentService are represented in OKBC, and hence the implementation of their OA is fully compliant with the FIPA specification, although the offered services are a subset of the possible ones. The only two attempts of integrating a FIPA-compliant OA into JADE, we are aware of, are [41], and [23]. Both follow the FIPA specification but adapt it to ontologies represented in OWL. The first proposal is aimed at storing and modifying OWL ontologies: the OA agent exploits the Jena library [36] to this aim. The second proposal, instead, faces the problem of “*answering queries about relationships between terms or between ontologies*”. The solution proposed by the authors exploits ontology matching techniques [29]. Apart from [23], no other existing proposal faces that problem. Among non FIPA-compliant solutions, we mention [37], which focuses on the process of mapping and integrating ontologies in a MAS thanks to a set of agents which collaborate together, and the proposal in [47], which implements the OA as a web service, in order to offer its services also over the Internet.

As far as semantic mediation at the social approach level is concerned, we are aware of no proposals in the literature. In order to take the context of count-as rules into account, we plan to face this research issue by exploiting context aware semantic matching techniques, that extend and improve those described in [38].

3.5 Software Infrastructures for Agents

The tools currently available to agent developers fail in supporting both semantic interoperability and goal-directed reasoning. Nowadays, the development of agents and multi-agent systems is based on two kinds of tools: agent platforms and BDI (or variations) development environments. Agent platforms, such as JADE [18, 16, 17] and FIPA-OS [2] provide only a transport layer and some basic services, but they do not provide any support for goal-directed behavior. Moreover, they lack support for semantic interoperability because they do not take into account the semantics of the ACL they adopt. The available BDI development environments, such as Jadex [22] and 2APL [27], support only syntactic interoperability because they do not exploit their reasoning engines to integrate the semantics of the adopted ACL.

The research on Agent Communication Languages (ACL) is constantly headed towards semantic interoperability [33] because the most common ACLs, e.g., KQML [30] and FIPA ACL [3], provide each message with a declarative semantics that was explicitly designed to support goal-directed reasoning. Unfortunately, the research on ACLs only marginally investigated the decoupling properties of this kind of languages (see, e.g., [19, 20]). To support the practical development of software agents, several programming languages have thus been introduced to incorporate some of the concepts from agent logics. Some languages use actions as their starting point to define commitments (Agent-0, [59]), intentions (AgentSpeak(L), [53]) and goals (3APL, [26]).

4 Expected Results

The achievements expected from this research are of different natures: scientific result that will advance the state of the art, software products deriving from the development of implementations, and upshots in applicative settings.

The formal model developed in MERCURIO will extend commitment protocols by introducing behavioral rules. The starting point will be the work done in [14, 13, 12]. This will advance the current state of the art with respect to the specification of commitment protocols and also with respect to the verification of interaction properties (like interoperability and conformance), for which there currently exist only preliminary proposals [61]. Another advancement concerns the declarative specification of protocols and their usage by designers and software engineers. The proposals coming from MERCURIO conjugate the flexibility and openness features that are typical of MAS with the needs of modularity and compositionality that are typical of design and development methodologies. The adoption of commitment protocols makes it easier and more natural to represent (inter)actions that are not limited to communicative acts but that include interactions mediated by the environment, namely actions upon the environment and the detection of variations of the environment ruled by “contracts”.

For what concerns the coordination infrastructure, a first result will be the definition of environments based on the A&A meta-model and on the CArtAgO

computational framework, that implement the formal models and the interaction protocols mentioned above. A large number of the environments, described in the literature supporting communication and coordination, have been stated considering purely reactive architectures. In MERCURIO we will formulate environment models that allow goal/task-oriented agents (those that integrate pro-activities and re-activities) the participation to MAS. Among the specific results related to this, we foresee an advancement of the state of the art with respect to the definition and the exploitation of forms of stigmergic coordination [54] in the context of intelligent agent systems. A further contribution regards the flexible use of artifact-based environments by intelligent agents, and consequently the reasoning techniques that such agents may adopt to take advantage of these environments. First steps in this direction, with respect to agents with BDI architectures, have been described in [51, 48].

The MERCURIO project aims at putting forward an extension proposal for the FIPA ACL standard, where the FIPA ACL-based communication is integrated with forms of interactions, that are enabled and mediated by the environment. This will lead to an explicit representation of environments as first-class entities (in particular endogenous environments based on artifacts) and of the related model of actions/perceptions. Furthermore we will formulate an improved version of the MAS programming language/framework JaCa, where we plan to integrate the agent-oriented programming language Jason, which is based on a BDI architecture, with the CArtAgO computational framework. This result will extend the work done so far in this direction [55, 57].

In MERCURIO we will implement a prototype of the reference infrastructural model defined by the project. The prototype will be based on the development and integration of existing open-source technologies including JADE [4], the reference FIPA platform, CArtAgO [1], the reference platform and technology for the programming and execution of environments, and agent-oriented programming languages such as Jason [5] and 2APL [27]. The software platform will include implementations of the “context sensitive” ontology alignment algorithms developed in MERCURIO. The algorithms will be evaluated against standard benchmarks and also against the case studies devised in MERCURIO.

Aside from the effects on research contexts, we think that the project may give significant contributions also to industrial applicative contexts, in particular to those companies working on software development in large, distributed systems and in service-oriented architectures. Among the most interesting examples are the integration and the cooperation of e-Government applications (services) spread over the nation. For this reason, MERCURIO will involve some companies in the project, and in particular in the definition of realistic case studies against which the project’s products will be validated. As regards (Web) services, some fundamental aspects promoted by the SOA model, such as autonomy and decoupling, are addressed in a natural way by the agent-oriented paradigm. Development and analysis of service-oriented systems can benefit from the increased level of abstraction offered by agents, by reducing the gap between the modeling, design, development, and implementation phases.

Acknowledgements

We thank S. Mantix for the valuable support and helpful discussions.

References

1. CARTAGO. <http://cartago.sourceforge.net>.
2. FIPA OS. <http://fipa-os.sourceforge.net>.
3. FIPA specifications. <http://www.fipa.org>.
4. JADE. <http://jade.tilab.com/>.
5. JASON. <http://jason.sourceforge.net>.
6. OKBC. <http://www.ai.sri.com/okbc/>.
7. OWL. <http://www.w3.org/TR/owl-features/>.
8. Fipa architecture board, fipa ontology service specification, 2001. <http://www.fipa.org/specs/fipa00086/>.
9. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable agent interaction in abductive logic programming: The sciff framework. *ACM Trans. Comput. Log.*, 9(4), 2008.
10. J. L. Arcos, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. E4MAS Through Electronic Institutions. In Weyns et al. [68], pages 184–202.
11. M. Baldoni, C. Baroglio, A. K. Chopra, N. Desai, V. Patti, and M. P. Singh. Choice, interoperability, and conformance in interaction protocols and service choreographies. In *Proc. of the 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009*, pages 843–850. IFAAMAS, 2009.
12. M. Baldoni, C. Baroglio, and E. Marengo. Behavior-oriented Commitment-based Protocols. In H. Coelho and M. Wooldridge, editors, *Proc. of 19th European Conference on Artificial Intelligence, ECAI 2010*, Lisbon, Portugal, August 2010. To appear.
13. M. Baldoni, C. Baroglio, and E. Marengo. Commitment-based Protocols with Behavioral Rules and Correctness Properties of MAS. In A. Omicini, S. Sardina, and W. Vasconcelos, editors, *Proc. of International Workshop on Declarative Agent Languages and Technologies, DALT 2010, held in conjunction with AAMAS 2010*, pages 66–83, Toronto, Canada, May 2010.
14. M. Baldoni, C. Baroglio, and E. Marengo. Constraints among Commitments: Regulatory Specification of Interaction Protocols. In A. Artikis, J. Bentahar, A. Artikis, and F. Dignum, editors, *Proc. of International Workshop on Agent Communication, AC 2010, held in conjunction with AAMAS 2010*, pages 2–18, Toronto, Canada, May 2010.
15. M. Baldoni, G. Boella, and L. van der Torre. Bridging Agent Theory and Object Orientation: Agent-like Communication among Objects. In R. H. Bordini, M. Dastani, J. Dix, and A. Seghrouchni, editors, *Post-Proc. of the International Workshop on Programming Multi-Agent Systems, ProMAS 2006*, volume 4411 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 149–164. Springer, 2007.
16. F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. JADE - A Java Agent Development Framework. In R. H. Bordini, M. Dastani, J. JDix, and A. El Fallah-Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 125–147. Springer, 2005.

17. F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE: A software framework for developing multi-agent applications. Lessons learned. *Information & Software Technology*, 50(1-2):10–21, 2008.
18. F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework. *Softw., Pract. Exper.*, 31(2):103–128, 2001.
19. F. Bergenti and F. Ricci. Three Approaches to the Coordination of Multiagent Systems. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC)*, pages 367–372, Madrid, Spain, March 2002. ACM.
20. F. Bergenti, G. Rimassa, M. Somacher, and L. M. Botelho. A FIPA Compliant Goal Delegation Protocol. In M.-P. Huget, editor, *Communication in Multiagent Systems, Agent Communication Languages and Conversation Policies*, volume 2650 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2003.
21. L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella. When are Two Web Services Compatible? In M.-C. Shan, U. Dayal, and M. Hsu, editors, *Technologies for E-Services, 5th International Workshop, TES 2004*, volume 3324 of *Lecture Notes in Computer Science*, pages 15–28, Toronto, Canada, August 2004. Springer.
22. L. Braubach, A. Pokahr, and W. Lamersdorf. Jadex: A bdi agent system combining middleware and reasoning. In *Software Agent-Based Applications, Platforms and Development Kits*. Birkhauser Book, 2005.
23. D. Briola, A. Locoro, and V. Mascardi. Ontology Agents in FIPA-compliant Platforms: a Survey and a New Proposal. In M. Baldoni, M. Cossentino, F. De Paoli, and V. Seidita, editors, *Proc. of WOA 2008: Dagli oggetti agli agenti, Evoluzione dell’agent development: metodologie, tool, piattaforme e linguaggi*. Seneca Edizioni, 2008.
24. A. K. Chopra and M. P. Singh. Nonmonotonic Commitment Machines. In F. Dignum, editor, *Advances in Agent Communication, International Workshop on Agent Communication Languages*, volume 2922 of *Lecture Notes in Computer Science*, pages 183–200, Melbourne, Australia, July 2003. Springer.
25. A. K. Chopra and M. P. Singh. An Architecture for Multiagent Systems: An Approach Based on Commitments. In *Proceedings of the AAMAS Workshop on Programming Multiagent Systems (ProMAS)*, 2009.
26. M. Dastani, B. M. van Riemsdijk, F. Dignum, and J.-J. Ch. Meyer. A Programming Language for Cognitive Agents Goal Directed 3APL. In M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Programming Multi-Agent Systems, First International Workshop, PROMAS 2003*, volume 3067 of *Lecture Notes in Computer Science*, pages 111–130, Melbourne, Australia, July 2003. Springer.
27. Mehdi Dastani. 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.
28. M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. AMELI: An Agent-Based Middleware for Electronic Institutions. In *AAMAS*, pages 236–243. IEEE Computer Society, 2004.
29. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.
30. T. Finin, Y. Labrou, and J. Mayfield. Kqml as an agent communication language. In Bradshaw J., editor, *Software Agents*. MIT Press, Cambridge, 1997.
31. N. Fornara, F. Viganò, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.
32. David Gelernter. Generative communication in linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, 1985.
33. S. Heiler. Sematic Interoperability. *ACM Computing Surveys*, 27(2):271–273, 1995.

34. J. F. Hubner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents: “Giving the organisational power back to the agents”. *Autonomous Agents and Multi-Agent Systems*, 20, 2009.
35. Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. Developing organised multiagent systems using the MOISE. *IJAOSE*, 1(3/4):370–395, 2007.
36. *Jena – A Semantic Web Framework for Java*. Online, accessed on June, 14th, 2010. <http://jena.sourceforge.net/>.
37. L. Li, B. Wu, and Y. Yang. Agent-based ontology integration for ontology-based application. In *Australasian Ontology Workshop, AOW 2005, Proceedings*, pages 53–59, 2005.
38. V. Mascardi, A. Locoro, and F. Larosa. Exploiting Prolog and NLP techniques for matching ontologies and for repairing correspondences. In *24th Convegno Italiano di Logica Computazionale, CILC 2009, Proceedings*, 2009.
39. C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social Roles and their Descriptions. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 267–277, Whistler, Canada, June 2004. AAAI Press.
40. P. McBurney and S. Parsons. Games That Agents Play: A Formal Framework for Dialogues between Autonomous Agents. *Journal of Logic, Language and Information*, 11(3):315–334, 2002.
41. M. Obitko and V. Snáél. Ontology repository in multi-agent system. In M. H. Hamza, editor, *Conference on Artificial Intelligence and Applications, AIA 2004, Proceedings*, 2004.
42. E. Oliva, P. McBurney, and A. Omicini. Co-argumentation Artifact for Agent Societies. In I. Rahwan, S. Parsons, and C. Reed, editors, *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007*, volume 4946 of *Lecture Notes in Computer Science*, pages 31–46, Honolulu, HI, USA, May 2007. Springer.
43. A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, 2008.
44. A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination Artifacts: Environment-Based Coordination for Intelligent Agents. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 286–293, New York, USA, August 2004.
45. Singh M. P. An Ontology for Commitments in Multiagent Systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
46. A. Passadore, C. Vecchiola, A. Grosso, and A. Boccalatte. Designing agent interactions with Pericles. In *2nd International Workshop on Ontology, Conceptualization and Epistemology for Software and System Engineering, ONTOSE 2007, Proceedings*, 2007.
47. A. Peña, H. Sossa, and F. Gutierrez. Web-services based ontology agent. In *2nd International Conference on Distributed Frameworks for Multimedia Applications, DFMA 2006, Proceedings*, pages 1–8, 2006.
48. M. Piunti and A. Ricci. Cognitive Use of Artifacts: Exploiting Relevant Information Residing in MAS Environments. In J.-J. Ch. Meyer and J. Broersen, editors, *Knowledge Representation for Agents and Multi-Agent Systems, First International Workshop, KRAMAS 2008*, volume 5605 of *Lecture Notes in Computer Science*, pages 114–129, Sydney, Australia, September 2008. Springer.
49. M. Piunti, A. Ricci, O. Boissier, and J. F. Hübner. Embodied Organisations in MAS Environments. In L. Braubach, W. van der Hoek, P. Petta, and A. Pokahr, editors,

- Multiagent System Technologies, 7th German Conference, MATES 2009*, volume 5774 of *Lecture Notes in Computer Science*, pages 115–127, Hamburg, Germany, September 2009. Springer.
50. M. Piunti, A. Ricci, O. Boissier, and J. F. Hübner. Embodying Organisations in Multi-agent Work Environments. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT*, pages 511–518. IEEE, 2009.
 51. M. Piunti, A. Ricci, L. Braubach, and A. Pokahr. Goal-Directed Interactions in Artifact-Based MAS: Jadex Agents Playing in CARTAGO Environments. In *IAT*, pages 207–213. IEEE, 2008.
 52. S. K. Rajamani and J. Rehof. Conformance Checking for Models of Asynchronous Message Passing Software. In *Computer Aided Verification CAV'02*, number 2404 in LNCS, pages 166–179. Springer, 2002.
 53. A. S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In W. Van de Velde and J. W. Perram, editors, *MAAMAW*, volume 1038 of *Lecture Notes in Computer Science*, pages 42–55. Springer, 1996.
 54. A. Ricci, A. Omicini, M. Viroli, L. Gardelli, and E. Oliva. Cognitive stigmergy: Towards a framework based on agents and artifacts. In Weyns et al. [68], pages 124–140.
 55. A. Ricci, M. Piunti, D. L. Acay, R. H. Bordini, J. F. Hübner, and M. Dastani. Integrating heterogeneous agent programming platforms within artifact-based environments. In L. Padgham, D. C. Parkes, J. Müller, and S. Parsons, editors, *AAMAS (1)*, pages 225–232. IFAAMAS, 2008.
 56. A. Ricci, M. Piunti, and M. Viroli. Environment Programming in MAS – An Artifact-Based Perspective. *Autonomous Agents and Multi-Agent Systems*. To appear.
 57. A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment Programming in CArtAgO. In *Multi-Agent Programming II: Languages, Platforms and Applications, Multiagent Systems, Artificial Societies, and Simulated Organizations*. 2009.
 58. A. Ricci, M. Viroli, and A. Omicini. The A&A programming model and technology for developing agent environments in mas. In M. Mehdi Dastani, A. El Fallah-Seghrouchni, A. Ricci, and M. Winikoff, editors, *PROMAS*, volume 4908 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2007.
 59. Y. Shoham. Agent-Oriented Programming. *Artif. Intell.*, 60(1):51–92, 1993.
 60. M. P. Singh. A social semantics for agent communication languages. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*, volume 1916 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2000.
 61. M. S. Singh and A. K. Chopra. Correctness Properties for Multiagent Systems. In M. Baldoni, J. Bentahar, M. B. van Riemsdijk, and J. Lloyd, editors, *DALT*, volume 5948 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2009.
 62. H. Suguri, E. Kodama, M. Miyazaki, H. Nunokawa, and S. Noguchi. Implementation of FIPA Ontology Service. In *Workshop on Ontologies in Agent Systems, OAS'01, Proceedings*, 2001.
 63. C. Vecchiola, A. Grosso, and A. Boccalatte. AgentService: a framework to develop distributed multi-agent systems. *Int. J. Agent-Oriented Software Engineering*, 2(3):290 – 323, 2008.
 64. M. Venkatraman and M. P. Singh. Verifying compliance with commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 2(3), 1999.
 65. M. Verdicchio and M. Colombetti. Communication Languages for Multiagent Systems. *Computational Intelligence*, 25(2):136–159, 2009.

66. M. Viroli, T. Holvoet, A. Ricci, K. Schelfhout, and F. Zambonelli. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):49–60, 2007.
67. D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, 2007.
68. D. Weyns, H. Van Dyke Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems III, Third International Workshop, E4MAS 2006, Hakodate, Japan, May 8, 2006, Selected Revised and Invited Papers*, volume 4389 of *Lecture Notes in Computer Science*. Springer, 2007.
69. P. Yolum and M. P. Singh. Commitment machines. In *Proc. of ATAL*, pages 235–247, 2001.
70. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.

Contextual Integrity and Privacy Enforcing Norms for Virtual Communities

Yann Krupa, Laurent Vercouter

Laboratory for Information Science and Technology (LIST),
ISCOD team
École des Mines de Saint-Étienne
Saint-Étienne, France
{krupa,vercouter}@emse.fr

Abstract. Contextual Integrity has been proposed to define privacy in an unusual way. Most approaches take into account a sensitivity level or a “privacy circle”: the information is said to be private or public and to be constrained to a given group of agents, *e.g.* “my friends”. In the opposite, Contextual Integrity states that any information transmission can trigger a privacy violation depending on the context of the transmission. We use this theory to describe a framework that one can use in an open and decentralised virtual community to handle privacy in a socially enforced way. This paper describes a framework, in which we can formally describe privacy constraints, that are used to detect privacy violations according to the Contextual Integrity theory. This framework is part of an ongoing work aiming at applying social control to agents that handle the information, so that malicious agents are excluded from the system.

1 Introduction

Most of the works on privacy focus on security as a means of preserving privacy, either by using a central authority that controls the information access[2, 3], cryptography[11], or by using trusted computing techniques[8, 6]. Some other views[9, 4] aim at designing some preferences that users can attach to the data they “own” without taking into account the possibility of deception by other agents.

While a central authority may be a good solution for a range of applications, it is not possible when working in a decentralized and open system with autonomous agents. Therefore, solutions like Purpose Based Access Control[3] cannot be applied.

One of the problems with Digital Right Management and Trusted Computing measures in general, is that they are very constraining. They impose the use of heavy infrastructure or limit the possibilities of information exchange. These constraints, if they are unacceptable for the users, lead them to interact outside the system that is provided, making every implemented security feature inefficient.

Social regulation is another approach where it is physically possible that violations occur in the system. However, users are observed by the society (usually

their neighbours) that can spot them and socially exclude them by ostracism if they commit violations.

So far, very few works consider privacy preserving under the social angle. Yet, it is a prominent problem in applications such as social networks, virtual communities and multi-agent systems, where a social framework will cope naturally with all the social components already working in these systems like trust, reputation, roles for exemple.

Our work tackles the problem of privacy by using social control in decentralized and open multiagent systems. It is based on Nissenbaum’s “Contextual Integrity” theory[7] which defines privacy in a socially relevant way. Therefore it is possible to assess privacy violations from the agent point of view, and apply a social control relying on available social mechanisms, such as the use of trust management techniques, to prevent further violations. Privacy violations will be reduced without requiring a central authority or invasive security measures.

Contextual Integrity states that any information, as inoffensive as it could seem, can potentially harm a set of agents. It means that Contextual Integrity does not make assessment towards the degree of sensitivity of a given information. All information is regarded as evenly sensitive/insensitive. We call the set of agents that can be harmed by an information, its *Targets*. We say that an agent is *harmed* by an information if it makes the agent lose any kind of resource *e.g.* time, reputation, acquaintances, role. An agent sending information is called *Propagator* and the agent receiving the information is called *Receiver*. During the different moments of the process and depending on the information, those attributions may change from one agent to another.

The goal of our work is to provide means to a propagator to use logical reasoning and trust mechanisms to make assessments about a further transmission: “will the transmission of information i to agent z be a violation of contextual integrity?”. A receiver should also be able to do the same process when receiving an information: “was the reception of information i from agent a a violation?”. If a violation is detected, social sanctions are thrown against the violating agents.

This paper describes this ongoing work, it proposes a framework in which we can formally describe privacy constraints according to the Contextual Integrity theory and norms in order to enforce these constraints. This framework is then used to detect the occurrence of privacy violations. The sequel of this article is organized as follows. Section 2 presents Nissenbaum’s Contextual Integrity theory and how we interpret it to build appropriateness laws. The characteristics of the application that we consider, virtual communities, is described in section 3 and it is formally described in order to be able to detect automatically privacy violations. Then, a set of privacy enforcing norms are defined in order to give a roadmap for agent’s behavior in section 4. Finally, section 5 shows how these social mechanisms are used to prevent and punish privacy violations on a sample application, and we conclude the paper in section 6.

2 Contextual Integrity

In this section we present the theory of Contextual Integrity[7] and introduce the concept of appropriateness extracted from this theory.

2.1 Original Works

In some approaches[8] privacy is viewed as binary (either the information is private or not). Other models consider different levels of privacy[2] or circles[4], whereas contextual integrity focuses on the appropriateness of a transmission, or use of information. Every information is potentially sensitive.

In order to have a complete description of the foundations of the theory, the reader should refer to the original article[7]. Here we will only focus our work on the concept of “violation”. Nissenbaum says that “whether a particular action is determined a violation of privacy is a function of :

1. the nature of the situation/context
2. nature of the information with regard to the context
3. roles of agents receiving the information
4. relation of agents to information subject
5. terms of dissemination defined by the subject”

Those ideas are way too vague to be used as-is in a software application, we define hereafter a more precise definition of the concepts.

2.2 Appropriateness

We use the term appropriateness to define the set of laws that makes a transmission inappropriate (*i.e.* will trigger a violation of privacy) if one of these laws is violated. The term “Appropriateness” is inspired by[1].

We use the term “target” instead of Nissenbaum’s term “subject”. A subject is directly related to the information, while a target may not even appear in the information. For example, if the information is a picture of Mr X’s dog doing bad things on X’s neighbour’s grass, the subject is the dog but the target, the one that can be harmed by the disclosure of the picture, is X. Therefore, we think that considering the target instead of the subject is more versatile.

We can then define a flow as appropriate if all of the following conditions hold, and inappropriate if one of the conditions does not hold (numbers in parenthesis refers to the corresponding statement in Nissenbaum’s definition above):

1. Transmission context must correspond to the information nature (1+2),
2. Agent must have a role within the transmission context (3),
3. Agents must not have incompatible relations with target¹ (4),
4. The target’s preferences must be respected (5)

If a flow is inappropriate then there is a privacy violation. Here we can see the point of this approach: an information is not “public” or “private”, every information can trigger a privacy violation if it is used inappropriately.

Thereafter, we illustrate the 4 statements of appropriateness with examples:

¹ This item is a work in progress and is not taken into account in the following parts.

1. In the large sense, the context of a transmission can be seen as the environment where and when the transmission takes place. In our framework, for simplification means, we will say that the context of a transmission is declared by the propagator. A context corresponds to the information if it reflects the nature of the information, *e.g.*: personal health information corresponds to medical context.
2. Agents participating in the transaction should have a role associated to this context[1]. For example, a medical doctor has a role belonging to the medical context.
3. Sometimes, it is not sufficient that the agent has some roles belonging to the context of the transmission. Because some relations between the target of the information and the agent receiving the information may be inappropriate. For example, consider the case of an agent A who has an illness, and an agent B who is both a medical doctor and A's boss. It may be inappropriate for B to know A's disease because those agents are having an "out of context" relationship (hierarchical relationship).
4. If one of the targets of the information specifies preferences regarding the propagation of the information, it is inappropriate to violate those preferences.

As appropriateness has been defined it is now necessary to define the kind of application we consider, information transmission in virtual communities. Afterwards, we propose a formalism of appropriateness to be used in this kind of application.

3 Framework

This section presents the application domain and all the components needed to handle contextual integrity as defined in the previous section, as well as a formalism of appropriateness.

3.1 Privacy Preservation in Virtual Communities

In several types of virtual communities, such as social networks or virtual enterprises², users communicate and share information using software systems that support the community. These applications raise a difficult problem of privacy preservation. On the one hand, it is the main goal of these communities to enable communication so that users can easily send information to their contacts. On the other hand, as it is stated by the contextual integrity theory, each piece of communicated information may result in a privacy violation. Indeed, if we consider the case of a virtual enterprise, the community includes users with different hierarchical roles, belonging to different services but also different enterprises. It is obvious that all information should not be sent to other users without analysing the nature of information and of the concerned users. The same case

² A virtual enterprise is a temporary collaborative network of enterprises made in order to share resources and competences.

occurs in professional or personal social networks in which users' contacts can be her colleagues, siblings, friends.

The goal of our work is to specify a software assistant agent that is able to help a user to preserve privacy in a virtual community. The assistance is both to preserve the user's privacy by providing advices when an information is communicated (should he send this information or not to a given contact?), and to preserve the other users' privacy by detecting when a privacy violation occurred and should be punished. This paper describes the first steps of this ongoing work by defining a language to express privacy constraints and means to detect privacy violations.

The virtual community that we consider has the following characteristics. It works as a peer-to-peer network, meaning that information is exchanged by a communication between one sender and one receiver. Moreover, it is a decentralized and open system. It is thus impossible to define a centralized control that relies on a global and complete perception of communications. We have chosen a system with these features to be as general as possible. By proposing a local assistance to users, the assistant agent can be used both in centralized and decentralized systems and it does not constrain the system scalability. The choice of peer-to-peer communication is also general enough to be able to represent other kinds of communications. For instance, if we want to consider a social network in which information is exchanged by publishing it on a page or a "wall" readable by the user's contacts, it can be represented by several one-to-one communications.

In order to be able to define privacy preservation according to contextual integrity, we need to introduce two concepts in the virtual community: **context** and **role**. The context describes the situation in which an information is exchanged. Examples of context are "Dave's work", "John's family", "health". Roles are defined within a context and attached to users. Examples of roles in the three contexts mentioned above are respectively "Dave's boss", "John's father", "medical doctor". There can multiple roles per context. In this paper, we assume that users' roles and their corresponding contexts are provided by organisational entities that act as repositories. These entities are able to return the role associated to a specific user and the context associated with a specific role. For this purpose, it is possible to use organisational multiagent infrastructures[5].

These concepts are useful to be able to express rather fine rules for Contextual Integrity. We use them in the next subsections to allow the assistant agent to reason on privacy violations.

3.2 Message Structure

Users exchange **information** encapsulated in a **message**. Information is raw data. We don't make assessment about the structure of the information and leave it free. A message encapsulates information plus meta-information described below.

First, from a given information, can be computed a unique reference that allows to refer unambiguously to the information without carrying itself the information (Hash algorithms like Message Digest[10] can be used).

Then, the message adds the following meta-information:

- Context Tags: tags referring to the context of the information
- Target Tags: tags referring to the targets of the information
- Privacy Policies: policies expressing preferences regarding further distribution of information
- Transmission Chain: a chain of transmissions that allows to keep track of the message path in the system

Each of these components may be digitally signed by agents that wish to support the meta-information accountability. When signing a meta-information an agent engages his responsibility. The semantics that relies behind the signature is a certification: *i.e.* the agent that signs the context tag “medical context” certifies that the information is about medical context. Therefore, it is very important that a meta-information, even if it can be detached from the information (which is possible), cannot be reattached to another information. We prevent that from happening by including the information hash before signing. Signatures are formed by a name and a signature (RSA signature for example[11]). The transmission chain allows to keep track of the message path among the agents. Every agent is required to sign the chain before propagating a message, an agent adds his signature including his own name and the name of the receiver of the message.

3.3 Primitives

To allow the agent to recover data regarding the concepts described earlier, like the meta-information or the roles of agents, we need to provide the agents a set of logical primitives. These primitives can then be used to express constraints about transmission of information.

1. Primitives based on meta-information:
 - `information(M,I)`. Means that I is the information³ encapsulated in message M.
 - `contexttag(C,A,M)`. Means that C is the context tag for message M signed by agent A.
 - `targettag(T,A,M)`. T is the target tag for message M, signed by A.
 - `policy(P,A,I)`. There is a policy P signed by agent A for information I.
2. Primitives based on transmission roles:
 - `receiver(X,M)`. Agent X is receiving the message M.
 - `propagator(X,M)`. Agent X is sending the message M.
3. Primitives based on agent beliefs:
 - `target(X,I)`. The agent believes that agent X is targeted by the information I.

³ The primitives are referring to an information I or a message M. This is because some primitives will be specific to a given message M, and some others will be common to all messages containing the same piece of information I.

- `policyvalid(P,I)`. The agent believes that the preferences expressed by policy P are respected for the information I.
- `context(C,I)`. Means that the agent believes that C is the context of information I
- `role(A,R)`. The agent believes that Agent A has the role R.
- `rolecontext(R,C)`. The agent believes that role R belongs to context C (role “surgeon” belongs to Medical context).
- `link(X,Y)`. The agent believes that agent X is capable of communicating with Y.

Now, based on this primitives, we are able to express preferences or norms.

3.4 Appropriateness Laws

Our goal is to obtain some simple laws that agents can rely on to be able to decide if a given transmission of information should be seen as a violation or not.

These appropriateness laws are thereafter abbreviated as A-laws.

This is the definition of the A-laws we propose in Prolog-like code:

- Context declared by the propagator must be equal to the information context (Fig. 1).
- Receiver must have a role within the transmission context (Fig. 2).

```
fitcontext(C,M):-
    information(M,I),
    propagator(P,M),
    context(C,I),
    contexttag(C,P,M).
```

Fig. 1. fitcontext

```
fitrole(C,M):-
    receiver(Rc,M),
    role(Rc,R),
    rolecontext(R,C).
```

Fig. 2. fitrole

- The target’s preferences must be respected:
 - In the case there is no⁴ policy defined by a target then `fitpolicy(M)` holds (Fig. 3).
 - If there is a policy defined by the target, the agent must respect it (Fig. 4).

Therefore, a transmission is defined as appropriate for a message M if the following formula holds:

```
appropriate(M):-
    fitcontext(C,M),
    fitrole(C,M),
    fitpolicy(M).
```

If the definition above does not hold, then we can say that the transmission is inappropriate, there is a violation of the contextual integrity.

⁴ \+ is the negation-as-failure in Prolog.

```

fitpolicy(M):-
  information(M,I),
  \+ (
    policy(P,T,I),
    target(T,I)
  ).

```

Fig. 3. fitpolicy (when no policy is defined)

```

fitpolicy(M):-
  information(M,I),
  policy(P,T,I),
  target(T,I),
  policyvalid(P,I).

```

Fig. 4. fitpolicy (when a policy exists)

3.5 Policies

The A-laws define what is appropriate or not in a general point of view, but targets can define policies (preferences) in order to constrain the information. These preferences are defined for a given information by a given agent who signs the policy. In the system, it is not possible to insure that a policy cannot be detached from the information it is referring to, *i.e.* an agent may erase the policy at some point. But it is possible to reattach a policy to another information, because the policy is signed, and contains a pointer to the information it refers to.

A **policy** is composed by several **statements**. A **statement** is composed by several **primitives** from the ones described in section 3.3 and by a type of statement that can be:

- **forbidden(I)**:-
Declares a situation that should not occur within a transmission of information I.
- **mandatory(I)**:-
Declares a situation that has to occur within a transmission of information I.

A given policy is fulfilled if none of its forbidden statements holds (if one holds, then it is unfulfilled) and one of its mandatory statements holds^[1]⁵.

An example of policy for a given information identified by 'info99' is given below. It is composed by two forbidden statements (do not send data to an agent who has a common contact with the target AND don't send data to the target) and one empty mandatory statement.

```

forbidden(info99):-
  information(M,info99),
  receiver(X,M),
  target(T,info99),
  link(X,Z),
  link(Z,T).

forbidden(info99):-
  information(M,info99),
  receiver(X,M),
  target(X).

mandatory(info99).

```

⁵ A statement is composed by a conjunction of primitives, therefore the disjunction is expressed by defining multiple statements of the same kind. This is why only one mandatory statement is required to validate the policy and one forbidden to invalidate it.

In order to test the primitive `policyvalid(P,I)`, an agent adds to his memory all the statements contained in policy P (we suppose here that we have a primitive `addpolicy(P)` to do just that):

```
policyvalid(P,I):-
  addpolicy(P),
  \+ forbidden(I),
  mandatory(I).
```

4 Privacy Enforcing Norms

As shown in the previous sections, we need the agents to check the transmissions, to be able to see if there are violations and punish the responsables. This section propose a set of norms that defines what should be the behavior of a compliant agent in the system. Then it describes the puniton mechanisms and finally discusses the inherent problems regarding the subjectivity of beliefs.

4.1 Definition

The basic component of the system is the set of A-laws, that express Contextual Integrity violation. But the keystone of the system are the Privacy Enforcing Norms (PENs), defined in this section, that instruct the agents to respect the A-laws and punish those who do not.

The PENs are the following :

1. Respect the Appropriateness laws
2. Sign the transmission chain before sending
3. Do not send information to untrusted agents
4. Delete information from violating or untrusted agents
5. Punish agents violating these norms (this one included)

The first norm (PEN 1) that we propose is meant to protect the A-laws from being violated : “Respect the Appropriateness laws”.

From our point of view, every agent must take responsibility when doing a transmission. Thus we define a norm stating that every agent has to sign the transmission chain (in order to backtrack the potential violation to its source). We also consider that sending information to an agent while knowing that he will commit a violation, is a violation itself. Two new norms are then defined : “Sign the transmission chain before sending (PEN 2) ; Do not send information to untrusted agents (PEN 3).” The PEN 3 also implements the social punishment, because agents will stop communicating with these untrusted agents.

The fourth norm aims at minimizing the violations by deleting information received from unreliable agents (PEN 4).

Norms that the agents should respect have been defined, but we want to be sure that the agents in the system will punish those who do not respect the norms, henceforth punishing those that do not punish agents not respecting the norms. This last norm (PEN 5) insures consistency of the PENs, because an

agent that decides to violate a norm will be punished, others will stop trusting him and eventually he will become socially isolated.

Therefore norms are not enforced by the system but by the agents themselves and agents refusing to enforce the norms will be punished by other agents. For now, the punishment is implemented as a social punishment: an agent witnessing a violation has to send a message to all of its contacts stating the details of this violation. The following section gives more details about this punishment mechanism.

4.2 Punishment

When an agent detects a violation of the PENs, PEN 5 states that he has to send a punishment message. This message is meant to describe the violation so that other agents can punish the culprit. The message has the same structure than all the messages in the system: information and meta-information. Here the information part contains:

- The meta-information of the original message source of the violation
- A description of the violation using the primitives of section 3.3

Sending the meta-information of the original message is useful to provide evidence to other agents that may not believe that there was a violation. The advantage of sending only the meta-information is that the agent will not transmit the information itself (which could in turn trigger a violation and so on).

The violation is described using the primitives, and the PEN that has been violated. For instance, if the PEN 3 has been violated by Bob, the following primitives will be sent:

```
pen3violation(Bob,mess45),
receiver(John,mess45),
propagator(Bob,mess45),
untrustworthy(John).
```

These primitives will be handled and verified by the receiving agent. If the agent agrees with every primitive in the argument, then he can propagate the punishment message and punish the culprit by revising its trust. There are situations where the agents may not have the same beliefs, *e.g.* John may or may not be trustworthy depending on the agent making the assessment.

4.3 Discussions on Subjectivity

Some of the PENs are very subjective, because they are based on beliefs. Therefore 2 given agents in the system may not have the same belief and interpret the norms differently. For instance, two agents, *A* and *B* may have different beliefs regarding agent *X* trustworthiness: *A* trusts *X* but *B* does not. Now *A* sends a message to *X* who in turn sends the message to *B*. In the transmission chain, *B* is able to see that the transmission occurred between *A* and *X*, which violates norm 3. Going back from *A* point of view, it would not be fair to be punished for this transmission as *X* seems trustworthy for him.

B witnessed a violation so he has to send a punishment message. The punishment message has to argue about the punishment. More than just saying “*A* does not respect the norms”, *B* makes a message stating that “*A* violated the third norm because *B* believes that *X* is untrustworthy and *A* sent a message to *X*”. The agent receiving this violation message is going to check these statements and if he agrees, he can revise his trust level towards *A*.

Along with the violation description, the punishment message contains the meta-information of the original message. This allows other agents to check the PENs and violation description. For instance, it will allow agents to check that *A* did send the information to *X* by looking at the transmission chain contained in the meta-information.

4.4 Usage

This section describes how the agents are meant to protect privacy using the tools provided in the previous sections. As it is said in the introduction, our goal here is to handle privacy from the agent perspective to minimise the number of violations in the whole system.

There will be two main situations:

- Receiving: When the agent receives an information: “Does the agent that sent me this information made a violation by sending it to me?”
- Propagating: When the agent is about to send information: “Am I going to make a violation if I send the information to this agent?”

Trust In the framework presented in this article, agents may perceive things differently. If we take a closer look at the primitive `context(C,I)` described earlier, for instance, it is stated that it means that the “agent **believes** that *C* is the context of information *I*”. Therefore, some agent *X* may believe for a given information that the context is *O*, and another agent *Y* may believe that the context is *P*. This situation can happen because the agents are autonomous and have beliefs that can be different from one to another. As they have different beliefs, some agent may think that a given transmission is inappropriate, and another may think that it is not. Because of this uncertainty, when an agent detects a violation, he is not able to be sure that the other agent made the violation on purpose, therefore it will be unfair to kick him directly from the system. This is where trust comes in, this kind of “soft security” is able to cope with detection errors while still being able to exclude the ones that make violations. Trust is one of the main components to decide who is reliable or not for handling our information. If someone is untrustworthy, we are not willing to send him any piece of information.

The trust management component of agents is not yet implemented and is being defined in our current ongoing work. We will probably use an adaptation of existing computational trust models for multi-agent systems such as LIAR[13] or Repage[12].

Receiving When the agent is receiving a message, he has to check if the transmission that just occurred is a PEN violation or not. First, the agent has to check the A-laws to see if the transmission is appropriate (PEN 1), as described in section 2.2. To do that, the agent will have to infer multiple things, for example: who is the target of the message? what is the context of the message? This is possible either by using personal knowledge, by using the context tags and target tags or by analysing the information directly. As the context tags (and target tags) are signed, it is possible to trust the agent that signed the given tag, to come to believe that this context tag corresponds to the context of the information.

If the agent detects a PEN violation, he sends a “punishment message” to other agents.

Finally, the agent readjusts the trust level he has towards the propagator that just made the violation.

Propagating This second situation happens when the agent is about to send information. Before sending, it is necessary to attach to the information all possible meta-information:

- If the agent can identify the target of the information (by using knowledge or information analysis), he adds a target tag for target Z that he signs. This states that the agent confirms that the target of the information is Z .
- If the agent is able to determine the context of the information (by using knowledge or information analysis), he adds and signs a context tag.
- If the agent is the target, he can specify some restrictions regarding further dissemination of the information, in this case, he adds a policy that he signs.
- The agent also signs the transmission chain to insure PEN 2.

Then, the agent should make all PEN assessments towards the receiver:

- Does the agent violate the A-laws (PEN 1) by sending the information to the receiver? An agent never violates A-laws, except if he is malevolent or ignorant, which in both cases, will be punished by other agents.
- Does the agent trust the receiver? (PEN 3) If he is untrustworthy, it means that he has probably made some privacy violations in the past. As the agent aims at protecting the information he holds, he only sends to the ones he trusts.
- And so on with the other PENs.

At the end, the agents send information from one to another, checking before sending and after receiving if some violation has occurred. When violations are detected, agents send “punishment messages” to their contacts, so that others become aware of the violation that occurred. Eventually, agents that make violations will be socially excluded from the system, because no agents communicate with untrustworthy agents.

5 Sample Application

Our aim here is to define a sample application to show how all the framework components instantiate on this application.

5.1 Photo Sharing

The application that we consider here is a photo sharing social network. Basically, users can share pictures with their contacts who can, in turn, share again those pictures with their own contacts and so on. We provide the users with an assistant agent that will do all the assessment described before to inform the user of any violation. The final decisions lies in the hands of the user, the assistant does not take any decision.

In this system, the pictures are the information that is exchanged.

5.2 Primitives Instantiation

Some of the primitives we defined earlier need to be specified for this application. The primitives based on meta-information always remain the same, because the nature of the meta-information does not change. So do the primitives for transmission roles.

We can explain in more detail the primitives based on agent beliefs because the way they are inferred is what is interesting here:

- **context(C,I)** For the agent to believe that *C* is the context of information *I*, there are alternative solutions:
 - Look if there is a context tag emitted by a trusted agent
 - Analyse the picture to find its context (using image analysis techniques)
 - Ask the user attached to the agent to determine the context of the picture
- **target(X,I)** The same process can be used for the agent to believe that *X* is the target of *I*:
 - Look if there is a target tag emitted by a trusted agent
 - Analyse the picture to find if the target is on the picture
 - Ask the user attached to the agent to determine the target of the picture
- **link(X,Y)** By analysing the transmission chain in the meta-information, the agent can discover links between other agents.
- **knows(X,I)** Using the same technique, the agent can extract from the transmission chain the list of agents that received the information in the past.
- **role(A,R)** The agent asks the organisational infrastructure to know the possible roles of *A*.
- **rolecontext(R,C)** The agent asks the organisational infrastructure to know the possible roles fitting in context *C*.
- **policyvalid(P,I)** The agent infers on his belief base to see if the policy is valid as explained in section 3.5.

With the primitives instantiated, it is easy to check the policies, the A-laws and all needed components. In the next section we show an example of what happens in the application.

5.3 Use Case

Alice wants to share a picture with Bob. The target of the information is James, who is in an awkward position on the picture. Some of James' friends already had this information before, therefore, there are tags describing the context as "James friends" and the target as "James". No policy has been attached. The unique identifier of the information is "pic254". The message is identified by "mess412".

When Alice clicks on the button to send the picture to Bob, the assistant agent checks the PENs:

- PEN 1: Does the agent violates the A-laws by sending the information to the receiver? This is the instantiation of the laws described in section 3.4:

- The declared context is set by the agent, so the declared context fits the context the agent believes to be the real one, the following formula holds:

```
fitcontext('James friends',mess412):-
    information(mess412,pic254),
    propagator('Alice',mess412),
    context('James friends',pic254),
    contexttag('James friends','Alice',mess412).
```

- The assistant agent is not able to find a role for Bob that fits into the context "James friends", the formula does not hold:

```
fitrole('James friends',mess412):-
    receiver('Bob',mess412),
    role('Bob',?),
    rolecontext(?, 'James friends').
```

- No policies were defined, therefore, the first `fitpolicy(M)` statement holds (no policy exists for none of the target of the information).

The following Appropriateness formula does not hold, because Bob is not a friend of James (the target):

```
appropriate(mess412):-
    fitcontext('James friends',mess412),
    fitrole('James friends',mess412),
    fitpolicy(mess412).
```

Beyond this point, the assistant agent knows that the transmission will be inappropriate, and therefore violates the PENs. Anyway, he asks the user (Alice), what to do: continue or abort the transmission?

Alice wants to continue. The message containing the picture and meta-information is sent to Bob.

Bob's agent handles the information by checking the PENs:

- Does the message violates contextual integrity? Bob's agent runs here the same test that Alice's agent did (using his own beliefs). As Bob is not a friend of James, no roles fits in the context "James friends" and a violation is therefore detected.

Bob’s agent adjusts his beliefs, he does not trust Alice anymore because it is not the first time that Alice deceives Bob. He sends to all his contacts a “punishment message” containing the meta-information (context tags, target tags, transmission chain) and the following description:

```
pen1violation(Alice,mess412),
information(mess412,pic254),
context(C,mess412),
\+ fitrole(C,mess412).
```

Dave’s agent is one among those who receives this message. Dave was about to send a message to Alice, when he clicks the “send” button, his agent checks the PENs. Then PEN 3 forbids to send a message to an untrusted partner. Dave’s agent warns him that Alice is untrustworthy and that the transmission will violate the PENs.

Users stop communicating with Alice because of the violation she made. Alice is now socially excluded, she is yet still in the system but nobody keeps communicating with her.

The example is a little bit hard on Alice in order to show the power of social exclusion. Normally, it will take multiple violations for someone to be excluded from the system and forgiveness could occur after a certain time.

6 Conclusions

The framework we presented in this article allows to protect users privacy in a decentralised and open system when it is not possible to apply computer security approaches. Based on Nissenbaum’s Contextual Integrity theory, we propose an approach using appropriateness laws that defines what is an appropriate information transmission (and therefore, what is inappropriate). Primitives are defined to express these laws and to allow agents to define preferences over the transmission of some specific information.

Agents in the system play both roles of actors and judge: they transmit information, and they detect violations. Agents also inform others when they spot a violation, so that the violating agents are excluded of the system. This behavior is directed by the Privacy Enforcing Norms (PEN) described in this article.

There are still some unsolved problems in the system, that may prevent it from working correctly:

- The trust related problems: “what happens if there are too many malevolent agents in the system?”
- “Journalist Problem”: “what happens if an agent decides to sacrifice himself to become a relay for information that violates privacy?” (the original source is never punished, only the journalist).
- Reputation Paradox: Information about reputation is libellous in a way, so it can generate privacy violation. But at the same time, it is required for maintaining information regarding agents that make violations.

In our future works, we will integrate the trust mechanisms directly in the decision process, *i.e.* decompose the primitives that rely on trust in predicates. We will, at the same time, investigate the problems related to the subjectivity and related to strategic manipulation (agents sending fake violation messages for example). Then an application will be developed to probe the system in the real world by providing assistant agents to users.

References

1. Barth, A., Datta, A., Mitchell, J., Nissenbaum, H.: Privacy and Contextual Integrity: Framework and Applications. 2006 IEEE Symposium on Security and Privacy (S&P'06) pp. 184–198, <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1624011>
2. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Tech. rep., Technical Report MTR-2547 (1973)
3. Byun, J., Bertino, E., Li, N.: Purpose based access control of complex data for privacy protection. In: Proceedings of the tenth ACM symposium on Access control models and technologies. p. 110. ACM (2005)
4. Crépin, L.: Les Systèmes Multi-Agents Hippocratiques. Ph.D. thesis (2009)
5. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems* 20(3), 369–400 (mai 2009), <http://www.springerlink.com/content/g115t233633v6h16>
6. Mont, M.C., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In: *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*. pp. 377–382 (2003)
7. Nissenbaum, H.: Privacy as Contextual Integrity. *Washington Law Review* pp. 101–139 (2004)
8. Piolle, G.: Agents utilisateurs pour la protection des données personnelles: modélisation logique et outils informatiques (2009)
9. Reagle, J., Cranor, L.F.: The platform for privacy preferences. *Communications of the ACM* 42(2), 48–55 (1999)
10. Rivest, R.: The MD5 Message-Digest Algorithm. *Distribution* (1992)
11. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public- Key Cryptosystems. *Communications* 21(2) (1978)
12. Sabater, J., Paolucci, M., Conte, R.: Repage: REPutation and ImAGE Among Limited Autonomous Partners. *Journal of Artificial Societies and Social Simulation* 9(2), 3 (2006)
13. Vercouter, L., Muller, G.: L.i.a.r.: Achieving social control in open and decentralised multi-agent systems. *Applied Artificial Intelligence* (2010), to appear



**IEEE FIPA Workshop on Design Process
Documentation and Fragmentation**

Co-located with the MALLOW 2010 Conference

Domaine Valpré, Lyon (France)
30-31, August 2010

Conference Organization

Programme Chairs

Massimo Cossentino Vincent Hilaire Ambra Moselini

Programme Committee

Estefania Argente Villaplana, Carole Bernon, Vicent Botti, Giacomo Cabri, Scott DeLoach, Giancarlo Fortino, Ruben Fuentes-Fernandez, Stephane Galland, Alfredo Garro, Nicolas Gaud, Paolo Giorgini, Alma Gomez, Juan Carlos Gonzalez Moreno, Zahia Guessoum, Marc-Philippe Huget, Renato Levy, Gleizes Marie-Pierre, Frederic Migeon, Vito Morreale, Andrea Omicini, Sascha Ossowski, Juan Pavon, Joaquin Pena, Anna Perini, Jolita Ralyte, Wolfgang Renz Luca Sabatucci, Valeria Seidita, Alberto Sienna, Pietro Storniolo, Jan Sudeikat, Angelo Susi, Kuldar Taveter, Juha-Pekka Tolvanen, Inge van de Weerd.

External Reviewers

Mariachiara Puviani

Table of Contents

Towards a New Approach for MAS Situational Method Engineering: a Fragment Definition	3
<i>Sara Casare, Zahia Guessoum, Anarosa Brandão, Jaime Sichman</i>	
A Glimpse of the ASPECS Process documented with the FIPA DPDF Template	17
<i>Massimo Cossentino, Stéphane Galland, Nicolas Gaud, Vincent Hi- laire, Abderrafaa Koukam</i>	
Process Documentation Standardization: An Initial Evaluation	29
<i>Massimo Cossentino, Juan Carlos González Moreno, Alma Gómez Rodríguez, Andrea Omicini, Ambra Molesini</i>	
Describing GORMAS using the FIPA Design Process Documentation and Fragmentation Working Group template	43
<i>Sergio Esparcia, Estefania Argente, Vicente Botti</i>	
The O-MaSE Process: a Standard View	55
<i>Juan C. Garcia-Ojeda, Scott DeLoach</i>	
Applying Process Document Standarization to INGENIAS	67
<i>Juan Carlos González Moreno, Alma Gómez Rodríguez</i>	
Exploring the Boundaries: when Method Fragmentation is not Convenient	79
<i>Chiara Leonardi, Luca Sabatucci, Angelo Susi, Massimo Zancanaro</i>	

Towards a New Approach for MAS Situational Method Engineering: a Fragment Definition

Sara Casare¹, Zahia Guessoum², Anarosa A. F. Brandão¹, Jaime Sichman¹

¹ Intelligent Techniques Laboratory – University of São Paulo - Brazil
{sara.casare, anarosa.brandao, jaime.sichman}@poli.usp.br

² Laboratoire d'Informatique de Paris 6 - LIP6 – University Pierre et Marie Curie - France
zahia.guessoum@lip6.fr

Abstract. This paper introduces a new definition of method fragment intended to represent MAS development approaches in a more standardized and coherent way, thus facilitating the configuration of situational methods. In order to do that, we take into account three complementary notions: (i) a method fragment *description* based on SPEM 2.0 elements; (ii) two method fragment *perspectives*, the internal and the external view, and (iii) four method fragment granularity layers. Moreover, this definition establishes some mechanisms for method fragments' encapsulation and identification. The proposed method fragment definition is illustrated through an example using Tropos.

Keywords: multiagent oriented software engineering, situational method engineering, method fragment, SPEM

1 Introduction

In order to structure the development and to manage the complexity associated with Multiagent Systems (MAS), several development methods have been proposed during the last decade, e.g. Gaia [18], Tropos [2], PASSI [6], and Adelfe [1]. The variety of Agent Oriented Software Engineering (AOSE) methods is due to the specific needs raised on MAS development and to the different approaches adopted by MAS developers. It shows that a method cannot be general enough in order to be applied to every MAS development project without some level of customization [11]. Moreover this customization requires deep knowledge on both the method and the MAS research field. Nevertheless, it seems that reinventing a new method for each new project situation wouldn't be a best practice, given that there are a great number of available methods for MAS development. This scenario suggests that Method Engineering techniques and, particularly, Situational Method Engineering [3] seems to be promising approaches to be considered for MAS development.

Situational Method Engineering is the sub-area of Method Engineering that addresses the controlled, formal and computer-assisted construction of situational methods out of method fragments. Roughly speaking, building a situational method consists of reusing parts of existing methods taking into account a given project situation that encompasses, for example, notions related to the class of the desired application (like traditional and pervasive computing) and project perspectives.

Several approaches concerning the notions of a *part of a method* and *situational method building* have been proposed in the Situational Method Engineering field. For instance, Brinkkemper and colleagues [3][4] introduce a Method Fragment notion, and Karlsson [14] introduces a Method Component notion. Method Fragments [3][4] are standardized building blocks based on a coherent part of a method that can reside on one of five layers of granularity: method, stage, model, diagram or concept. The notion of coherence should be interpreted while considering a method as a connected graph of products or processes. For instance, an entire process can be considered as a method fragment. A situational method can be built by combining a number of method fragments in a bottom-up fashion. Such a combination must follow certain assembly rules in order to adhere to the construction principles into the process perspective and the product perspective.

A Method Component [14] consists of an exchangeable and reusable part of method composed of descriptions for actions, notations, artifacts and concepts that can be viewed into two perspectives: an internal view and an external view. While the internal view presents all method component elements (as action, artifacts, and roles), the external view aims to describe method component output in order to identify how it contributes to a chain of goal achievements. On the one hand, this approach emphasizes principles as method modularization and method reusability. On the other hand, it proposes a way for using these principles in order to define a procedure for method configuration involving the notion of Base Method: a method chosen as starting point for the configuration process, allowing a top-down fashion to create situational methods, eliminating, adding or exchanging additional fragments captured from another method.

This paper proposes a definition of method fragment that combines these two notions of part of a method. This definition allows representing MAS development approaches in a more standardized and coherent way. Moreover, it establishes mechanisms for method fragments encapsulation and identification in order to provide a solid base for developing / building MAS situational methods. The paper is organized in five sections. Section 2 presents the proposed definition for MAS method fragment, while Section 3 shows an application of such definition to Tropos. Section 4 presents an overview of the current research concerning situational method engineering applied to MAS field. Finally, Section 5 presents a discussion about the proposed approach for MAS method fragment definition.

2 A New Definition for MAS Method Fragment

The MAS Method Fragment definition proposed in this paper has been mainly inspired on the Method Fragment notion proposed by Brinkkemper and colleagues [3][4] as well as on the notions of method component view and Base Method proposed by Karlsson [14]. From Brinkkemper and colleagues we adopted the simple and intuitive idea of *part of a method* and from Karlsson we adopted the black box perspective of *part of a method* offered by the Method Component view, as well as the Base Method notion to provide a solid foundation for top-down situational method configuration. Additionally, we have adopted some concepts of Software Engineering proposed by Jacobson and colleagues [13] and have used SPEM 2.0 (Software and

Systems Process Engineering Metamodel) [15] as a common meta-model for describing method fragment. The former is among the most popular software development processes and the latter is the standard “de facto” to model development process.

Our proposed definition is:

“A *MAS Method Fragment* is a **standardized building block** based on a **coherent part of a MAS development approach**”.

The standardization of building blocks considers the notions of (i) identification of method fragments using well established naming rules in order to convey their desired semantics; (ii) encapsulation of original work products; (iii) utilization of common roles for MAS developers to be used as task performers; and (iv) classification of method fragment based on a semiotic criteria [5]. The coherence of method fragments is assured by the notions of (i) a *method fragment description* based on the SPEM 2.0 elements (task, work product, role, activity and so on) and their associations; (ii) the proposition of *two method fragment views* (internal and external views); and (iii) the use of four *method fragment granularity layers* (activity, phase, iteration, process).

In the following subsections, we will describe the main characteristics of the proposed definition for coherence.

2.1 Standardizing Building Blocks

In order to have a standardized and common semantics for specifying method fragment objectives and work products, we have defined a MAS Work Product Framework mainly based on the MAS components proposed in the Vowel approach [9] - **A**gent, **E**nvironment, **I**nteraction, **O**rganization. This approach offers a natural and coherent way for describing MAS components and has been adopted in several MAS research [16] with successful results. Nevertheless, it does not deal with the notion of users requirements that should be gathered before specifying MAS components. Then, the proposed MAS Work Product Framework involves also an element related to MAS User Requirement, in order to encapsulate work products used to describe the system-to-be requirements.

Such a work product framework is used to encapsulate original MAS development work products, explicitly stating their involvement with user requirements or to the main MAS components. This approach allows enhancing work product flow into a situational method and making clear the main goal of each work product independently of their name in the context of the original MAS development approach. Finally, the method fragment characteristics are specified through the MAS Semiotic Taxonomy [5] that provides a set of semiotic criteria to categorize MAS Method Fragments taking into account their meaning, usage, structure and so on.

2.2 MAS Method Fragment Main Elements

The proposed MAS Method Fragment description is based on SPEM 2.0 elements and related associations. To improve readability we use Arial font to concepts proposed in

this work and *Comic Sans* font to describe SPEM elements. Therefore, the main elements used to compose a MAS method fragment description are: *Process Pattern*, *Activity*, *Phase*, *Milestone*, *Iteration*, *Task Definition*, *Task Use*, *Step*, *Role Definition*, *Role Use*, *Work Product Definition*, *Work Product Use*, *Category*, and *Guidance*. As proposed by SPEM 2.0, these elements are separated into method content elements (*Task Definition*, *Step*, *Role Definition*, *Work Product Definition*) and their application in the development process (*Process Pattern*, *Activity*, *Phase*, *Milestone*, *Iteration*, *Task Use*, *Role Use*, *Work Product Use*).

A *Process Pattern* represents building blocks for assembling processes. It describes a reusable cluster of *Activities* that provides a consistent development approach to common problems. An *Activity* represents a general unit of work assignable to specific roles, relying on input work products and producing output work products. We have chosen this as the main element of MAS Method Fragment in the Activity Layer. A *Phase* consists of a significant period in a project, ending with major management checkpoint, as a *Milestone* that represents a significant event for a development project. We have used *Phase* and *Milestone* as main elements of MAS Method Fragment in the Phase Layer. Moreover, *Milestone* is used to define fragments in the Process Layer. An *Iteration* is a set of nested *Activities* that are repeated more than once, allowing the organization of work in repetitive cycles. It has been used to define MAS Method Fragments in the Iteration Layer.

A *Task Definition* represents an assignable unit of work involving generally a few hours to a few days and usually affecting one or only a small number of work products. A *Step* describes a meaningful and consistent part of the overall work described for a *Task Definition*. A *Task Use* represents a proxy for a *Task Definition* in the context of one specific *Activity*. *Tasks* and *Steps* constitute the main element of the *Activities* used to build MAS Method Fragments.

A *Role Definition* describes a set of related skills, competencies, and responsibilities of an individual or a set of individuals. A *Role Use* represents a *Role Definition* in the context of one specific *Activity*. Roles represent both the development roles originally specified by the AOSE methods and the common MAS role set involved in the MAS Method Fragments definition. *Work Product Definition* represents pieces of work that are used, modified, and produced by *Task Definitions*, while a *Work Product Use* represents a *Work Product Definition* in the context of a specific *Activity*. *Work Product Definitions* represent the artifacts proposed by the MAS development approaches, as models, specifications and diagrams.

A *Category* represents the classification structure used to group SPEM elements based on the user's criteria. It allows defining tree-structures of nested categories used for browsing MAS Method Fragments based on a semiotic criteria [5]. A *Guidance* represents a specific description related to other SPEM elements. It can be a formal description, such as concepts description, or informal description such as guidelines, white papers, checklists, examples or roadmaps. *Guidance* represents some elements proposed by the MAS development approaches, such as work product examples, main references, concepts and tool mentor.

The aforementioned elements will be depicted in conjunction of the proposed method fragment views and some standardization notions in Fig. 1. Finally, we have applied an important notion proposed by SPEM - called **Variability Elements** - to improve tasks and work products defined according to the original MAS development approaches. It allows reaching the completeness and standardization involved in a MAS Method Fragment definition without modifying the original method elements. For example, we can use a **Task Variability** to complete a task definition introducing performing role and a **Work Product Variability** to define composite work products, as Agent Model or Requirement Model.

2.3 Method Fragment Views

The *Internal* and *External* views of MAS Method Fragments depict, respectively, the whole set of elements that compose them and the main elements that constitute their interface. They have been inspired on the method component views concept proposed by Karlsson [14]. The Internal View offers a detailed and deep representation of a method fragment that allows analyzing all elements involved in its composition, such as activities, roles, tasks, guidance, categories, work products and milestone, as well as their relationships (see Fig. 1 for details).

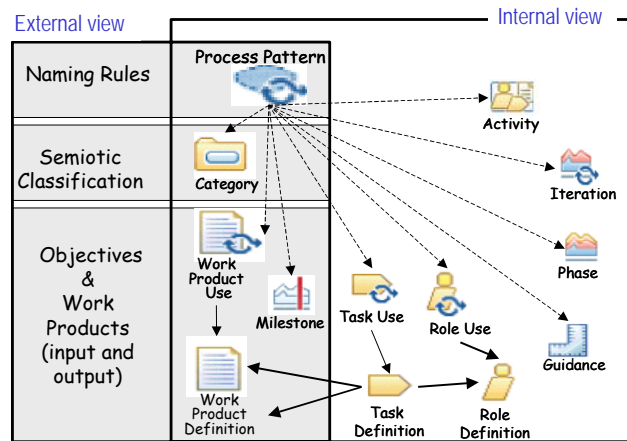


Fig. 1 Main elements of MAS Method Fragment

The External View goals are twofold. First, it describes, in a standard way how a method fragment can be used in a situational method configuration by: (i) identifying the method fragment through naming rules, (ii) specifying method fragment objectives in terms of milestones and/or work products, and (iii) describing fragment characteristics as meaning, usage, structure and so on. Second, it describes how a method fragment can contribute to achieve a situational method objective, specifying fragment output work products through a common semantic. The naming

rules adopted to identify method fragments are based on simple concepts and will be explained through the method fragments examples in the next section.

These two method fragment perspectives improve method fragment coherence because it allows analyzing method fragments as standard black boxes without losing the details of the description. **Fig. 1** shows the views and associated elements in a diagrammatic perspective. For instance, it depicts **Process Pattern** as a kind of “logical container” for building MAS Method Fragments, **Category** to define part of the External View of MAS Method Fragment, called Method Fragment Semiotic Classification and the use of **Milestone** to represent MAS Method Fragment expected objectives.

2.4 Method Fragment Layers

The four layers of MAS Method Fragment – activity, iteration, phase and process - have been defined according to Jacobson et al [13] and SPEM homonym concepts.

The definition of a MAS method fragment in the Activity Layer, for short an Activity Method Fragment, is based on the notion of Activity proposed by Jacobson et al [13]. An Activity Method Fragment consists of a tangible unit of work performed by a worker that yields a well-defined result based on a input set of artifacts. The unit of work has defined boundaries that are likely to be referred in a project plan when tasks are assigned to individuals.

Fig. 2 depicts the components of an Activity Method Fragment. It is worthy to notice that this figure represents only the main relationships between SPEM elements used to define an Activity Method Fragment. For instance, relationships between **Category** and **Roles**, **Tasks** and **Work Products** are not depicted. Moreover, we have labeled the relationship arrows with cardinalities that constitute a constraint over SPEM elements definition. For example, in general a **Process Pattern** can be associate to zero or many **Activities**, while the **Process Pattern** used in the context of an Activity Method Fragment must be associated with exactly one **Activity**.

An Activity Method Fragment is composed of one **Process Pattern** associated with one **Activity** and one or more **Categories**. Such **Activity** must be associated with at least one **Task Use** that must produce one or more **Work Products** as output. However, in the context of an Activity Method Fragment an **Activity** must not be associated with an **Iteration** nor to a **Phase** or **Milestone** elements, given that these elements are used to define other layers of method fragment. As we can see in Fig. 2, **Category** and **Work Product** elements constitute the external view of the Activity Method Fragment. In summary, an Activity Method Fragment must contain one **Activity** that is composed by at least one **Task**, one **Role** and one output **Work Product**. Moreover, it must be classified by **Categories**.

The definition of a MAS method fragment in the Phase layer, for short a Phase Method Fragment, is based on the notion of phase proposed by Jacobson et al [13]. These authors consider that a phase should be concluded with a major milestone. Moreover, they state that any software process needs to have a sequence of clearly articulated milestone in order to be effective. Therefore, a Phase Method Fragment represents a significant period in a project and consists of a **Process Pattern**

classified by *Categories*, associated with exactly one *Phase*, one (major) *Milestone* and several *Activity Method Fragments* and/or *Iteration Method Fragments*.

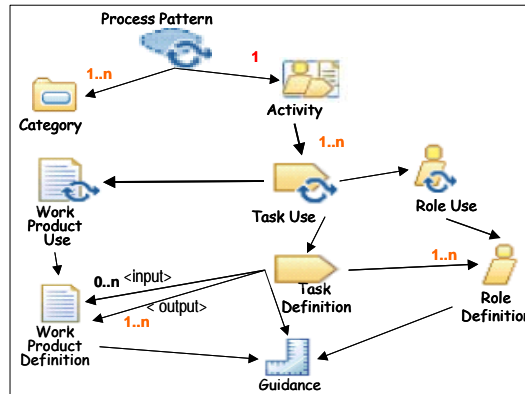


Fig. 2: Activity Method Fragment representation as UML Class Diagram

The definition of a MAS method fragment in the Iteration layer, for short an Iteration Method Fragment, is based in the SPEM homonym element. An Iteration Method Fragment consists of a *Process Pattern* that involves a set of *Activity Method Fragments* and/or a set of *Phase Method Fragments* that are repeated more than once during the process development lifecycle, offering a structuring fragment to organize work in repetitive cycles. Moreover, it must be classified by *Categories*.

Finally, a MAS method fragment in the Process layer, for short a *Process Method Fragment*, represents a whole MAS development cycle. It is composed of a *Process Pattern* classified by *Categories* that contains several *Phase Method Fragments* and/or *Iteration Method Fragments* and ends with a (major) *Milestone*. The goal of a *Process Method Fragment* is twofold. First, it depicts the notion of Base Method presented in Section 2, allowing a top-down fashion to configure a MAS situational Method. Second, it allows describing MAS original methods as a MAS Method Fragment ready to be used when an existing MAS method totally matches a given project situation.

The main advantages of having these four method fragment layers are: (i) the reuse of original AOSE methods in several level of granularity; (ii) the representation of MAS development approaches that do not provide a full development method (such as method fragments related to Agent Organizations models), and (iii) the utilization of bottom-up or top-down mechanisms for situational method configuration.

3 Applying the Proposed Definition to Tropos

In this section, we use the proposed definition to describe MAS method fragments sourced from Tropos. Tropos proposes a process for building MAS involving the following phases: *Early Requirements*, *Late Requirements*, *Architectural Design*, *Detailed Design* and *Implementation*. The goal of the first two phases is to provide a set of functional requirements as well as non-functional requirements for the system

to be built, while *Architectural Design* and *Detailed Design* phases focus on the system specification. Finally, the *Implementation* phase transforms the results of the preceding phases using an agent development platform in order to code the MAS.

We have used the Eclipse Process Framework Composer (EPF Composer) [12] to represent the MAS Method Fragments extracted from Tropos. EPF Composer is a tool developed by Eclipse Foundation that fully implements SPEM 2.0. After using the adequate SPEM element to represent each activity, step, role, diagram and model proposed by Tropos we have defined MAS Method Fragments into the Activity Layer, Phase Layer and Process Layer. We have not defined fragments into the Iteration Layer because Tropos does not propose iteration development cycles. Due to space constraints, we will describe only one example of fragment for each layer.

Fig. 3 depicts the External View of the Process Method Fragment called MMF Tropos Base Method, represented as a *Process Pattern* (called *Capability Pattern* in EPF Composer). On the left frame we can see that this fragment is classified in several categories of the MAS Semiotic Taxonomy, e.g. in the social level and iteration degree, it is classified as part of the Low Iteration Fragment Category.

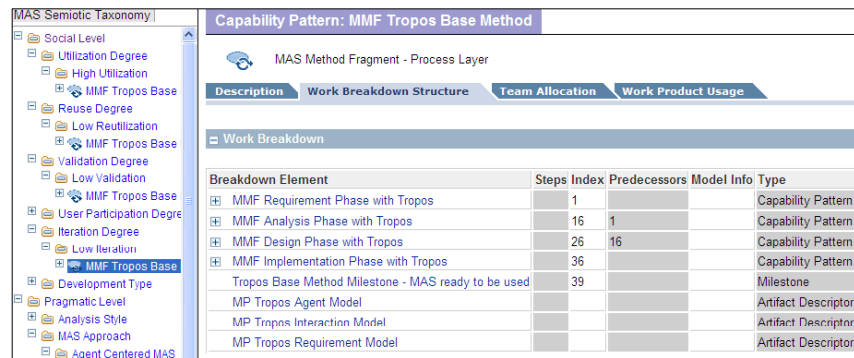


Fig. 3 External view of the MAS Method Fragment Tropos Base Method

On the right frame we can see that this fragment is composed of four Phase Method Fragments - MMF Requirement Phase with Tropos, MMF Analysis Phase with Tropos, MMF Design Phase with Tropos and MMF Implementation Phase with Tropos. As said before, we propose applying standard naming rules for method fragment identification. For instance, phase names should convey either the software development discipline covered by the phase - as *Requirement*, *Analysis*, *Design*, *Implementation*, *Test* - or the phase main development goal - as *Inception*, *Elaboration*, *Construction*, *Transition* – as currently used in iterative development approaches [13].

Moreover, the MMF Tropos Base Method fragment is composed of a *Milestone* called Tropos Base Method Milestone - MAS ready to be used that involves three work products: MP Tropos Agent Model, MP Tropos Interaction Model and MP Tropos Requirement Model, where MP stands for **M**AS work **P**roduct. Such work products encapsulate Tropos original work products and provide a standardized and common semantic for describing MAS work products based on the Vowels approach, as

proposed in Section 2. For instance, the work product MP Tropos Requirement Model encapsulates the *Tropos Actor Diagram* and *Tropos Goal Diagram*, both related to Tropos requirement phase.

As we have seen in Section 2, given that MMF Tropos Base Method is a Process Layer Fragment it can be used into two distinct ways: as standard representation of Tropos or as a Base Method in a top-down configuration mechanism in order to build a MAS situational Method.

Fig. 4 depicts the External View of the method fragment called MMF Requirement Phase with Tropos as well as some elements of its Internal View (highlighted rectangle of the right frame). On the left we can see the fragment classification using the MAS Semiotic Taxonomy. On the right we can see that this fragment is composed of three Activity Method Fragments: MMF Identify Initial Requirement with Tropos, MMF Detail Requirement with Tropos, and MMF Identify Additional Requirement with Tropos. Moreover, it contains the MAS Objectives Described Milestone that involves the work product MP Tropos Requirement Model.

Breakdown Element	Steps	Index	Predecessors	Model Info	Type
Requirement Phase with Tropos		1			Phase
MMF Identify Initial Requirement with Tropos		2			Capability Pattern
Identify Initial Requirement with Tropos		3			Activity
MTV Identify Stakeholders	●●●●	4			Task Descriptor
MMF Detail Requirements with Tropos		5	2		Capability Pattern
Detail Requirements with Tropos		6			Activity
MTV Analyze Goals and Plans	●●●●●	7			Task Descriptor
MMF Identify Additional Requirement with Tropos		8	5		Capability Pattern
MMF Detail Requirements with Tropos		11	8		Capability Pattern
MAS Objectives Described Milestone		14			Milestone
MP Tropos Requirement Model					Artifact Descriptor

Fig. 4 Phase MAS Method Fragment - Requirement Phase with Tropos

The Internal View of the MMF Detail Requirement with Tropos involves an homonym Activity that contains a Task Use (Task Description in EPF Composer) called MTV Analyze Goals and Plans. The acronym MTV used as task name prefix stands for MAS Task Variability and indicates that we are dealing with a task that extends another one. Such task variability has been defined over the Tropos original task in charge of analyzing goals and plans in order to provide basic MAS roles. In addition, it also changes Tropos original work products by the related MAS work products that forms the External Fragment View (see **Fig. 5** for more details).

It is worthy to notice that the fragment MMF Detail Requirement with Tropos is executed twice in the MMF Requirement Phase with Tropos: first after identifying initial requirements (concerning stakeholders identification) and latter after identifying additional requirements (defining system actors).

Finally, **Fig. 5** complements the Internal View of the fragment MMF Detail Requirement with Tropos showing its main elements.


Task Descriptor: MTV Analyze Goals and Plans		
 MAS Task Variability of Tropos Task Analyze Goals and Plans Based on Method Task: MTV Analyze Goals and Plans Expand All Sections		
Relationships		
Roles	Primary: • System Analyst	Additional: • MAS Designer
Inputs	Mandatory: • MPV Tropos Actor Diagram	Optional: • MPV Tropos Goal Diagram
Outputs	• MPV Tropos Goal Diagram	
Steps		
Expand All Steps <ul style="list-style-type: none"> ⊕ Analyzing goals through means-end analysis ⊕ Analyzing goals contributors ⊕ Decomposing goals ⊕ Analyzing plans through means-end analysis ⊕ Analyzing plans contributors 		

Fig. 5 Internal View of the fragment MMF Detail Requirement with Tropos

As the fragment MMF Detail Requirement with Tropos consists of an Activity Method Fragment, its elements are defined through the task MTV Analyze Goals and Plans. These are the following: (i) the roles System Analyst and MAS Designer, (ii) the mandatory input work product MPV Tropos Actor Diagram, (iii) the output work product MPV Tropos Goal Diagram, and (iv) several steps, as Analyzing goals through means-end analysis and decomposing goals.

These examples of usage show that the proposed definition of MAS Method Fragment offers a more standardized way for representing each phase and activity of Tropos, as well as Tropos own method as a whole. Moreover, it establishes mechanisms for the encapsulation and identification of original Tropos tasks, roles and work products. Finally, it allows reusing method fragments even in the context of its original method.

4. Related Work

In AOSE field there are several researches concerning method fragment notion. Among them we can cite the FIPA method fragment definition [10] and its refinement proposed by Cossentino et al. [7], as well as the fragment description for adaptive methodology proposed by Rougemaille et al. [17] and the three method fragment levels of granularity introduced by [8]. In general, such researches propose the use of SPEM as a common meta-model for representing MAS method fragment. For instance, Rougemaille et al. [17] highlight how SPEM can participate to design adaptive methodology process and claim that being compliant with SPEM is important to broaden the use of agent-oriented methodologies and principles. Nevertheless, the method fragment levels of granularity (atomic, composed, and phase level) proposed in [8] do not involve any metamodel. Instead, these method fragment levels are only briefly outlined in natural language.

The preliminary version of the FIPA method fragment definition [10], published in 2003 by the FIPA Methodology Technical Committee (TC)¹, states that a method fragment is a portion of the development process that involves the following elements: (i) a definition of a portion of process using SPEM describing what should be done; (ii) one or more deliverables; (iii) a required input data representing the preconditions to start the process specified in the fragment; (iv) a list of concepts related to the MAS meta-model to be defined / refined for the fragment; (v) guidelines that illustrate how to apply the fragment as well as best practices related to that; (vi) a glossary of terms used in the fragment; (vii) composition guidelines describing the context/ problem treated by the fragment; (viii) aspects of fragment such as the platform to be used, and finally (ix) the dependency relationships useful to assemble fragments. It is worthy noting that some of these elements are not mandatory, as input data and guidelines.

Our MAS method fragment definition is based on SPEM, as shown in Section 2, and it is fully compliant with this FIPA definition since the portion of process is represented by a **Task** element while **Work Product** elements describe fragment deliverables as well as fragment inputs. Moreover, we use **Guidance** elements to describe MAS concepts, to provide fragment guidelines and glossary of terms. Finally, our method fragment definition encompasses an external view that aims to describe the aspects that should be taken into account during method fragment selection and assembling, applying **Category** elements to classify the fragments based on their context use, their deliverable work products and their development platforms, among other criteria covered by the MAS Semiotic Taxonomy.

The refinement of FIPA definition proposed by Cossentino et al. [7] involves four different point of views for describing method fragments: (i) the process fragment view that deals with the process related aspects of a fragment, including workflows, activities and work products; (ii) the reuse fragment view for representing fragment elements such as the MAS meta-model, glossary of terms, guidelines and fragment dependency; (iii) the fragment storing view that deal with retrieving method fragments from the method base and, finally, (iv) the implementation view that concerns the implementation aspects of the process fragment view elements.

This refinement is distinct of our MAS method fragment definition in two main points: the SPEM compliance and the representation of MAS components. First, our approach is fully defined over SPEM (since we do not introduce new elements neither new associations in order to define method fragment elements) while Cossentino et al. [7] introduce new elements such as the Workflow and the MAS Model elements. In our opinion the SPEM compliance offer important benefits: on the one hand, SPEM is the “de facto” standard for method metamodel and, on the other hand, such compliance allows using SPEM based tools, as EPF Composer, for building the Method Base and configuring the MAS Situational Method. Second, this refinement requires dealing with MAS metamodel elements in a fine grained granularity and involves MAS metamodel integration while we propose representing MAS components in a coarse grained granularity based on the Vowel approach (Agent, Environment, Interaction, Organization). Therefore, our definition of method

¹ The activity of this TC stopped during the transition towards the new FIPA structure as part of IEEE Computer Society Standards Committee since 2005

fragment does not depend on a previous MAS metamodel integration in order to configure MAS situational methods. Such independence represents an important benefit since it allows taking advantage of Situational Method Engineering techniques for creating MAS situational methods without waiting for MAS community reaching a consensus about MAS main concepts.

Summing up, at the best of our knowledge, in the AOSE field there is no explicit method fragment definition tailored to facilitate the use of situational configuration mechanisms and to offer a standardized description of a method fragment elements and objectives. The IEEE-FIPA Design Process Documentation and Fragmentation Working Group² (DPDF WP), recently formed to deal with a definition of method fragment for situational method engineering process, corroborates that these topics constitute open issues in AOSE field.

5 Conclusions

In this paper, we presented a new definition for MAS method fragment based on Situational Method Engineering and Software Engineering techniques that can be used to represent MAS development approaches in a more standardized and coherent way. It offers two method fragment perspectives - internal and external fragment view - and four method fragment granularity layers: activity, iteration, phase and process. According to the proposed definition, a method fragment is considered coherent when its internal view is described using SPEM 2.0 and their elements and associations follow one of these four method fragment layers definition.

Moreover, we show how this definition can be used to represent method fragments extracted from Tropos, that is a well known MAS method. However, we think that it is not possible to identify a univocal criteria to set the best level of granularity for extracting method fragments, since fragment creation depends more on the experience of the method engineers than on a rigid extraction criteria. We claim that the absence of such a univocal criteria could be mitigated in two ways. First, using a common model, like SPEM, to represent method fragment. Second, establishing a procedure for extracting and storing fragments in a method repository. Such procedure is part of our research future work.

It is worthy to notice that part of the proposed method fragment definition is general enough to be applied to classical software development methods and not only to MAS methods. In fact, the MAS methods distinctive aspects are taken into account mainly through the following notions of our definition: the MAS Work Product Framework based on Vowel approach and some semiotic criteria, as MAS approach (agent /organization centered) and MAS nature (open/closed MAS). Therefore, in general lines, the proposed method fragment definition could be used to represent fragments sourced from classical software development methods after replacing these notions for other more suitable to a giving development paradigm.

We claim that this MAS Method Fragment definition can provide the backbone for defining a method repository for MAS situational method configuration, given that it provides mechanisms for (i) identifying method fragments using a well established

² <http://www.fipa.org/subgroups/DPDF-WG.html>

naming rules, (ii) conveying method fragment objectives and encapsulating original work products using a common work product framework, and (iii) categorizing method fragment based on a semiotic criteria. Finally, given that we propose distinct method fragment granularity layers, we can build a method repository involving fragments extracted from AOSE methods as well as from other MAS development approaches, such as Agent Organization models. In a top-down situational method configuration fashion, the former could provide a Base Method for the configuration while the latter could contribute with specific Activity Method Fragments in order to improve and complete a original MAS method.

Acknowledgments. This work is a product of the MEDEIA project, supported by FAPESP (Brazil) and CNRS (France). The first and the last two authors are partially supported by CNPq and CAPES (Brazil).

References

- [1] Bernon, C., Camps, V., Gleizes, M.-P., Picard, G.: Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds) Agent-Oriented Methodologies, Idea Group Pub. USA, 171–202 (2005)
- [2] Bresciani, P.; Giorgini, P.; Giunchiglia, F.; Mylopoulos, J.; Perini, A.: Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, vol 8(3), 203--236 (2004)
- [3] Brinkkemper, S.: Method Engineering: Engineering of Information Systems Development Methods and Tools. *Information and Software Technology*, vol. 38 (4), 275--280 (1996)
- [4] Brinkkemper S.; Saeki, M.; Harmsen, F. Meta-Modelling Based Assembly Techniques for Situational Method Engineering, *Information Systems*, 24(3), 209--228 (1999)
- [5] Casare, S.; Brandão, A. A. F.; Sichman, J. S. A Semiotic Perspective for Multiagent Systems Development (Extended Abstract), *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, 1373-1374 (2010)
- [6] Cossentino, M.: From Requirements to Code with the PASSI Methodology. In: Henderson-Sellers, B., Giorgini, P. (Eds.), *Agent-Oriented Methodologies*, Idea Group Publishing, 79--106 (2005)
- [7] Cossentino, M.; Gaglio, S.; Garro, A. ; Seidita, V. Method Fragments for agent design methodologies: from standardization to research. In: *International Journal on Agent Oriented Software Engineering (IJAOSE)*. 1(1) (2007)
- [8] Cossentino, M.; Fortino, G.; Garro, A.; Mascillaro, S. ; Russo. W..PASSIM: a simulation-based process for the development of multi-agent systems. *Int. Journal of Agent-Oriented Software Engineering*, 2(2):132:170, Inderscience Enterprises Ltd., UK (2008)
- [9] Demazeau, Y. From interactions to collective behavior in agent-based systems. *Proc. of the 1st. European Conference on Cognitive Science*. Saint-Malo, 117—132 (1995)
- [10] FIPA. Foundation for Intelligent Physical Agents, Methodology TC. Method Fragment Definition, Preliminar version 2003/11/21 (2003) Available on <<http://www.pa.icar.cnr.it/cossentino/FIPAmeth>>.
- [11] Guessoum, Z; Cossentino, M.; Pavón, J.: Roadmap of Agent-Oriented Software Engineering – The AgentLink Perspective. In: F. Bergenti, M. P. Gleizes, & F. Zambonelli

- (Eds.), Methodologies and software engineering for agent systems, Kluwer Academic Publishers, 431--450 (2004)
- [12] Haumer, P. Eclipse Process Framework Composer – Part 1 – Key Concepts. (2007)
Available on: <<http://www.eclipse.org/epf>>.
 - [13] Jacobson, I.; Booch, G.; Rumbaugh, J. The unified software development process. Addison-Wesley (1999)
 - [14] Karlsson, F.: Method Configuration - Method and Computerized Tool support. Doctoral Dissertation Dept. of Computer and Information Science, Linköping University (2005)
 - [15] OMG. Object Management Group. Software & Systems Process Engineering Meta-Model Specification, version 2.0. OMG document number: formal/2008-04-01 (2008) Available on <http://www.omg.org/spec/SPEM/2.0/PDF>.
 - [16] Pavon, J. Ingenias: Développement Dirigé par Modèles des Systèmes Multi-Agents. Dossier d'Habilitation à Diriger des Recherches de l'Université Pierre et Marie Curie. Paris, France (2006)
 - [17] Rougemaille S.; Migeon, F.; Millan, T.; Gleizes, M.-P.: Methodology Fragments Definition in SPEM for Designing Adaptive Methodology: A First Step. In: Luck, M.; Gomez-Sanz, J.J. (Eds.): AOSE 2008, LNCS, vol. 5386, Springer, 74--85 (2009)
 - [18] Zambonelli, F., Jennings, N. R.; Wooldridge, M.: Developing multiagent systems: The Gaia methodology. ACM Transaction on Software Engineering and Methodology, vol 12(3), 417--470 (2003)

A Glimpse of the ASPECS Process documented with the FIPA DPDF Template

Massimo Cossentino¹, Stéphane Galland², Nicolas Gaud², Vincent Hilaire²,
and Abderrafaa Koukam²

¹Istituto di Calcolo e Reti ad Alte Prestazioni
Consiglio Nazionale delle Ricerche
Palermo, Italy
cossentino@pa.icar.cnr.it

²University of Technology of Belfort Montbéliard.
90010 Belfort cedex, France
vincent.hilaire@utbm.fr
+33 384 583 009

Abstract. The FIPA DPDF working group aims at proposing a definition of method fragment to be used during a situational method engineering process, the fundamental elements it is composed of and the metamodel it is based on. Using the FIPA DPDF template, this paper introduces fragments issued from the ASPECS methodology. The process of this methodology, the underlying metamodel and the workproducts related to its first main phase, dedicated to system requirements analysis, are presented.

1 Introduction

It is currently admitted in mainstream software engineering and agent oriented software engineering that there is no one-size-fit-all methodology or process. Indeed, as stated in [6] ”*traditional rigid IS engineering methods are inadequate to provide the necessary support in new IS developments. New methods, more flexible and better adaptable to the situation of every IS development project, must be constructed*”.

One possible solution is based on the situational method engineering paradigm [5]. This latter provides means for constructing ad-hoc software engineering processes following an approach based on the reuse of portions of existing design processes, the so-called method fragments, stored in a repository called method base.

The Foundation for Intelligent Physical Agents (FIPA) is part of the IEEE Computer Society and promotes agent-based technology and interoperability of its standards with other technology. Among the current existing FIPA sub-groups, the Design Process Documentation and Fragmentation (DPDF) working group aims at proposing a definition of method fragment to be used during a situational method engineering process, the fundamental elements it is composed of and the metamodel it is based on.

The result of the work of the working group members is the definition of a template in order to document method fragments [8]. This paper illustrates the use of this template for a specific methodology, namely ASPECS¹ [1].

The paper structure respects the FIPA DPDF template. Section 2 introduces ASPECS with its global process and the metamodel which defines the underlying concepts of the methodology. After this initial section, the FIPA DPDF template contains a section per phase of the methodological process. Due to the lack of space, only a part of the first phase of ASPECS is described in Section 3. Eventually, Section 4 concludes.

2 Documented introduction to ASPECS

2.1 Global process overview

The ASPECS life cycle consists of three phases that are explained below and illustrated by Figure 1. It is not very different from typical iterative processes. Each iteration (through a phase) refine previous ones. The **System Requirements** phase aims at identifying a hierarchy of organisations, whose global behaviour may fulfil the system requirements under the chosen perspective. It starts with a Domain Requirements Description activity where requirements are identified by using classical techniques such as use case driven functional analysis. Domain knowledge and vocabulary associated to the problem domain are then collected and explicitly described in the Problem Ontology Description activity. Then, requirements are associated to newly defined organisations. Each organisation will therefore be responsible for exhibiting a behaviour that fulfils the requirements it is responsible for. This activity is called Organisation Identification and it produces an initial hierarchy of organisations that it is later extended and updated, with further iterations, in order to obtain the global organisation hierarchy representing the system structure and behaviour. The behaviour of each organisation is realised by a set of interacting roles whose goals consist in contributing to the fulfilment of (a part of) the requirements of the organisation within which they are defined. In order to design modular and reusable organisation models, roles are specified without making any assumptions on the structure of the agent that may play them. To meet this objective, the concept of capacity has been introduced. A capacity is an abstract description of a know-how, i.e. a competence of a role. Each role requires certain skills to define its behaviour and these skills are modelled by capacities. Besides, an entity that wants to play a role has to be able to provide a concrete realisation for all the capacities required by the role. Finally, the last step of the system requirements phase is the capacity identification activity. It aims at determining the capacities required by each role.

The second phase is the **Agent Society Design** phase that aims at designing a society of agents whose global behaviour is able to provide an effective solution to the problem described in the previous phase and to satisfy associated

¹ <https://aspecs.org>

requirements. The objective is to provide a model in terms of social interactions and dependencies among entities (holons and agents). Previously identified elements such as ontology, roles and interactions, are now refined from the social point of view (interactions, dependencies, constraints, etc). At the end of this design phase, the hierarchical organisation structure is mapped into a holarchy (hierarchy of holons) in charge of realising the expected behaviours. Each of the previously identified organisations is instantiated in form of groups. Corresponding roles are then associated to holons or agents. This last activity also aims at describing the various rules that govern the decision-making process performed inside composed holons as well as the holons' dynamics in the system (creation of a new holon, recruitment of members, etc). All of these elements are finally merged to obtain the complete set of holons involved in the solution.

The third and last phase (that may be decomposed in two sub-phases), namely **Implementation and Deployment** firstly aims at implementing the agent-oriented solution designed in the previous phase by deploying it to the chosen implementation platform, in our case, *Janus* [4]. Secondly, it aims at detailing how to deploy the application over various computational nodes. Based on *Janus*, the implementation phase details activities that allow the description of the solution architecture and the production of associated source code and tests. It also deals with the solution reusability by encouraging the adoption of patterns. The code reuse activity aims at integrating the code of these patterns and adapting the source code of previous applications inside the new one. It is worth to note that system developed by using other platforms can be designed as well with the described process. This phase ends with the description of the deployment configuration; it also details how the previously developed application will be concretely deployed; this includes studying distribution aspects, holons physical location(s) and their relationships with external devices and resources. This activity also describes how to perform the integration of parts of the application that have been designed and developed by using other modelling approaches (i.e. object-oriented ones) with parts designed with ASPECS.

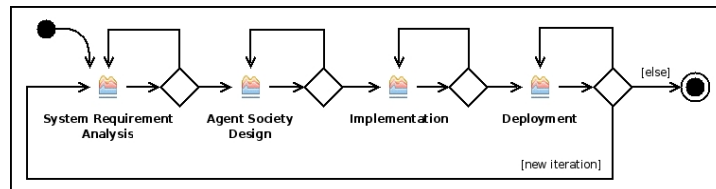


Fig. 1. ASPECS phases

2.2 Metamodel

The Problem Domain metamodel (see Figure 2), describing the concepts of the first phase, includes elements that are used to catch the problem requirements and perform their initial analysis: Requirements (both functional and non-functional) are related to the organisation that fulfils them. An organisation is composed of Roles, which are interacting within scenarios while executing their Role plans. An organisation has a context that is described by an ontology. Roles participate to the achievement of their organisation goals by means of their behaviours and capacities. In this subsection we will discuss the three most important elements of this domain: organisation, role, capacity. Definitions of all ASPECS metamodels can be found in [1] and on the ASPECS website².

An organisation is defined by a collection of roles that take part in systematic institutionalised patterns of interactions with other roles in a common context. This context consists in a shared knowledge, social rules/norms, social feelings, and it is defined according to an ontology. The aim of an organisation is to fulfil some requirements. An organisation can be seen as a tool to decompose a system, and it is structured as an aggregate of several disjoint partitions. Each organisation aggregates several roles and it may itself be decomposed into sub-organisations.

In our approach, a Role defines an expected behaviour as a set of role tasks ordered by a plan, and a set of rights and obligations in the organisation context. The goal of each Role is to contribute to the fulfilment of (a part of) the requirements of the organisation within which it is defined.

In order to cope with the need of modelling system boundaries and system interactions with the external environment, we introduced two different types of roles: Common Role and Boundary Role. A Common Role is located inside the designed system and interacts with either Common or Boundary Roles. A Boundary Role is located at the boundary between the system and its environment and it is responsible for interactions happening at this border (i.e. GUI, Database wrappers, etc).

Roles use their capacities for participating to organisational goals fulfilment; a Capacity is a specification of a transformation of a part of the designed system or its environment. This transformation guarantees resulting properties if the system satisfies a set of constraints before the transformation. It may be considered as a specification of the pre- and post-conditions of a goal achievement. This concept is a high level abstraction, which proved to be very useful for modelling a portion of the system capabilities without making any assumption about their implementations as it should be at the initial analysis stage.

A Capacity describes what a behaviour is able to do or what a behaviour may require to be defined. As a consequence, there are two main ways of using this concept:

² <http://janus-project.org>

- it can specify the result of some role interactions, and consequently the results that an organisation as a whole may achieve with its behaviour. In this sense, it is possible to say that an organisation may exhibit a capacity.
- capacities may also be used to decompose complex role behaviours by abstracting and externalising a part of their tasks into capacities (for instance by delegating these tasks to other roles). In this case the capacity may be considered as a behavioural building block that increases modularity and reusability.

In order to complete the description of the possibilities offered by the application of our definitions of Organisation, Roles and Capacity, let us consider the need of modelling a complex system behaviour. We assume it is possible to decompose it from a functional point of view, and in this way we obtain a set of more finer grained (less complex) behaviours. Depending on the considered level of abstraction, an organisation can be seen either as a unitary behaviour or as a set of interacting behaviours. The concept of organisation is inherently a recursive one [2].

The same duality is also present in the concept of holon as it will be shown later in this article. Both are often illustrated by the same analogy: the composition of the human body. The human body, from a certain point of view, can be seen as a single entity with an identity, its own behaviour and personal emotions. Besides, it may also be regarded as a cluster/aggregate of organs, which are themselves made up of cells, and so on. At each level of this composition hierarchy, specific behaviours emerge. The body has an identity and a behaviour that is unique for each individual. Each organ has a specific mission: filtration for kidneys, extraction of oxygen for lungs or blood circulation for the heart.

An organisation is either an aggregation of interacting behaviours, and a single behaviour composing an organisation at an upper level of abstraction; the resulting whole constitutes a hierarchy of behaviours that has specific goals to be met at each level. This recursive definition of organisation will form the basis of the analysis activities performed within ASPECS. In most systems, it is somewhat arbitrary as to where we leave off the partitioning and what subsystems we take as elementary (cf. [7, chap. 8]). This remains a pure design choice.

3 Phase: Domain

3.1 Process roles

Two roles are involved in the System Requirements discipline: the System Analyst and the Domain Expert. They are described in the following subsections.

System Analyst. S/he is responsible of:

1. Use cases identification during the Domain Requirements Description (DRD) activity. Use cases are used to represent system requirements.

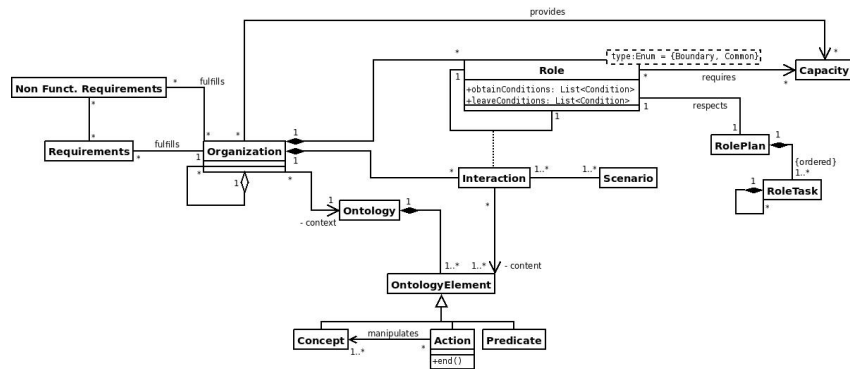


Fig. 2. Metamodel of the ASPECS problem domain

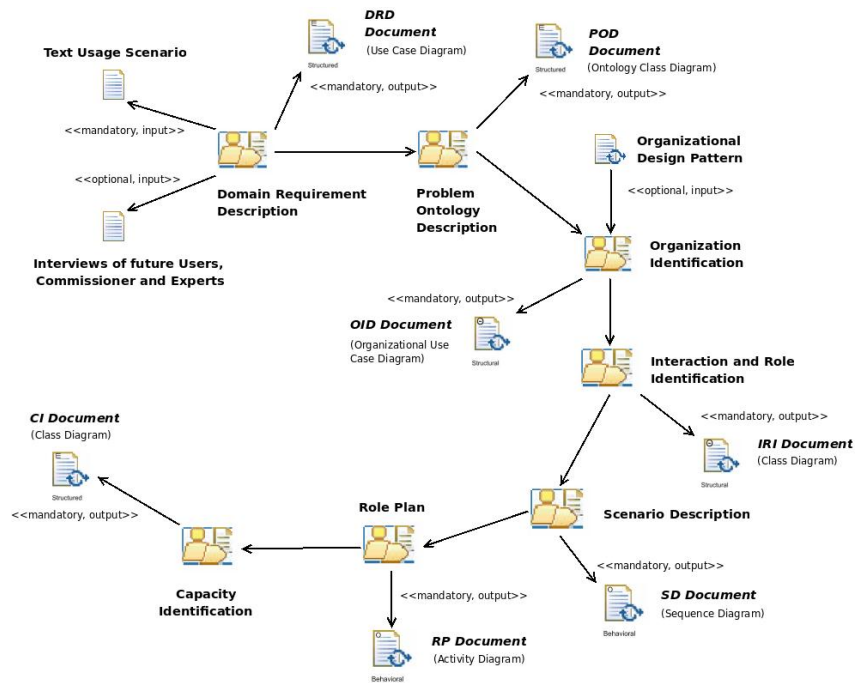


Fig. 3. System Requirements Phase: activities and workproducts

2. Use cases refinement during the DRD activity. Use cases are refined with the help of a Domain Expert.
3. Definition of an ontology for the conceptualisation of the problem during the Problem Ontology Description (POD) activity.
4. Use cases clustering during the Organisation Identification (OID) activity. The System Analyst analyses the use case diagrams resulting from the first activity and the domain concepts resulting from the second activity and attempts to assign use case to organisations in charge of their realisation.
5. Identification of interacting roles for the previously identified organisations and use cases constitutes the Interaction and Role Identification (IRI) activity.
6. Refinement of the interactions between roles during the Scenario Description (SD) activity by means of scenarios designed in form of sequence diagrams thus depicting the details of role interaction.
7. Refinement of role behaviours during Role Plan (RP) activity by means of state-transition diagrams specifying each role behaviour.
8. Identification of capacities that are required by roles or provided by the organisations during the Capacity Identification (CI) activity. The capacities are added to the class diagram depicting the organisations composed of interacting roles.

Domain Expert. The domain expert has knowledge about the domain of the problem to be solved and is able to decide if the requirements are identified (end of the Domain Requirements Phase).

3.2 Activity details

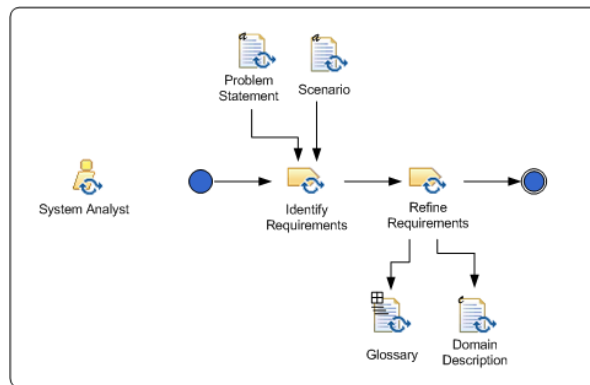


Fig. 4. Domain Requirement Description activity

Domain Requirement Description (DRD) The global objective of the Domain Requirements Description (DRD) activity is gathering needs and expectations of application stake-holders and providing a complete description of the behaviour of the application to be developed. In the proposed approach, these requirements should be described by using the specific language of the application domain and a user perspective. This is usually done by adopting use case diagrams for the description of functional requirements; besides, conventional text annotations are applied to use cases documentation for describing non-functional requirements. In ASPECS, we advocate the use of a combination between use-case driven and goal-oriented requirements analysis where the description of functional requirements is completed by the one of associated goals and goal failures.

Table 1. ASPECS Domain Requirement Description tasks

Activity	Task	Task description	Roles involved
Domain Requirements Description	Identify Use Cases	Use cases are used to represent system requirements	System Analyst (perform)
Domain Requirements Description	Refine Use Cases	Use cases are refined with the help of a Domain Expert	System Analyst (perform) Domain Expert (assist)

Organisation Identification (OID) The goal of the Organisation Identification activity is to bind each requirement to a global behaviour, embodied by an organisation. Each requirement is then associated to a unique organisation in charge of fulfilling it. As already said, an organisation is defined by a set of roles, their interactions and a common context. The associated context is defined according to a part of the Problem Ontology, described in the previous activity.

Starting from use cases defined in the DRD activity, different approaches could be used to cluster them and identify organisations. We advocate the use of a combination between a structural (or ontological) approach mainly based on the analysis of the problem structure described in the POD and a functional approach based on requirement clustering.

Structural analysis focuses on the identification of the system structure. It is based on the association between use cases and related ontological concepts. In structural organisation identification, use cases that deal with the same ontological concepts are often put together in the same organisation. This approach assumes the same knowledge is probably shared or managed by the different members of the organisation. The structure of the ontology itself can often con-

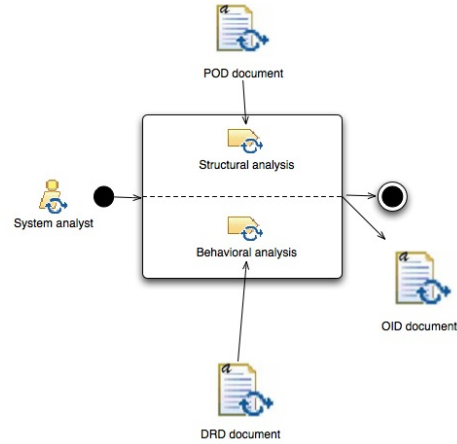


Fig. 5. Organisation Identification activity

stitute a good guideline to identify organisations, their composition relationships, and later their roles.

Behavioural analysis aims at identifying a global behaviour for the organisation intended to fulfil the requirements described in the corresponding use case diagram. The set of organisation roles and their interactions have to generate this higher-level behaviour. For this task, the use of *Organisational Design Patterns* may be useful to the designer. In behavioural organisation identification, use cases dealing with related pieces of the system behaviour are grouped (for instance an use case and another related to it by an include relationship). This means that members of the same organisation share similar goals.

3.3 Workproducts

The global objective of the Domain Requirements Description (DRD) activity is gathering needs and expectations of application stake-holders and providing a complete description of the behaviour of the application to be developed. In the proposed approach, these requirements should be described by using the specific language of the application domain and a user perspective. This is usually done by adopting use case diagrams for the description of functional requirements; besides, conventional text annotations are applied to use cases documentation for describing non-functional requirements.

The global objective of the Problem Ontology Description is to provide an overview of the problem domain. Problem ontology is modelled by using a class diagram where concepts, predicates and actions are identified by specific stereotypes.

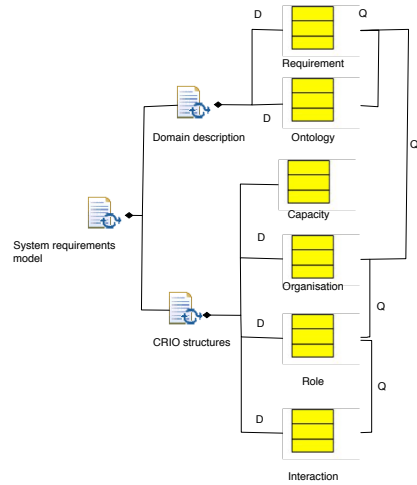


Fig. 6. ASPECS System Requirements Workproducts

Table 2. ASPECS Workproduct kinds

Name	Description	Workproduct kinds
DRD document	A text document composed by the Domain Description diagram, a documentation of use cases reported in it and the non-functional requirements of the system	Composite (Structured + Behavioural)
POD document	An ontology in the form of a class diagram stereotyped according to [3]	Structured
OID document	A class diagram reporting use cases and organisations as packages	Composite (Structured + Behavioural)
IRI document	A stereotyped class diagram	Structured
SD document	A stereotyped sequence diagram	Behavioural
RP document	An activity diagram	Behavioural
CI document	A stereotyped class diagram	Structured

The workproduct of the Organisation Identification activity (OID) refines the use case diagram produced by the DRD activity and add organisations as packages encapsulating the fulfilled use cases.

The result of the Interaction and Role Identification is a class diagram where classes represent roles (stereotypes are used to differentiate common and boundary roles), packages represent organisations and relationships describe interactions among roles or contributions (to the achievement of a goal) from one organisation to another.

Scenarios of the Scenario Description (SD) activity are drawn in form of UML sequence diagrams and participating roles are depicted as object-roles. The role name is specified together with the organisation it belongs to.

The resulting work product of the Role Plan (RP) activity is an UML activity diagram reporting one swimlane for each role. Activities of each role are positioned in its swimlane and interactions with other roles are depicted in form of signal events or object flows corresponding to exchanged messages.

The workproduct produced by the Capacity Identification is a refinement of the IRI diagram by adding capacities (represented by classes) and relating them to the roles that require them.

4 Conclusion

This paper has presented the use of the FIPA DPDF working group template with a specific MAS methodology, namely ASPECS [1]. Only the first of the three phases composing ASPECS is presented and in this phase two activities are detailed. The aim were twofold, first to prove the usability of the FIPA DPDF template and second to show a glimpse of the fragmentation of the ASPECS methodology. For more details about the methodology, the reader may refer to either the ASPECS reference paper [1] or its website: <http://aspecs.org>.

In this section we will discuss the work done for and the results obtained by adopting the novel FIPA process documentation template to the ASPECS process in order to evaluate its suitability and to estimate possible advantages of producing such a documentation. There are several steps to consider in the path towards the documentation of a process according to the new FIPA template. First, the (original) process has to be documented at a level of detail that is not common to most existing contributions from literature. Second, the documentation has to be converted to the FIPA specification. This means: (i) adopting the SPEM metamodel for process-related aspects, (ii) adopting a proper decomposition of the process (according to FIPA template), (iii) conveniently documenting the different parts, and finally, (iv) studying the metamodel and its relationship with activities and workproducts. The documentation of ASPECS according has proved to be quite an easy task. This is due to the specific origin of ASPECS. Differently from almost all the other AOSE processes, ASPECS has been built by rigorously following a situational method engineering approach (PRODE more specifically). This means ASPECS is essentially composed by process fragments originating from PASSI, RIO and from specifically created new ones. In such an

approach, the fragments have been built (and documented) largely before the new process and therefore their resulting assembly has been easily documented. Referring to the previously listed of steps for obtaining a FIPA compliant documentation, the first step (obtaining a sufficiently refined documentation) has not really been an effort. The ASPECS development history directly produced that. As regards the second step (converting the documentation to the FIPA specification), we will discuss the specific details: (i) as regards SPEM adoption, we already were adopting it during ASPECS development so we had no troubles with that, (ii) as regards process decomposition, we identified the correct granularities and (iii) we extracted the proper information from existing ASPECS documents. This has been the greatest part of the job and it took about one day of work. Finally, the metamodel-related part of the work (iv) was really straightforward because creating ASPECS with PRODE caused to have a deep study of metamodel and its relationships with activities and work products as required by the FIPA template. Trying an evaluation of the obtained results, we can say that the work done was not so much but the outcome documentation has some evident advantages. The most evident consisting in highlighting the similarities (and differences) with the methodologies that are the parents of ASPECS: PASSI and RIO. Summarizing, the template proved to be efficient, of easy application to a huge process like ASPECS and it supported all the needs we faced in the work.

References

1. Massimo Cossentino, Nicolas Gaud, Vincent Hilaire, Stéphane Galland, and Abderrafîaa Koukam. ASPECS: an Agent-oriented Software Process for Engineering Complex Systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, march 2010.
2. Jacques Ferber. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison Wesley, London, 1999.
3. Foundation For Intelligent Physical Agents. *FIPA RDF Content Language Specification*, 2001. Experimental, XC00011B.
4. Nicolas Gaud, Stéphane Galland, Vincent Hilaire, and Abderrafîaa Koukam. An Organisational Platform for Holonic and Multiagent Systems. In *PROMAS-6@AAMAS'08*, Estoril, Portugal, May 12-16th 2008.
5. Brian Henderson-Sellers. Method engineering: Theory and practice. In *Proc. of Information Systems Technology and its Applications, 5th International Conference ISTA 2006*, pages 13–23, 2006.
6. Jolita Ralyté and Colette Rolland. An approach for method reengineering. *Lecture Notes in Computer Science*, 2224:471–484, 2001.
7. Herbert A. Simon. *The Science of Artificial*. MIT Press, Cambridge, Massachusetts, 3rd edition, 1996.
8. FIPA DFDG WG. Design Process Documentation Template. Technical report, 2010.

Process Documentation Standardization: An Initial Evaluation

Massimo Cossentino¹, Alma Gómez-Rodríguez², Juan C. González-Moreno², Ambra Molesini³, and Andrea Omicini³

¹ Istituto di Calcolo e Reti ad Alte Prestazioni
National Research Council, Palermo, Italy
cossentino@pa.icar.cnr.it

² Departamento de Informática, Universidade de Vigo, Ourense, Spain,
{alma,jcmoreno}@uvigo.es

³ ALMA MATER STUDIORUM – Università di Bologna, Italy
ambra.molesini@unibo.it, andrea.omicini@unibo.it

Abstract. The creation of new ad-hoc methodologies through the Situational Method Engineering approach needs the process fragments to be defined and available. Thus, it is necessary to previously define and extract such fragments from the global development process. So, it is important to provide the means of documenting the whole process from which fragments will be obtained. This paper presents an experimental evaluation of the methodologies documentation template proposed by the IEEE FIPA Design Process Documentation and Fragmentation working group. The template will be used for documenting three different agent-oriented methodologies in order to evaluate the template's strengths and weaknesses.

1 Introduction

Nowadays, in the software engineering field, there is a common agreement about the fact that there is not a unique methodology or process that fits all the application domains; this means that the methodology or process must be adapted to the particular characteristics of the domain for which the new software is developed. There are two major ways for adapting methodologies: tailoring (particularisation or customisation of a pre-existing processes) or Situational Method Engineering (SME) [1, 2]. In the last case the process is assembled from pre-existing components, called fragments, according to user's needs. This approach enhances reusability since a method component can be used several times.

The research on SME has become crucial in the Agent-Oriented Software Engineering (AOSE) since a variety of (special-purpose) agent-oriented (AO) methodologies have been defined in the past years [3–7] to discipline and support the multi-agent system (MAS) development. Each of the AO methodologies proposed up to now exhibits a specific metamodel, notation, and process. All of these features are fundamental for a correct understanding of a methodology, and should be suitably documented for supporting the creation of new ad-hoc

AO methodologies. In fact, the SME technique is strictly related to the documentation of the existing methodologies since the successful construction of a new process is based on the correct integration of different fragments that should be well formalized. So, methodologies' documentation should be done in a standard way in order to facilitate the user's understanding, and the adoption of automatic tools able to interpret the fragment documentation.

In this context, the IEEE FIPA Design Process Documentation and Fragmentation (DPDF) working group [8] has recently proposed a template for documenting AO methodologies. This template takes into account the three aforementioned methodologies' features. In first place, it has been conceived without considering any particular process or methodology, and this should guarantee that all processes can be documented using the proposed template. Moreover, the template is also neutral regarding the MAS metamodel and/or the modelling notation adopted in describing the process. Secondly, the template has a simple structure resembling a tree. This implies that the documentation is built in a natural and progressive way, addressing the process general description and metamodel definition which constitute the root elements of the process itself. Then, the process phases are described as branches of the tree. Finally, thinner branches like activities or sub-activities can be documented. This means the template can support complex processes and very different situations. In third place, the use of the template is easy for any software engineer as it relies on very few previous assumptions. Moreover, the suggested notation is the OMG's standard Software Process Engineering Metamodel (SPEM) [9] with few extensions [10].

So, the goal of this paper is to present an experimental evaluation of the FIPA DPDF template by means of the application of such a template to three different AO methodologies: PASSI [11], INGENIAS [12], and SODA [13].

Accordingly, the remainder of the paper is organized as follows. Section 2 provides a brief description of the FIPA DPDF template, while Section 3 presents the application of the template to the three chosen AO methodologies. Section 4 presents some proposals for the improvement of the current version of the FIPA template, whereas Section 5 presents a discussion about the results obtained by the application of the template to the documentation of the three chosen methodologies. Finally, the conclusions of the whole work are reported in Section 6.

2 Process Documentation Template in a Nutshell

The IEEE FIPA DPDF working group has recently proposed a template for documenting AO methodologies. Here we report only a brief presentation of the template—interested readers can refer to [8] for the details of the template.

The template is based on the definition of process and process model as proposed by [14]. A process model is supposed to have three basic components: the stakeholders (i.e. roles and workers), the consumed and generated products (i.e. work products), and the activities and tasks undertaken during the process—

these being particular instances (i.e. work definitions) of the work to be done. Another important component of the template is the MAS metamodel, as previously considered in [10], because it is thought that the MAS metamodel may constrain the way in which fragments can be defined and reused.

1.Introduction 1.1.The (process name) Process lifecycle 1.2.The (process name) Metamodel 1.2.1. Definition of MAS metamodel elements 2.Phases of the (process name) Process 2.1.(First) Phase 2.1.1.Process roles 2.1.2.Activity Details 2.1.3.Work Products 2.2 (Second) Phase 2.2.1.Process roles 2.2.2.Activity Details 2.2.3.Work Products ... (further phases) ... 3.Work Product Dependencies

Table 1. The proposed process template

The template schema reported in Table 1 introduces the fundamental components of the process model definition. As it can be easily seen, the template has a structure that provides a natural decomposition of the process elements in a tree-like structure where the *Introduction* – including a description of the process lifecycle and the MAS metamodel – is at the root. Introduction is meant to give a general overview of the process detailing the original objectives of the process/methodology, its intended domain of application, scope, limits and constraints (if any), etc. The *Metamodel* part provides a complete description of the MAS metamodel adopted in the process with the definitions of its composing elements. This means the different conceptual elements considered when modeling the system must be identified and described. The focus on the MAS metamodel is not new in the agent-oriented community, and is also coherent with the current emphasis on model-driven approaches, which are always based on the system metamodel. The process is supposed to be composed – from the work-to-be-done point of view – by phases. Each phase is composed of activities that, in turn, may be composed of other activities or tasks. This structure is compliant to the SPEM specification which is explicitly adopted as a part of this template although with some (minor) extensions (see [10]). The next template part is represented by several Phase sections, one for each phase composing the whole process. The main aim of each phase is to define the phase from a pretty process-oriented point of view; that is, workflow, work products and process roles are the center of the discussion. Initially, the phase workflow is introduced by using a SPEM activity diagram which reports the activities composing the

phase, and by doing a quick description of work products and roles. A SPEM diagram follows reporting the structural decomposition of each activity in terms of the involved elements: tasks, roles and work products.

In the last section, the template discusses work products with a twofold goal: the first part aims at detailing the information content of each work product by representing which MAS model elements are reported in it (and which operations are performed on them). The second part focuses on the modeling notation adopted by the process in the specific work product. The work products are described by using a SPEM work product structured document. This diagram is a structural diagram reporting the main work product(s) delivered by each phase, and the diagrams are completed by a table that describes the scope of each work product. Finally, work product dependencies are reported in a specific diagram.

3 Case Studies

The next subsections discuss the documentation of three AO processes (PASSI, INGENIAS, and SODA) using the previously proposed template. In this way, the paper tries to evaluate the suitability of the template for modeling processes. The validation is significant because the chosen methodologies follow different kinds of process and have significant differences in other composing elements. For space reason, here we report only some excerpts of the methodologies documentation—the interested reader can find more details in [8]. In particular, the next subsections present the requirement analysis phase for each of the three AO methodologies considered in this paper.

3.1 PASSI

PASSI (Process for Agent Societies Specification and Implementation) is a step-by-step requirement-to-code methodology for designing and developing multi-agent societies.⁴ The methodology integrates design models and concepts from both object oriented software engineering and artificial intelligence approaches.

The PASSI design process is composed of five models: the System Requirements Model regards system requirements; the Agent Society Model deals with the agents involved in the solution in terms of their roles, social interactions, dependencies, and ontology; the Agent Implementation Model is a model of the solution architecture in terms of classes and methods; the Code Model depicts the solution at the code level; and the Deployment Model describes the distribution of the parts of the system.

Following the schema proposed in Section 2, Figure 1 summarizes the description of the System Requirements phase. In particular, this phase involves two different process roles, eight work products (four UML models and four text

⁴ PASSI documentation can be found at http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/PASSI_SPEM_2_0_ver0.2.8.pdf.

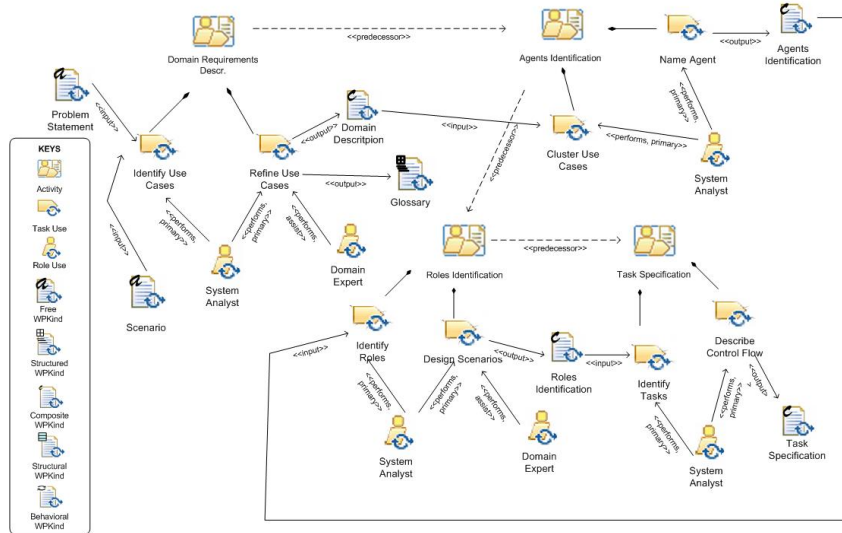


Fig. 1. The System Requirements activities, work products and stakeholders

documents) and four guidance documents (one for each UML model). The phase is composed of four activities: Domain Requirements Description, Agents Identification, Roles Identification and Task Specification, each of them composed of one or more tasks (for instance Identify Use Cases and Refine Use Cases) and delivering one work product as described by Figure 1. Tasks are under the responsibility of one or more stakeholders involved with the responsibility to perform or assist in the work to be done.

The System Requirements Model generates four composed work products (text documents including diagrams). Their relationships with the MAS meta-model elements (MMM) are described in Figure 2 where the containment relationship between each MMM element and a work product is labelled according to the action performed on the element (D means define/instantiate, R means relate, Q means quote/cite, RF means refine).

3.2 INGENIAS-UDP Process

The INGENIAS methodology covers the analysis and design of MAS and is intended for general use—that is, with no restrictions on application domains.⁵ The software development process proposed by the methodology is based on Rational Unified Process [15]. The methodology distributes the tasks of analysis

⁵ INGENIAS documentation can be found at <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/INGENIAS.pdf>.

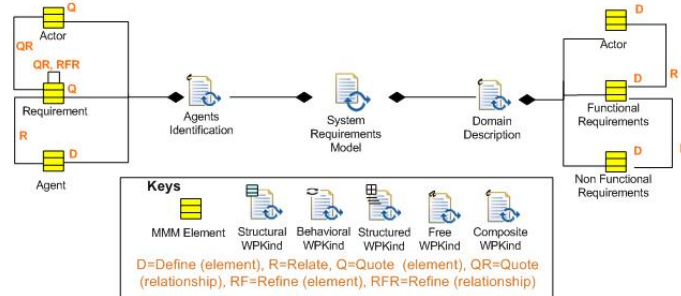


Fig. 2. The PASSI System Requirements documents structure

and design in three consecutive phases: Inception, Elaboration and Construction. Each phase may have several iterations (where iteration means a complete cycle of development). The sequence of iterations of each phase leads to the procurement of the final system. Figure 3 shows a detailed description of Inception phase of INGENIAS process.

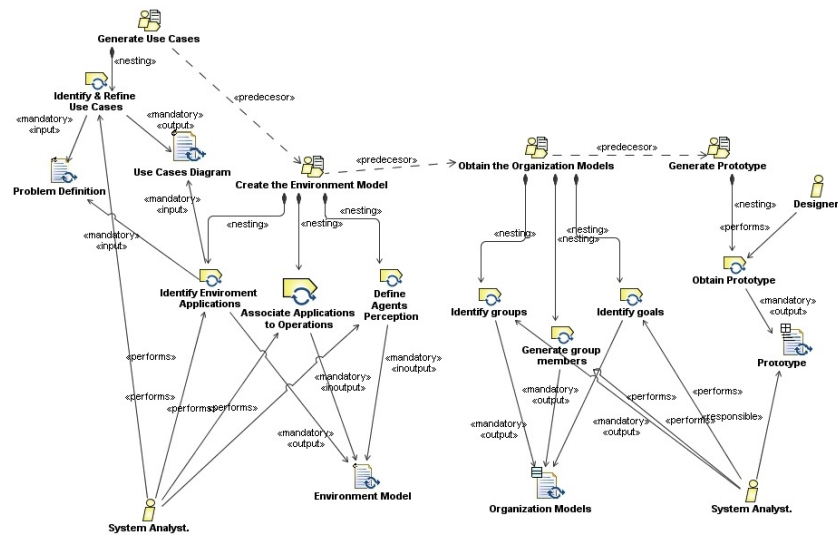


Fig. 3. INGENIAS Inception activities, workproducts and stakeholders

INGENIAS considers the development as starting from the document describing the problem, which is a mandatory input in this phase. The Inception phase is composed of several activities: generate use cases, create the Environ-

ment Model, Obtain the Organization Model and Generate Prototype. All these activities imply an important set of tasks and produce several workproducts as output, such as the Environment Model, the Organization Model or the Prototype. Besides, two roles are responsible as this phase: the System Analyst and the Designer.

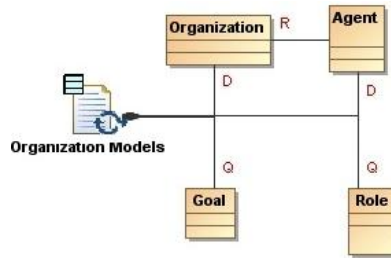


Fig. 4. INGENIAS Inception work products

As discussed above, one of the diagrams proposed in the template relates workproducts to the metamodel elements. In 4, the diagram is used for describing that the INGENIAS organisation models defines (D) the organization of the system and the agents related (R) to this organisation; while goal and role elements are only used (Q) but must have been defined previously.

3.3 SODA Process

SODA (Societies in Open and Distributed Agent spaces) [7, 16] is an agent-oriented methodology for the analysis and design of agent-based systems, which adopts the Agents & Artifacts meta-model (A&A) [17], and introduces a *layering principle* as an effective tool for scaling with the system complexity, applied throughout the analysis and design process.⁶ The SODA process is organised in two phases, each structured in two sub-phases: the *Analysis phase*, which includes the Requirements Analysis and the Analysis steps, and the *Design phase*, including the Architectural Design and the Detailed Design steps. In addition, since the SODA process is iterative and incremental, each step can be repeated several times, by suitably exploiting the layering principle: so, for instance, if, during the Requirements Analysis step (Figure 5), the Requirements Analyst – one of the roles involved in the SODA process – recognizes some omissions or lacks in the requirements’ definition, he/she can restart the *Requirements modelling* activity adding a new layer in the system or selecting a specific layer and then refining it through the *Requirement Layering* activity.

⁶ SODA documentation can be found at <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/SODA.pdf>.

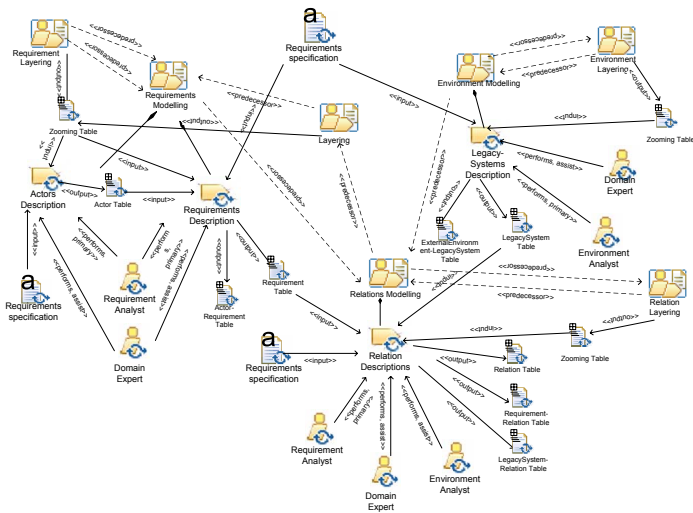


Fig. 5. The Requirements Analysis activities, work products and stakeholders

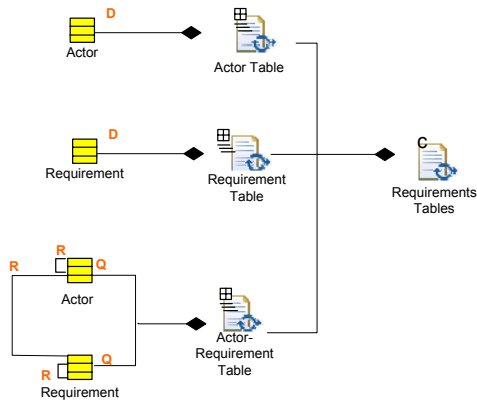


Fig. 6. The Requirements Analysis documents structure

In particular, the Requirements Analysis step involves three different process roles, nine work products (relational tables). The step is composed of three *normal* activities and four layering activities: the *normal* are Requirements Modelling, Relations Modelling, Environment Modelling, while the layering are Requirement Layering, Relation Layering, Environment Layering, and Layering. Tasks are under the responsibility of one or more roles involved with the responsibility to perform or assist in the work to be done.

Figure 6 reports only an excerpt of the Requirements Analysis documents structure. In SODA the work products are represented as relational tables organized in different sets. In particular, the diagram in Figure 6 reports the Requirements Tables set. This set describes the system requirements in terms of **Requirement** and **Actor** concepts of the SODA’s metamodel: each table has a specific relationships with one or more MAS metamodel elements. For example, the Actor table and the Requirement table *define* (D) [10] respectively the **Actor** and the **Requirement** while Actor-Requirement table *quote* (Q) both and *relate* (R) **Actor** and **Requirement**.

4 Proposals for FIPA DPDF Template Improvement

During the application of the FIPA DPDF template to the three methodologies, we collected some important feedbacks on its effectiveness. Most of them will be discussed in the next section as an assessment of the work done, whereas a couple are now proposed in terms of proposals for the improvement of the FIPA DPDF template.

The first issue concerns the absence in the template of a clear indication of where to describe the techniques and guidances [9] applied both in the overall process and in some specific part of the process. In particular, when trying to document SODA we had some problems in the documentation of the layering technique. This is quite a peculiar aspect of SODA, which adopts the layering principle as a tool for managing the system complexity and it spreads all over the process—excluding the Detailed Design phase. We found two issues related to the layering documentation: *(i)* where to put the layering technique description; *(ii)* the definition of the best structure for the documentation.

In order to manage the above issues we created a new sub-section in the template introduction (Table 1); this is like the description of the single phases, since the layering technique has a portion of process with its specific activities and tasks, and obviously its work products. The proposed change perfectly suits the need for introducing the layering technique before the description of the details of the process and the structure of the section is flexible enough to fit similar needs arising in other processes.

Another limit we found in the FIPA DPDF template is the lack of a specification for detailing the content of a task. Activities are decomposed – or better decomposable according to the needs – in tasks in section 2.1.2 (see Section 2) of the template but this may not be general enough. What about the description of quite a complex task? SPEM provides the method engineer with the opportu-

nity to use the *Step* element for decomposing tasks. It is worth to remind that – according to SPEM specification [9] – a Step is “*a Section and Work Definition that is used to organize a Task Definition as Content Description into parts or subunits of work. . . . A Step describes a meaningful and consistent part of the overall work described for a Task Definition. The collection of Steps defined for a Task Definition represents all the work that should be done to achieve the overall development goal of the Task Definition*”. According to this definition the usage of the Step element may prove to be very useful. It may happen – and we actually found some occurrences of that in our processes – that one specific task is too complex to be exhaustively described in the text proposed in section 2.1.2 of the template (see Section 2). It may be even the case to describe a task with an activity diagram reporting the flow of work to be done. Steps would be the main components of this diagram and, in turn, they would need a text description of the work to be done inside them.

Actually, the FIPA DPDF template specification document [8] proposes the structure for describing activities as showed in Table 2.

<p>4.1.2.1. Activity 1 GOAL: Describe the work to be done within this activity STRUCTURE: Details of tasks and sub-activities are specified with a table that includes the following columns:</p> <ul style="list-style-type: none"> - Activity: name of the activity studied in this subsection. - Tasks/Sub-Activity: sub-activity or task described in this row. - Task/Sub-activity Description. - Roles involved. <p>Optionally, the control flow within a Task can be illustrated by a stereotyped UML Activity Diagram. These diagrams explain the execution of complicated Tasks by denoting the possible sequences of Steps, which are identified by the << steps >> stereotype. Details on this modeling of Tasks can be found in the current SPEM specification.</p> <p>When documenting a Task in this way, the diagrams are appended and each diagram is discussed in a separated paragraph that explains the illustrated steps and their relations.</p>

Table 2. The activity description section in the current FIPA DPDF template

The FIPA DPDF template already prescribes the possibility to detail tasks by using steps in forms of activity diagrams, however it does not give any hint on how to document them. In order to improve the template, we propose to introduce a new optional subsection in each activity description section as depicted in Table 3.

As an example of such an extension, the decomposition of the INGENIAS Identify Environment Application task is presented in Table 4. The activity diagram depicting the workflow is omitted because of space concerns and also because the steps are performed one after the other in a simple way.

<p>4.1.2.1.1 Decomposition of Task x of Activity 1</p> <p>GOAL: Describe the work to be done within Task x of Activity 1.</p> <p>STRUCTURE: The workflow may be depicted by using an activity diagram reporting the steps to be done within the task.</p> <p>Details of steps are specified with a table reporting the following columns:</p> <ul style="list-style-type: none"> - Activity: name of the activity the task studied in this subsection belongs to. - Task: name of the task detailed in this subsection. - Step: name of the Step described in this row of the table - Step Description: plain text describing the work to be done within this step. - Roles involved: roles involved in executing this step.

Table 3. The proposed extension to the FIPA DPDF template for a detailed description of tasks decomposition in steps.

Activity	Task	Step	Step Description	Roles involved
Create the Environment Model	Identify Environment Applications	Identify External Applications	Find standard packages and external software agents need to use or to communicate with	System Analyst
Create the Environment Model	Identify Environment Applications	Identify Internal Applications	Identify centralized software services agents has to shared and whose nature is not like that of an agent	System Analyst

Table 4. Steps in the INGENIAS Identify Environment Application Task

5 Discussion

This paper evaluates a template for process documentation that seems to provide a good framework in the documentation of processes for AO development. This template is based on an approach similar to the one proposed by Rumbaugh [18] in introducing UML. The approach prescribes the removal of all cluttering information – for instance, different notations – in order to highlight commonalities (and differences). As a result, the study of a new methodology becomes easier to a designer who is already fluent with the documentation style adopted in this approach. The FIPA DPDF template proposes a division of the process in phases, activities and tasks as introduced in Section 2. In this paper, we were able to identify (with a similar granularity) the phases, activities and tasks for the processes introduced in PASSI, INGENIAS and SODA (see Figures 1, 3 and 5) without specific problems—thus proving the soundness of the approach.

In particular, the figures show the flow of activities, the work products and the stakeholders of the first phase of each methodology. By analysing the figures, it is easy to understand the specific flow of activities and tasks to be followed when using the methodologies. On the other hand, the diagrams highlight the

differences among the three methodologies such as for instance the different attention paid to the environment modelling. This is a primary activity in SODA, a task in INGENIAS, while PASSI delays the study of the environment to the next phase. Another difference regards the identification of roles and agents: this is done in the first phase of the process in PASSI and INGENIAS, whereas SODA defines the same abstractions only in the design phase.

An important feature of the template is the attention paid to the MAS meta-model adopted in the process. Such a feature provides an interesting point in methodological comparison. For instance, from the comparison of the documentations produced in this study, we easily deduce that INGENIAS (Figure 4) has a reduced set of models, which however are quite complex since each of them includes many concepts. On the other hand, PASSI and SODA – Figures 2, 6 – have respectively more diagrams and tables, but each of them introduces only a few concepts. The use of the template easily supports the identification of such differences.

Furthermore, in the template, the MAS metamodel elements and their relationships are also related with work products depicting them—see respectively Figures 2, 4, and 6. Considering work products as a part of the process is fundamental for fragment definition and usage, as long as, the user must take into account the desirable results for selecting a fragment or she/he must consider what inputs are needed before it is possible to initiate a phase or an activity of the process. Moreover, the definition of different processes for several methodologies using this template – see also [8] for the documentation of others AO methodologies – suggests that it is general enough, because good results have been obtained for three different methodologies.

As previously discussed, we point out several benefits in using the proposed template. First of all, the template makes it easier to understand the process workflow, and also produces a documentation that may help in studying it. Moreover, it seems evident that it will be easier to study a new methodology when this new one is documented with an already known approach. For instance, PASSI and SODA metamodels are different in the content – different elements, concepts and models – but the similar description approach largely allows for an easy identification and study of differences between them.

Another important benefit of defining the process is that it provides a starting point for fragments extraction, and therefore for process elements reuse. The reuse would start by identifying fragments considering, for instance, as a starting point the work products produced by the fragment. The template provides diagrams that facilitate the identification. For instance the work products dependencies diagram makes it possible to introduce an order in the work products obtained. On the other hand, the diagrams detailing the activities identify the input and output work products of each task. All such information should be considered when defining a fragment.

Some limitations were noticed when documenting the processes. One issue is related to the SPEM notation: the presence of the layering activities in SODA leads to the construction of diagrams that are very difficult to understand due

to the huge number of strictly-related activities. In particular, in Figure 5 there are four different layering activities – one for the iteration and three for the models refinement –, and the only *predecessor* relation is not powerful enough for explaining the right flow of activities in the Requirements Analysis phase—so this diagram alone is not sufficiently expressive. In this diagram, at the best of our knowledge, there is no way for expressing too many information in a clear way. The problem mainly arises in SODA because of the adoption of the layering technique but generally speaking it may regard other methodologies. The essence of the problem is deeply relied to the SPEM notation and we have not solved this problem yet; we plan to find alternative solutions in the future.

Some other minor problems have arisen when documenting the processes. These problems are more related with identifying specific details of the process from available documentation rather than with the template itself. Usually, when defining a methodology authors are more worried about identifying the models to construct, the concepts to define, etc. than in detailing phases and activities to be done or in indicating the order of these activities. In some way, the adoption of the template would force the designer for a new methodology to produce a complete specification thus improving the quality of the result.

6 Conclusions and Further Work

In this paper we used the FIPA DPDF template for documenting three different AO design processes. Documentation of processes has many advantages such as: comparing and evaluating methodologies in an easy way, simplifying fragment definition and selection, and so on. This work has demonstrated the power of the template in process documentation, and sketched some of its advantages. Nevertheless, it has been made available to the scientific community, so that other processes and/or other methodologies may be defined used the template.

This work is an initial step toward the definition of a standard for fragment documentation, extraction and use. This means that in the future the models produced for these methodologies will be used for identifying and documenting fragments. The fragments will then be reused and integrated so as to provide new ways of developing AO systems. Besides, although all this work has been done within the frame of AO development, we guess that the template could be general enough to define methodologies in other fields of development. Further work should be done to prove such a statement.

7 Acknowledgements

This work has been partially supported by the project *Novos entornos colaborativos para o ensino* supported by Xunta de Galicia with grant 08SIN009305PR and by the FRASI project of the Italian Ministry of Education and Research (MIUR).

References

1. Brian Henderson-Sellers, C.G.P.: Metamodelling for software engineering. ACM Press New York, NY, USA (2003)
2. Sorbonne, U.D.P., Rolland, C., Rolland, C.: A primer for method engineering. In: Proceedings of the INFORSID Conference. (1997) 10–13
3. Cuesta, P., Gómez, A., González, J., Rodríguez, F.J.: The MESMA methodology for agent-oriented software engineering. In: Proceedings of First International Workshop on Practical Applications of Agents and Multiagent Systems (IW-PAAMS'2002). (2002) 87–98
4. O'Malley, S.A., DeLoach, S.A.: Determining when to use an agent-oriented software engineering paradigm. In Wooldridge, M., Wei, G., Ciancarini, P., eds.: Agent-Oriented Software Engineering. Second Int. Workshop, AOSE 2001. Volume 2222 of Lecture Notes in Computer Science. Springer-Verlag (2002)
5. Bernon, C., Cossentino, M., Pavón, J.: Agent-oriented software engineering. *Knowl. Eng. Rev.* **20** (2005) 99–116
6. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. Multi-Agent Systems and Applications III **2691** (2003) 394–403
7. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In Ciancarini, P., Wooldridge, M.J., eds.: Agent-Oriented Software Engineering. Volume 1957 of LNCS. Springer-Verlag (2001) 185–193
8. IEEE FIPA Design Process Documentation and Fragmentation: IEEE FIPA Design Process Documentation and Fragmentation Homepage. <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/> (2009)
9. O.M.G.: Software Process Engineering Metamodel Specification. Version 2.0, formal/2008-04-01. <http://www.omg.org/> (accessed on June 25, 2009) (2008)
10. Seidita, V., Cossentino, M., Gaglio, S.: Using and extending the spem specifications to represent agent oriented methodologies. In Luck, M., Gómez-Sanz, J.J., eds.: AOSE. Volume 5386 of Lecture Notes in Computer Science., Springer (2008) 46–59
11. Cossentino, M.: From requirements to code with the PASSI methodology. [19] chapter IV 79–106
12. Pavón, J., Gómez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. [19] chapter IX 236–276
13. Molesini, A., Nardini, E., Denti, E., Omicini, A.: Situated process engineering for integrating processes from methodologies to infrastructures. In Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M., eds.: 24th Annual ACM Symposium on Applied Computing (SAC 2009). Volume II., Honolulu, Hawai'i, USA, ACM (2009) 699–706
14. Cernuzzi, L., Cossentino, M., Zambonelli, F.: Process models for agent-based development. *Engineering Applications of Artificial Intelligence* **18** (2005) 205–222
15. Kruchten, P.: The Rational Unified Process, An Introduction. Addison Wesley (1998)
16. Molesini, A., Omicini, A., Denti, E., Ricci, A.: SODA: A roadmap to artefacts. In Dikenelli, O., Gleizes, M.P., Ricci, A., eds.: Engineering Societies in the Agents World VI. Volume 3963 of LNAI. Springer (2006) 49–62
17. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* **17** (2008) 432–456
18. Rumbaugh, J.E.: Notation notes: Principles for choosing notation. *JOOP* **9** (1996) 11–14
19. Henderson-Sellers, B., Giorgini, P., eds.: Agent Oriented Methodologies. Idea Group Publishing, Hershey, PA, USA (2005)

Describing GORMAS using the FIPA Design Process Documentation and Fragmentation Working Group template

Sergio Esparcia, Estefanía Argente and Vicente Botti

Grupo de Tecnología Informática - Inteligencia Artificial
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n, 46022 Valencia, Spain
{sesparcia, eargente, vbotti}@dsic.upv.es

Abstract. This work presents a way to describe GORMAS, an Agent-Oriented Software Engineering methodology, using the template proposed by the FIPA Design Process Documentation and Fragmentation Working Group. This template uses SPEM 2.0 notation and it is aimed at identifying the fragments of each process, in order to extract and reuse them in some different processes.

Key words: agent-oriented software engineering, design process, fragmentation, GORMAS

1 Introduction

When developing Multi-Agent Systems (MAS), designers should be provided with methodologies and design processes that can help them to achieve well-designed systems. In the last years, the community of Agent-Oriented Software Engineering (AOSE) researchers has proposed several methods (see [1] for a survey on this topic). Some of the most recent methodologies are a refinement of previous methodologies, such as GORMAS [2], that refines INGENIAS [3], and ANEMONA [4] methodologies. Some others are the composition of different processes, such as MEnSA [5], that integrates concepts from Tropos [6], Gaia [7], SODA [8] and PASSI [9]. Therefore, it can be seen that designers of new methodologies make use of fragments from existing methods. It is necessary to be equipped with techniques that help designers to extract the fragments of a given design process. For example, the Situational Method Engineering (SME) [10] paradigm provides means for constructing ad-hoc software engineering processes following an approach based on the reuse of portions of existing design processes, the so-called method fragments stored in a repository, called method base. Each existing design process can be considered as composed of self-contained components, named fragments. Nowadays, a standard definition of fragment does not exist, so it is an open issue for designers, that have to decide what is a fragment in each method. Therefore, techniques for fragment selection and composition are required.

To give support in these topics, the IEEE FIPA Design Process Documentation and Fragmentation working group¹ (IEEE FIPA DPDF WG) is working on providing a solution in terms of a shared and easily adoptable specification for the documentation of the design process and of the process fragment. More in details, this working group aims to propose a definition of method fragment to be used during a situational method engineering process, the fundamental elements of which it is composed, and the metamodel on which it is based. The first step (currently undergoing) is the identification of the most suitable metamodel and notation for the process: (i) for the representation of the existing design processes from which the fragments have to be extracted; and (ii) for the representation of fragments themselves. This step will outcome in the definition of a proper template for the description of agent-oriented design processes. Such a template will, obviously, refer to the selected process metamodel and will suggest the adoption of good practices in documenting existing processes as well as defining new ones. The final step will be the definition of the Method Fragment Structure and Documentation Template. This template claims the authors to use SPEM 2.0 notation to describe their processes, in order to achieve a standardization. Currently, this FIPA group is working on specifications of the following methodologies: ADELFE [11], ASEME [12], ASPECS [13], INGENIAS [3], PASSI [9], SODA [8] and GORMAS [2].

This paper describes an application of the FIPA DPDF WG template to a specific MAS methodology. The main objective of this work is to describe the GORMAS methodology by means of the template proposed by the IEEE FIPA Design Process Documentation and Fragmentation working group. By using this template, we are looking to achieve the following:

- To facilitate the knowledge and diffusion of the GORMAS methodology by using a standardized description of the process.
- To make an assessment of the possibilities that the proposed template offers, evaluating its advantages and the changes that the document could need.
- To establish the fragments of the GORMAS process, that can be reused to improve other proposed processes.

The rest of this work is organized as follows. Section 2 describes the FIPA DPDF WG template. Section 3 gives an example of using the template with the GORMAS process. Section 4 presents a discussion on the proposed template. Finally, section 5 gives our conclusions on this work.

2 FIPA DPDF WG Template

In the same way that the Unified Modeling Language (UML) [14] does, the template [15], proposed by the FIPA DPDF WG to describe a process, identifies the fundamental concepts in the definition of design processes (regarding Agent-Oriented Systems) independently of the notation (text, icon, diagram, etc.) used for defining such concepts.

¹ <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg>

The design process documentation template proposed in this specification is also particularly relevant to researchers and practitioners working on Situational Method Engineering (SME) [10] approaches. SME proposes the reuse of fragments from known methods to obtain ad-hoc methods suitable for specific development situations. The method fragment (i.e. a portion of a design process) is the key-concept in SME and, although different definitions can be found for it, all of them share the idea of fragment as a self-contained component. Following this idea, in order to build a new process, designers have different previously defined fragments available that can be assembled [16, 17]. The method fragment process, must then be focused on the definition of these fragments, by requiring the whole process to be previously described in a standard way that makes identifying and defining them easier. Therefore, this must be the main aim of the FIPA DPDF WG specification, being important to provide here the means of defining the whole process from which fragments will be obtained.

The proposed template is suitable for process definition, since it has been conceived without referring to any specific process or methodology. Moreover, the template has a simple structure resembling a tree. This allows the definition to be given in a natural and progressive way. The proposed documentation is composed of three *main sections* (*Introduction*, *Phases of the Process*, and *Work Product Dependencies*). The *Introduction* section contains an *overview* of the process and a description of the *metamodel* used on it. The second section is split into as many subsections as phases that the process has. Every subsection contains a description of the *activities* executed in that phase, the *roles* that are involved into it and the *work products* that will be generated in this phase.

Finally, the template allows presumably easy use by process designers with a background on software engineering. It relies only on a few initial assumptions common in the field. Moreover, the notation suggested for documenting the process is the SPEM [18] standard with few extensions.

The next section uses this template for the GORMAS methodology.

3 Describing GORMAS with the FIPA DPDF WG template

The GORMAS methodology was completely described using the DPDF WG proposed template. This description can be found in a technical report [19] which is available on the web². Due to space limitations, in this section we will only describe the Mission Analysis phase of GORMAS using the proposed template.

3.1 Introduction

GORMAS (**G**uidelines for **OR**ganizational **M**ulti-**A**gent **S**ystems) defines a set of activities for the analysis and design of Virtual Organizations, including the design of their organizational structure and their dynamics. With this method,

² <http://www.dsic.upv.es/docs/bib-dig/informes/etd-05182010-133045/GORMASTechRep.pdf>

all services offered and required by the Virtual Organization are clearly defined, as well as its internal structure and the norms that govern its behavior.

GORMAS is based on a specific method for designing human organizations by Moreno-Luzón *et al.* [20], which consists of diverse phases for analysis and design. These phases have been appropriately adapted to the MAS field, to catch all the requirements of the design of an organization from the agents' perspective. Thus, the methodological guidelines proposed in GORMAS cover the typical requirement analysis, architectural and detailed designs of many relevant *Organization-Centered Multi-Agent Systems* (OCMAS) [21], (such as SODA and INGENIAS) methodologies, but it also includes a deeper analysis of the system as an open organization that provides and offers services to its environment.

3.2 Phases of the process

The proposed guideline allows being integrated into a development process of complete software, which may include the phases of analysis, design, implementation, installation and maintenance of MAS. GORMAS methodology is focused on the analysis and design processes, and it is composed of four phases (see Fig. 1), covering the analysis and design of a MAS: first phase is **mission analysis**, that involves the analysis of the system requirements, the use cases, the stakeholders and the global goals of the system; the **service analysis** phase specifies the services offered by the organization to its clients, as well as its behavior, and the relationships between these services; the **organizational design** phase defines the structure for the Virtual Organization, establishing the relationships and restrictions that exist in the system; and finally, at the **organization dynamics design** phase, communicative processes between agents are established, as well as processes that control the acquisition of roles along with processes that enable controlling the flow of agents entering and leaving the organization. Additionally, some norms that are used to control the system are defined. Finally, the organization dynamics design phase is responsible of designing guides that establish a suitable reward system for the organization. Implementation is carried out in the THOMAS [22] framework which mostly covers the organization software components that are required, such as organizational unit life-cycle management, service searching and composition and norm management.

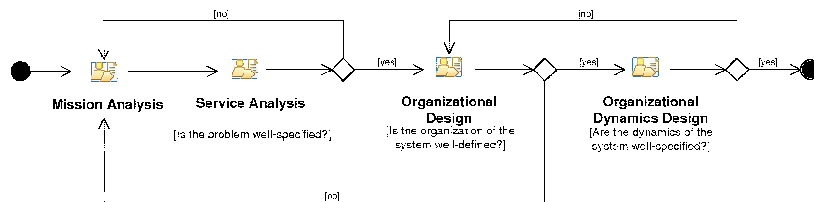


Fig. 1. GORMAS design process

This methodology allows designing large scale, open and service-oriented MAS, where organizations are able to accept external agents into them. In order

to model this kind of systems, GORMAS is supported by a CASE tool named EMFGormas [23], that uses the MDA Eclipse Technology.

3.3 Mission Analysis phase

The *Mission Analysis* phase, the first of the GORMAS methodology, involves the analysis of the system requirements, identifying the use cases, the stakeholders and the global goals of the system. This phase involves two different process roles, four work products (one model diagram and three text documents) and one guidance document, as described in figure 2. This phase is composed of five activities. The process flow at the level of activities is reported in figure 3.

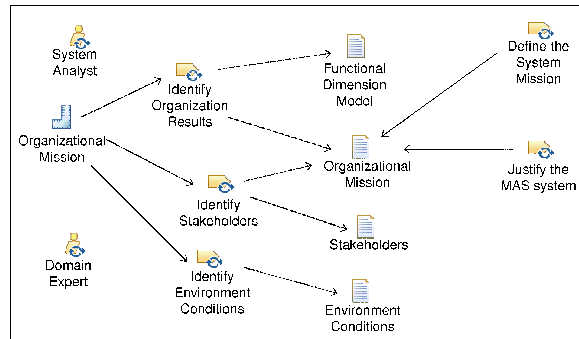


Fig. 2. Resources and products used in mission analysis phase

As a result of the activities carried out in this phase, a diagram of the *Functional Dimension Model* is drawn, detailing the products and services offered by the system, the global goals (mission) pursued, the stakeholders, the existing links between them, the system results and the resources or services needed.

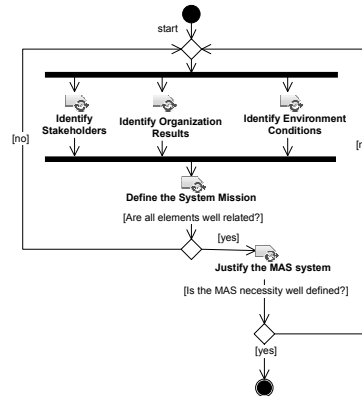


Fig. 3. Activity diagram of Mission Analysis phase

Process roles. There are two roles involved in the *Mission Analysis* phase: the System Analyst and the Domain Expert. The *System Analyst* is responsible for: (i) defining the mission and the context of the organization, by means of identifying the system results, the stakeholders and the environment of the organization; (ii) creating the documents that define the mission of the system; and (iii) defining the *Functional Dimension Model* diagram. The *Domain Expert* is responsible of supporting the system analyst during the *Mission Analysis* phase, by giving him all the information that he could need.

Activity details. In the Mission Analysis phase, the following concepts are defined:

- The global goals of the system (mission).
- The services and products that the system provides to other entities.
- The stakeholders with whom the system contacts (clients or suppliers of resources/services), describing their needs and requirements.
- The conditions of the environment or context in which the organization exists (i.e. complexity, diversity, restrictions, etc.).
- The justification of the existence of the MAS system that it is being designed, in order to look that the GORMAS definition on a MAS could contribute on defining an organization.

In order to identify all these items, five activities are needed, detailed in table 1. These activities are aimed at looking for the system mission, by means of: (i) identifying the organization results; (ii) identifying the stakeholders; and (iii) identifying the environment conditions. Moreover, global goals of the system are described. Finally, it is necessary to justify whether the GORMAS approach for creating organizations is suitable for the current problem under study.

Activity	Activity Description	Roles Involved
Identify organization results	Describe the results (products or services) that the system provides, to understand what the result is, what it does and who is interested in.	System analyst and domain expert
Identify stakeholders	Identify and describe the main stakeholders that the organization is related to (external actors, clients, users, etc.)	System analyst and domain expert
Identify environment conditions	Identify and define the kind of environment in which the organization will be developed, knowing if it is a physical environment or a virtual environment; if it is a distributed environment, etc.	System analyst and domain expert
Define the System Mission	Identify the global goals pursued by the system. These goals compose the mission of the organization.	System analyst and domain expert
Justify the MAS system	Justify the existence of this kind of system, comparing it to other existing similar systems (that can use agents or not), and analyzing the advantages and disadvantages, and the singularities of the proposed system.	System analyst

Table 1. Mission Analysis phase activities

Work products. The following section describes the products generated on the *Mission Analysis* phase. Firstly, the *Functional Dimension Model* diagram is defined, and three documents, related to organizational mission, stakeholders and environment conditions are generated (see table 2).

Fig. 4 describes their relation with the elements of the GORMAS metamodel [19]. In this figure, each of the work products reports one or more elements from the GORMAS meta-model; each MAS meta-model element is represented using a UML class icon and, in the documents, such elements can be Defined, reFined, Quoted, Related or Relationship Quoted, as explained in the template [15].

Name	Description	Work Product Kind
Functional Dimension Model	A diagram using the GORMAS graphical notation (based on GOPPR notation) that details the specific functionality of the system, based on services, tasks and goals.	Behavioral
Organizational Mission	A document describing the basic aspects of the organization that will be defined.	Structured text
Stakeholders	A document describing the stakeholders that will take part in the organization.	Structured text
Environment conditions	A document describing the conditions that the environment of the organization will have.	Structured text

Table 2. Product for Mission Analysis phase

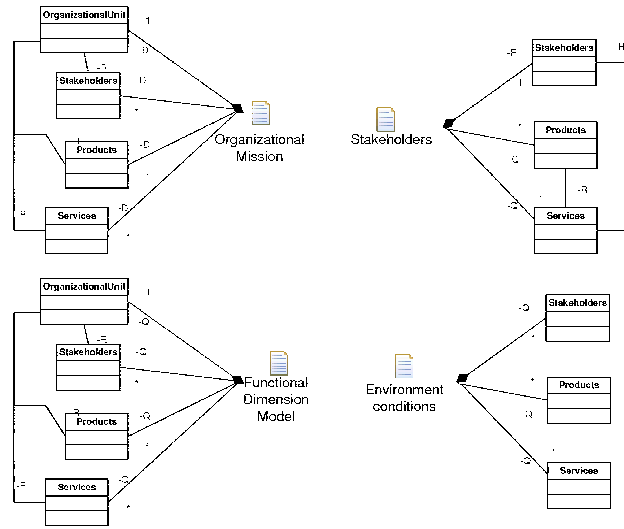


Fig. 4. Mission Analysis phase. Relations between work products and metamodel elements. Caption: D: defined element, introduced for the first time; F: element refined; Q: quoted element, already defined; R: element related with another element.

Organizational Mission. This document is employed to define the mission of the organization. It is a structured text document, whose template is shown in

table 3. As shown, it is necessary to give a name, a domain and an environment for the organization. Additionally, it is necessary to set the results that the system will provide and the stakeholders that are interested on keeping a relation with the organization. Finally, a justification for designing the system must be provided.

Organizational mission	
Name:	general name of the system or organization to be generated
Domain:	kind of market or interest area of the organization
Results:	set of products or services offered by the organization to its clients - <i>Purpose:</i> Description of the motivation because this result is offered. - Is it <i>tangible?</i> : If the result is storable, printable and/or reusable, it is a product. If it is a used up functionality, then it is a service.
Stakeholders:	actors that set up the market of the organization. - Is it a <i>consumer?</i> : the actor consumes the products or services that the organization provides. - Is it a <i>producer?</i> : the actor provides some resources or services that are required by the organization to work.
Kind of environment:	location of the system (unique or distributed): Ability to access to the real and physical world.
Context restrictions:	a set of restrictions that are imposed by the context or environment of the organization, and could affect to its structure, services, etc.
Justification:	reason of the existence of the organization - <i>Similar systems:</i> to detail the existing systems that provide a similar orientation than the one we are considering. - <i>Advantages:</i> set of advantages that we want to offer with the new proposal, i.e. optimal use of the resources or services. - <i>Disadvantages:</i> limitations that the new proposal has. - <i>Singularities:</i> competitive elements of the organization.

Table 3. Template for Organizational Mission document

Stakeholders. This document is employed to describe the stakeholders of the organization, that have been defined in the Organizational Mission document. It is also a structured text document, whose template is shown in table 4. The identification of the stakeholders must be completed by providing the kind of stakeholder, the objectives that each group follows, their products and services provided and required, the benefits obtained by them and their position into the organization.

Stakeholders	
Name	An identifier for the stakeholder
Beneficiary	Indicate if the stakeholder is a primary (essential) or a secondary beneficiary.
Type	Indicate if the stakeholder is a client, a provider or a regulator.
Objectives	Describe the objectives pursued by the stakeholder.
Requires	A set of products and/or services that the stakeholder consume.
Provides	A set of products and/or services that the stakeholder offers to the organization.
Frequency	To point out if this stakeholder contacts the organization frequently, occasionally or in an established period of time.
Benefits	Describe the benefits that the stakeholder wants to achieve.
Decision power	Indicate if their needs are affecting to the requirements of products or services.
Under the influence of the system?	Indicate if the organization can affect the interests of the stakeholder.
Contribution	To point out what the organization obtains for its relationship with stakeholders.

Table 4. Stakeholders document

Environment conditions. This document is employed to describe the environmental conditions in which the organization will be placed. It is a structured text document, whose template is shown in table 5. This document analyzes five

conditions: the change rate, the complexity, the uncertainty, the receptivity and the diversity of an environment.

Environment conditions
Change rate: Are the stakeholders constants through time? Are their requirements constants? Are they modified in a cyclical and a predictable way? Is it possible to estimate the consumption of a product? Is the demand of a product or a service constant through time? If the answer is affirmative, the environment is stable. If not, it is an unstable or dynamic environment.
Complexity: Is there a lot of different elements? Are there a lot of clients? Are there a lot of types of products and services to offer? Are there a lot of types of providers? Are providers not related between them? If any of the answers is affirmative, the environment is complex. If not, it is a simple environment.
Uncertainty: If the environment is dynamic and complex, uncertainty is high. If the environment is stable and simple, uncertainty is low.
Receptivity: Are the inputs and resources available? Are they obtained in an easy and secure way? If the answer is affirmative, the environment is munificent. If not, it is an hostile environment.
Diversity: Are different groups of clients served? Is it provided a set of different products or services, with no relationship between them? If any of the answers is affirmative, the environment is diverse. If not, it is a uniform environment.

Table 5. Environment Conditions document

Functional Dimension Model. This work product is a GORMAS diagram. GORMAS uses a UML-like graphical notation called GOPPR [24] (used to define diagrams in INGENIAS and ANEMONA methodologies), but adding some entities proposed by GORMAS like services and norms.

As stated before, the *Functional Dimension Model* details the specific functionality of the system, based on services, tasks and goals, as well as system interactions. Figure 5 shows an example of a *Functional Dimension Model* diagram. An Organizational Unit representing the system (UPV), as well as the stakeholders (Students, Governing organs and Teachers), the objectives pursued by the organization ('Efficient management of the financial resources', 'Increase scientific production'), the products (Databases and Bills) and the services (Budget mnt. and PhD mnt.) of the system are defined on this diagram.

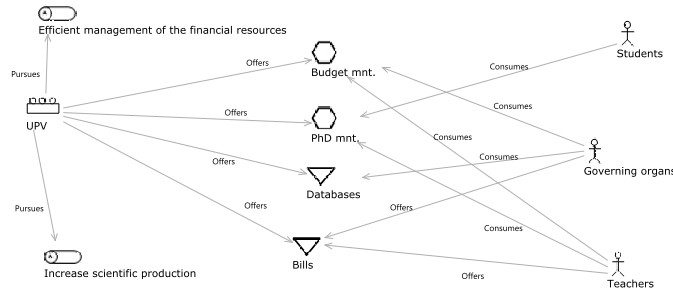


Fig. 5. Example of a *Functional Dimension Model* diagram

4 Discussion

This section describes the advantages and disadvantages presented by the proposed template. On the one hand, the advantages found during this work are:

- This template allows describing a development process using a standard language like SPEM 2.0. The use of a standard will improve the comparison of GORMAS with other methodologies.
- The phases of the GORMAS methodology were clearly depicted in its original definition. Therefore, a correct fragmentation of the process was easily obtained after applying the template to GORMAS.
- The metamodel used by the GORMAS methodology has its own section inside the template, providing a clear and easy-reading description of it.
- The activities developed inside every phase of GORMAS can be easily defined. Therefore, the methodology is not only described in an overview, but in a detailed way.
- The work products are described and an example is given. Additionally, their evolution during the whole process is shown and their dependencies.
- The roles participating in the process are identified and the activities that they are responsible for are identified.
- Several tables are used in order to improve the identification of some elements such as the work products and the activities of the methodology.

On the other hand, these are the disadvantages of this template:

- The *Phases of the process* section of the template describes the functionality of every phase by means of its activities, and the tasks composing each activity. Activities are provided with a deeper description than tasks. However, in the *Organizational Dynamics Design* phase of our example, tasks are more important than activities as they contain the most relevant information related to this phase. In order to solve this problem, we defined the tasks of *Organizational Dynamics Design* phase using the best possible way that the template guidelines allowed us, by means of descriptive tables. Designers should be allowed to describe tasks in a similar way to activities, i.e. with a similar deep description, so as to achieve the desired level of detail in every phase of a process.
- The template has not a discussion nor conclusion section. It can be very useful to add a section like this, in order to provide developers with the possibility of remarking or pointing out some features that could be considered as important. Additionally, a section describing guidance documents can be very interesting, in which work products were described.
- The template provides a deep and detailed description of a design process, but it does not include a 'light view' of the process. This feature reduces the focus of possible future readers, that will be mainly bounded to process designers and developers.
- The adaptation of a methodology from its classic representation to the representation proposed by the template must be done by a human. That is, we must know how to deal with human errors, that could make a methodology not to be properly translated to the structure of the template.

As seen, there are more interesting advantages than disadvantages on using this template. The use of a well-known standard graphic language such as SPEM

2.0 drives the methodologies to a better understanding of their processes and approaching them to the standardization. Moreover, the usage of this template will improve the task of identifying and extracting the fragments of a methodology. These fragments will be available for the rest of the AOSE community, that would use them to improve their existing methodologies or to create new design processes based on a compilation of these fragments.

Possibly, the main lack of AOSE is a standard methodology. Using this template, all the methodologies will adopt the same structure, allowing the authors to compare them to test every feature of the processes, in order to find which is the methodology that best solves a concrete problem. After this analysis, there will be possible to define a methodology, taking the best from the existing methodologies, that could be considered as a standard.

5 Conclusion

This work presents the template to describe an AOSE methodology proposed by the IEEE FIPA DPDF WG. As an example, the GORMAS methodology, used to describe Virtual Organizations, was described using this template. After using this template over a concrete design process, it becomes expressed in a standard notation such as SPEM 2.0 and their fragments are identified. In our concrete example, GORMAS became properly translated into the structure of the template. Nevertheless, there could be some methodologies whose adaptation to the template is harder. As a conclusion, we can state that the template allows to describe a methodology in a proper way and it is recommended to adopt it to improve the understanding and study of the design processes.

Acknowledgments. This work is supported by TIN2009-13839-C03-01, TIN2008-04446 and PROMETEO/2008/051 projects of the Spanish government and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. E. Argente, G. Beydoun, R. Fuentes-Fernandez, B. Henderson-Sellers, G. Low, and F. Migeon. Modelling with agents. In *Postproceedings of AOSE 2009*, page In Press. Springer, 2010.
2. E. Argente, V. Botti, and V. Julian. Organizational-Oriented Methodological Guidelines for Designing Virtual Organizations. In *Proc. 10th International Work-Conference on Artificial Neural Networks*, page 162. Springer, 2009.
3. J. Pavón, J. Gómez-Sanz, and R. Fuentes. The INGENIAS methodology and tools. *Agent-Oriented Methodologies*, pages 236–276, 2005.
4. V. Botti and A. Giret. *Anemona. A Multi-agent Methodology for Holonic Manufacturing Systems*. Springer Series in Advanced Manufacturing. Springer, 2008.
5. R. Ali, V. Bryl, G. Cabri, M. Cossentino, F. Dalpiaz, P. Giorgini, A. Molesini, A. Omicini, M. Puviani, and V. Seidita. MEnSA Project - Methodologies for the Engineering of complex Software systems: Agent-based approach. Technical Report 1.2, UniTn, 2008.
6. P. Bresciani, P. Grogini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems*, 8:203–236, 2004.

7. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems. In *1st Int. Workshop on Agent-Oriented Software Engineering*, pages 127–141, 2000.
8. A. Omicini. SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems. *Agent-Oriented Software Engineering*, 1957:185–193, 2001.
9. M. Cossentino. From requirements to code with the PASSI methodology. *Agent-oriented methodologies*, pages 79–106, 2005.
10. S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Inf. Softw. Technol.*, 38(4):275–280, 1996.
11. C. Bernon, V. Camps, M.P. Gleizes, and G. Picard. Engineering adaptive multi-agent systems: The ADELFE methodology. *Agent-oriented methodologies*, pages 172–202, 2005.
12. N. Spanoudakis and P. Moraitis. The Agent Systems Methodology (ASEME): A Preliminary Report. In *Proc. 5th European Workshop on Multi-Agent Systems*, 2007.
13. M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, 2010.
14. J. Rumbaugh. Notation notes: Principles for choosing notation. *Journal of Object-Oriented Programming*, 9(2):11–14, 1996.
15. M. Cossentino. Design Process Documentation Template. Technical report, FIPA, 2010.
16. J. Ralyté and C. Rolland. An assembly process model for method engineering. In *Advanced information systems engineering*, pages 267–283. Springer, 2001.
17. D.G. Firesmith and B. Henderson-Sellers. *The OPEN process framework: An introduction*. Addison-Wesley Professional, 2002.
18. V. Seidita, M. Cossentino, and S. Gaglio. Using and extending the SPEM specifications to represent agent oriented methodologies. In *9th Int. Workshop on Agent Oriented Software Engineering (AOSE), Estoril, Portugal*, 2008.
19. S. Esparcia, E. Argente, V. Julian, and V. Botti. GORMAS: A methodological guideline for organizational-oriented MAS. Technical report, Universidad Politécnica de Valencia, 2010.
20. M.D. Moreno-Luzón, F.J. Peris Bonet, and T.F. González Cruz. *Gestión de la calidad y diseño de organizaciones*. Pearson Educación, SA, 2001.
21. E. Argente, A. Giret, S. Valero, V. Julian, and V. Botti. Survey of MAS Methods and Platforms focusing on organizational concepts. In *Recent Advances in Artificial Intelligence Research and Development*, volume 113 of *Frontiers in Artificial Intelligence and Applications*, pages 309–316. IOS Press, 2004.
22. A. Giret, V. Julian, M. Rebollo, E. Argente, C. Carrascosa, and V. Botti. An open architecture for service-oriented virtual organizations. In *PROMAS 2009 Post-Proceedings*, pages 1–15. Springer, 2010.
23. E. Garcia, E. Argente, and A. Giret. A modeling tool for service-oriented open multiagent systems modeling tool. In *Proc. 12th Int. Conf. on Principles of Practice in Multi-Agent Systems*, volume 5925 of *LNAI*, pages 345–360. Springer-Verlag, 2009.
24. S. Kelly, K. Lyytinen, and M. Rossi. MetaEdit+: A fully configurable multi-user and multi-tool CASE and CAME environment. *Lecture Notes in Computer Science*, 1080:1–21, 1996.

The O-MaSE Process: a Standard View

Juan C. Garcia-Ojeda¹ and Scott A. DeLoach²

¹ Facultad de Ingenieria de Sistemas, Universidad Autonoma de Bucaramanga,
Calle 48 No 39-234, El Jardin. Bucaramanga, Santander - Colombia
jgarciao@unab.edu.co

² Kansas State University, 234 Nichols Hall, Manhattan, Kansas USA
sdeloach@k-state.edu

Abstract. Method engineering has widely been proposed as an approach to delivering industrial strength software development processes to spur the adoption of agent-based software in industry. The Foundation for Physical Agents Technical Committee (FIPA-TC) Methodology group is currently attempting to define a template for documenting process fragments. This paper presents our experience in applying the template for the Organization-based Multiagent System Engineering methodology

Keywords: agent-oriented software engineering, method engineering, design process documentation, software processes

1 Introduction

Today's software industry is tasked with building ever more complex software applications. Businesses today are demanding applications that operate autonomously, adapt in response to dynamic environments, and interact with other distributed applications in order to provide wide-ranging solutions [8,10]. Multiagent system (MAS) technology is a promising approach capable of meeting these new demands [10]. Unfortunately, there is a disconnect between the advanced technology being created by the multiagent community and its application in industrial software. The obstacles to industrial adoption have been the focus of several discussions. Jennings, Sycara, and Wooldrige [8] mention two major obstacles to widespread adoption of agent technologies in the industry: (1) the lack of complete processes to help designers to specify, analyze, and design agent-based applications, and (2) the lack of industrial-strength agent-based toolkits. To overcome this situation, several MAS researchers and engineers have suggested the use of method engineering [1,2,9,11]. Method engineering is a discipline in which process engineers construct processes (i.e., methodologies) from a set of existing method fragments.

In a related effort, the Foundation for Physical Agents Technical Committee (FIPA-TC) Methodology group is currently attempting to define reusable method fragments from existing agent-oriented processes [13]. As part of this effort, the group is currently defining a Design Process Documentation Template (DPDT)

specification, which uses SPEM 2.0 as its base [5]. In this paper, we present our experience of applying the DPDT guidelines for the Organization-based Multiagent System Engineering (O-MaSE) methodology. After discussing background material in Section 2, we present a partial definition of O-MaSE following the DPDT in Section 3 followed by a discussion of our experiences with the template in Section 4.

2 Background

Method Engineering is an approach where process engineers construct processes (i.e., methodologies) from a set of method fragments as opposed to modifying or tailoring monolithic, “one-size-fits-all” processes to suit their needs. Method fragments are generally created by extracting useful tasks and techniques from existing processes and redefining them in terms of a common metamodel. The fragments are then stored in a repository for later use. During creation, process engineers select suitable method fragments from the repository and assemble them into complete processes meeting project specific requirements [1].

The Software and Systems Process Engineering Meta-model (SPEM) is “a process engineering meta-model as well as conceptual framework, which can provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting developments processes” [12]. SPEM distinguishes between reusable *method content* and the way it is applied in actual *processes*. SPEM method content captures and defines the key Tasks, Roles, and Work Products¹ in a software development processes. Essentially, Tasks are performed by Roles, taking a set of Work Products as inputs and producing set of Work Products as its output.

Development processes are assembled into a set of Activities, populated with Tasks and their associated Roles and Work Products. Thus, Activities are aggregates of either basic content or other Activities. SPEM defines three special types of Activities: Phases, Iterations and Processes. Phases are special Activities that take a period of time and end with a major milestone or set of Work Products. Iterations are Activities that group other Activities that are often repeated. Finally, Processes are special Activities that specify the structure of a software development project.

2.2 FIPA Design Process Documentation Template Specification

The *Design Process Documentation Template* specification [5] introduces a set of guidelines whose goal is the identification of the fundamental concepts in the design of agent-oriented design processes independent of the notation (text, icons, diagrams, etc.). This specification follows the situational method engineering approach for reusing fragments from known methods to obtain custom methods suitable for specific development situations. For designing agent-oriented processes, the specification suggests the use of a process documentation template. The template

¹ SPEM 2.0 defines as Method Content with Task Uses, Role Uses, and Work Product Uses as instances of Task Definitions, Role Definitions, and Work Product Definitions in actual processes. This paper refers to both forms as Tasks, Roles or Work Products.

guides process designers to build processes by documenting three main sections: Introduction, Phases of the Process, and Work Product Dependencies. The goal of the *Introduction* is: (i) to introduce the scope and limits of the process, (ii) organize the design process phases according to the selected lifecycle and, (iii) to provide a complete description of the MAS metamodel adopted in the process with the definition of its composing elements. The aim of the *Phases of the Process* is to detail the whole process by decomposing it on the basis of workflows at different levels of granularity (phase-activity-task). Finally, the *Work product dependencies* represent the dependencies between the work products and thus (indirectly) between the activities that produce them. Finally, the template suggests the adoption of SPEM 2.0 as the standard for modeling *some* design process aspects.

3 Applying the DPDT to O-MaSE

In this section, we present a partial definition of O-MaSE using the DPDT. Due to page length considerations, we show selected diagrams with abbreviated descriptions.

3.1 Introduction

O-MaSE is a new approach in the analysis and design of agent-based systems, being designed from the start as a set of method fragments to be used in a method engineering framework [7]. The goal of O-MaSE is to allow designers to create customized agent-oriented software development processes. O-MaSE consists of three basic structures: (1) a metamodel, (2) a set of methods fragments, and (3) a set of guidelines. The O-MaSE metamodel defines the key concepts needed to design and implement multiagent systems. The method fragments are tasks that are executed to produce a set of work products, which may include models, documentation, or code. The guidelines define how the method fragments are related to one another.

The aT³ Process Editor (APE) [6] supports the creation of custom O-MaSE compliant processes [6]. APE has five key elements: a Method Fragment Library, the Process Editor, a set of Task Constraints, a Process Consistency checker, and a Process Management tool. The *Library* is a repository of O-MaSE method fragments, which can be extended by APE users. The *Process Editor* allows users to create and maintain O-MaSE compliant processes. The *Task Constraints* view helps process engineers specify Process Construction Guidelines to constrain how tasks can be assembled, while the *Process Consistency* mechanism verifies the consistency of custom processes against those constraints. Finally, the *Process Management tool* provides a way to measure project progress using the custom process.

The O-MaSE Lifecycle. As O-MaSE was designed as a set of fragments to be assembled by developers to meet their specific requirements, it does not actually commit to any specific set of phases. This is a major difficulty with trying to map O-MaSE to the DPDT. To alleviate this problem, we assume we are following a traditional waterfall approach shown in Figure 1. There are three main Phases: Requirements Analysis, Design, and Implementation, with the main Activities

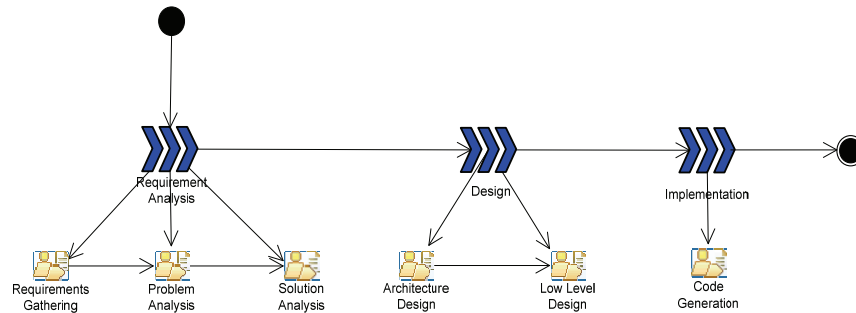


Figure 1. Using Waterfall Phases with O-MaSE

allocated as expected. When actually using O-MaSE, the process designer must define their own set of phases and iterations and then assign Activities and Tasks to those phases and iterations. As this will be unique for each system being developed, there are no hard and fast rules on what activities should be placed in which phase.

The O-MaSE Metamodel. The O-MaSE metamodel defines the main concepts and relationships used to define multiagent systems. The O-MaSE metamodel is based on an organizational approach and includes notions that allow for hierarchical, holonic, and team-based decomposition of organizations. The O-MaSE metamodel was derived from the Organization Model for Adaptive Computational Systems (OMACS). OMACS captures the knowledge required of a system's organizational structure and capabilities to allow it to organize and reorganize at runtime [3]. The key decision in OMACS-based systems is which agent to assign to which role in order to achieve which goal. As shown in Figure 2, an Organization is composed of six entities: Goals, Roles, Agents, Organizational Agents, a Domain Model, and Policies, which are defined in Table 1.

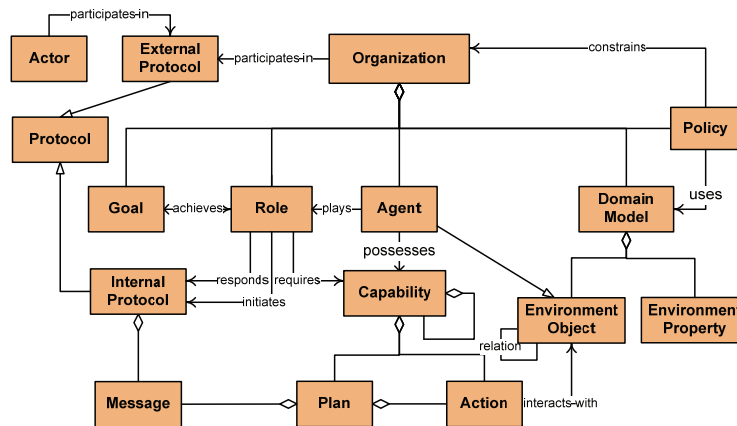


Figure 2. O-MaSE Metamodel

Table 1. Metamodel Entities

Goal	Goals are a desirable state; goals capture organizational objectives
Role	Roles capture behavior that achieves a particular goal or set of goals
Agent	Agents are autonomous entities that perceive and act upon their environment; agents play roles in the organization
Capability	Capabilities capture soft abilities (algorithms) or hard abilities of agents
Domain model	Captures the environment including objects and general properties describing how objects behave and interact
Policy	Policies constrain organization behavior often in the form of liveness and safety properties
Protocol	Protocols define interaction between agents, roles, or external Actors; they may be internal or external
External actor	External Actors exist outside the system and interact with the system
Plan	Plans are abstractions of algorithms used by agents; plans are specified in terms of actions with the environment and messages in protocols

3.2 Phases

As a reminder, the phases presented here are not actually part of the O-MaSE definition, but only included to help define O-MaSE according to the DPDT.

Requirements Analysis. In traditional software engineering practice, the requirement analysis phase attempts to define and validate requirements for a new or modified software product, taking into account the views of all major stakeholders. A generic example of an O-MaSE requirements analysis phase is shown in Figure 3.

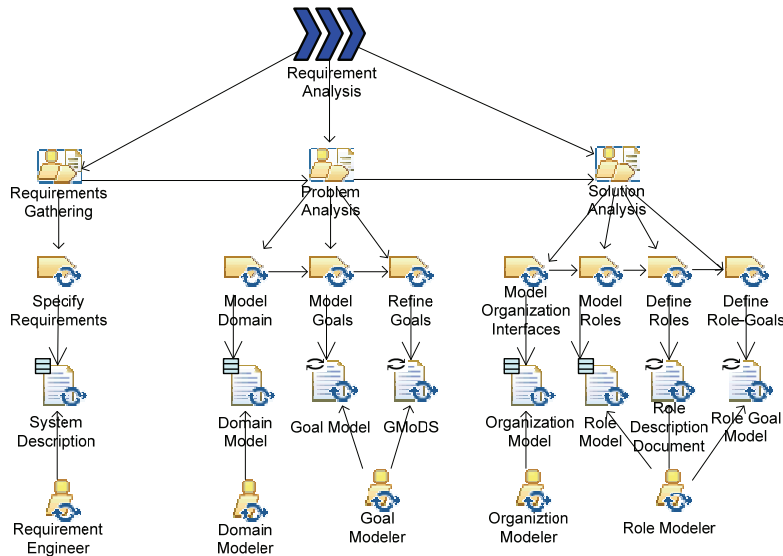


Figure 3. Requirements Analysis Phase

Process Roles. This phase uses five roles: Requirements Engineer, Goal Modeler, Domain Modeler, Organization Modeler, and Role Modeler. The *Requirements Engineer* captures and validates the requirements of the system. Thus, the person in this role must be able to think abstractly, work at high-levels of abstraction, and be able to collaborate with stakeholders, domain modelers, and project managers. The *Goal Modeler* is responsible for the generation of the GModS goal model. Thus, Goal Modeler must understand the system description/SRS, be able to interact openly with various domain experts and customers, and must be proficient in GModS AND/OR Decomposition and ATP Analysis [4]. The *Domain Modeler* captures the key concepts and vocabulary in the current and envisioned environment of the system, helping to further refine and validate requirements. The *Organization Modeler* is responsible for documenting the Organization Model. Thus, the Organization Modeler must understand the system requirements, Goal Model, and the Domain Model and be skilled in organizational modeling techniques. The Role Modeler creates the Role Model and the Role Description work products, which requires knowledge of the role model specification, and a general knowledge of the system.

Activity Details. In the Requirements Analysis phase, there are three activities: Requirements Gathering, Problem Analysis, and Solution Analysis. Requirements Gathering is the process of identifying software requirements from a variety of sources. Typically, requirements are either functional requirements, which define the functions required by the software, or non-functional requirements, which specify traits of the software such as performance quality, and usability. Problem Analysis captures the purpose of the product and documents the environment in which it will be deployed. O-MaSE captures this information in a Goal Model, which captures the purpose of the product, and a Domain Model that captures the environment in which the product exists. Finally, Solution Analysis defines the required system behavior based on the goal and domain models. The end result is a set of roles and interactions in the Organization Model.

Work product kinds. There are six work products produced in the Requirements Analysis phase: System Description Specification, Goal Model, GModS Model, Domain Model, Organization Model, and Role Model as defined in Table 2.

Table 2. Requirements Analysis Work Products

Name	Description	Work Product Kind
System Description Specification	describes the technical requirements for a particular agent-oriented software	Textual
Goal Model	captures the purpose of the organization as a goal tree; includes goal attributes, precedence relationships and triggering relationships	Behavioral
Organizational Model	documents the interaction between the organization and the external actors	Structural
Role Model	depicts organization roles, the goals they achieve and interactions between roles/external actors	Behavioral

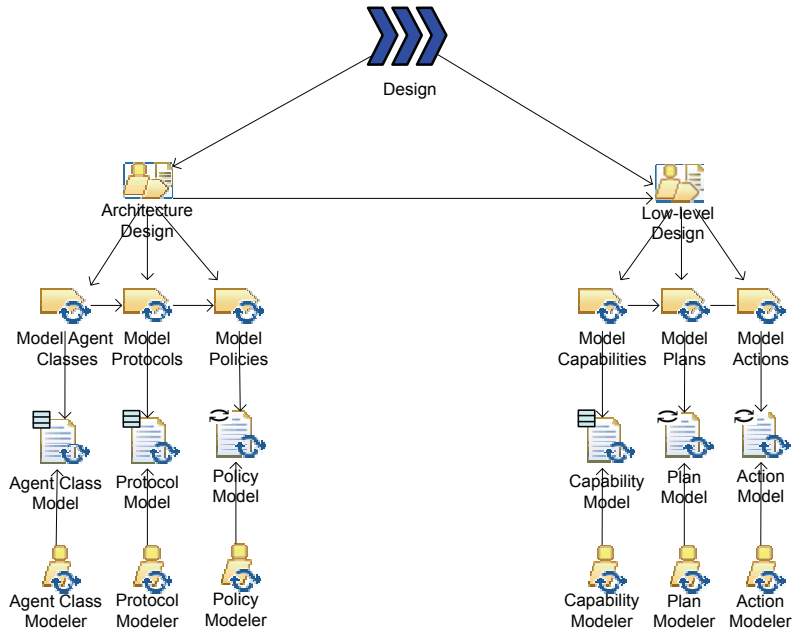


Figure 4. Design Phase

Design. The design phase consists of two activities: Architecture Design and Low Level Design. Once the goals, the environment, the behavior, and interactions of the system are known, *Architecture Design* is used to create a high-level description of the main system components and their interactions. This high-level description is then used to drive *Low Level Design*, where the detailed specification of the internal agent behavior is defined. This low-level specification is then used to implement the individual agents during the Implementation phase (see Figure 4).

Process Roles. There are six roles in the design phase: Agent Class Modeler, Protocol Modeler, Policy Modeler, Capability Modeler, Plan Modeler, and Action Modeler. The *Agent Class Modeler* is responsible for creating the Agent Class Model and requires general modeling skills and knowledge of the O-MaSE Agent Class Model specification. The *Protocol Modeler* designs the protocols required between agents, roles, and external actors and requires protocol modeling skills. The *Policy Modeler* is responsible for designing the policies that govern the organization. The *Capability Modeler* is responsible for defining the Capability Model and requires modeling skills and O-MaSE Capability Model specification knowledge. The *Plan Modeler* designs the plans necessary to play a role; required skills include understanding of Finite State Automata and O-MaSE Plan Model specification knowledge. Finally, the *Action Modeler* documents the Action Model, which requires the ability to specify appropriate pre- and post-conditions for capability actions.

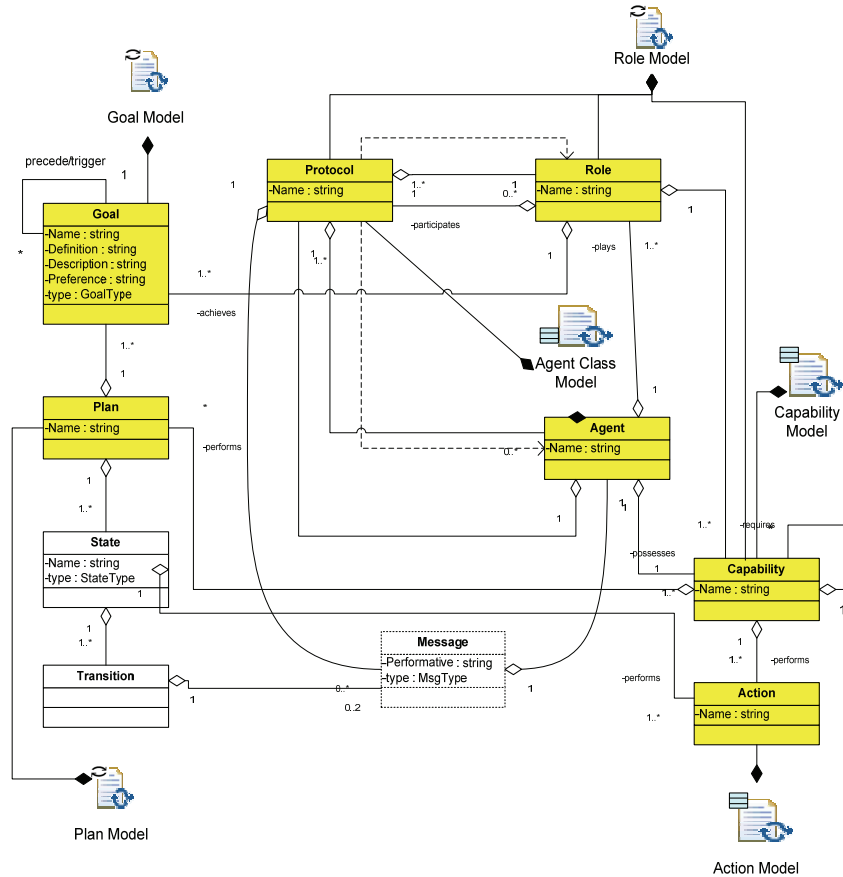


Figure 5. Relationship among several Work Products and the Metamodel

Activity Details. In the Architecture Design we focus on documenting the different agents, protocols, and policies using three tasks: Model Agent Classes, Model Protocols, and Model Policies. The *Model Agent Classes* task identifies the types of agents that may participate in the organization. Agent classes may be defined to play specific roles, or they may be defined in terms of capabilities, which implicitly define the types of roles that may be played. The *Model Protocols* task defines the details of the interactions between agents or roles. The Protocol Model produced defines the types of messages sent between the two entities. Finally, the *Model Policies* task defines a set of formally specified rules that describe how an organization may or may not behave in particular situations. During the organization design, the Policy Modeler captures the desired system properties and documents them in a formal or informal notation.

Table 3. Requirements Analysis Work Products

Name	Description	Work Product Kind
Agent Class Model	defines the agent classes and sub-organizations that will populate the organization.	Structural
Protocol Model	represents the different relations/interaction between external actors and agents/roles.	Structural
Policy Model	describes all the rules/constraints of the system	Behavioral
Capability Model	defines the internal structure of the capabilities possessed by agents in the organization.	Structural
Plan Model	captures how an agent can achieve a specific type of goal using a set of actions (which includes sending and receiving messages).	Behavioral
Action Model	defines the low-level actions used by agents to perform plans and achieve goals.	Structural

In the Low-level design we focus on the capabilities possessed by, actions performed by, and plans followed by agents. The *Model Capabilities* task defines the internal structure of the capabilities possessed by agents in the organization, which may be modeled as an Action or a Plan. An *action* is an atomic functionality possessed by an Agent and defined using the *Model Actions* task. A *plan* is an algorithmic definition of a capability and is defined using the *Model Plans* task. The *Model Plans* task captures how an agent can achieve a specific type of goal using a set of actions specified as a Plan Model (a Finite State Machine). Finally, the *Model Actions* task defines the low-level actions used by agents to perform plans and achieve goals. Actions are typically defined as a function with a signature and a set of pre and post-conditions. In some cases, actions may be modeled by providing detailed algorithmic information. Figure 5 shows the relationship between some work products (i.e., Goal Model, Role Model, Agent Class Model, Capability Model, Plan Model, and Action Model) and the entities used to design a typical system. Notice for instance, that the Goal Model defines goals; the Capability Model defines capabilities, while the Role Model uses those goals and capabilities to define roles and protocols in the Role Model. Likewise, the Plan Model defines plans in terms of actions defined by the Action Model.

Work product kinds. There are six work products produced in the Design phase: Agent Class Model, Protocol Model, Policy model, Capability Model, Plan Model, and Action Model as defined in Table 3.

Implementation. Finally, the design is translated to code. The purpose of this phase is to take all the design models created during the design and convert them into code that correctly implements the models. Obviously, there are numerous approaches to code generation based on the runtime platform and implementation language chosen. In this phase there is a single Role, the *Programmer* who is responsible for writing code based on the various models produced during the Design phase. The output of the *Generate Code* task is the source code of the application. While not currently covered in the process, system creation ends with testing, evaluation, and deployment of the systems.

Table 4. Work Product Dependencies

<i>Work product</i>	<i>Work product Kind</i>	<i>Dependency</i>
System Description	Structural	None
Goal Model	Behavioral	System Description
GMoDS	Behavioral	System Description, Goal Model
Domain Model	Structural	System Description
Organization Model	Structural	System Description
Role Model	Structural	GMoDS, Organization Model
Role Description Document	Behavioral	Role Model
Role-Goal Model	Behavioral	GMoDs, Role Model
Agent Class Model	Structural	Organization Model, GMoDS
Protocol Model	Structural	Role Model, Agent Class Model
Plan Model	Behavioral	Role Model, Agent Class Model, GMoDS
Policy Model	Behavioral	Agent Class Model, Role Description Document, GMoDS
Capability Model	Structural	Domain Model, Agent Class Model, Role Model
Action Model	Behavioral	Domain Model, Capability Model
Code	Composite	Capability Model, Action Model

3.3 Workproduct Dependencies

Table 4 shows the dependencies between the different work products in O-MaSE. These dependencies characterize different pieces of information or physical entities produced during the different stages of the development process and serve as inputs to and outputs of work units (i.e., either activities or tasks). Also, each work product is specified in terms of the kind of model/information/data documented. For instance, a structural work product is used to model static aspects of the system. In turn, a behavioral work product is used to model dynamic aspects of the system. Finally, a composite work product is used to model both static and dynamic aspects of the system (for further details on the different work product kind's see [13]). Figure 6 presents the dependencies between the various O-MaSE work products.

4 Conclusions and Future Work

In this paper, we presented a part of the O-MaSE documentation produced by following the DPDT specification. To be able to fit into the DPDT template, we had to select an example set of phases, which we did base on a simple waterfall approach. Then, we proceeded to document the different phases of our simple process in terms of the different activities, tasks, roles, and work products.

Based on our experience, we do not believe that requiring the process to be defined in terms of phases is necessarily the best approach. While we were able to use a simple waterfall model and describe our activities and tasks as if they were all be used

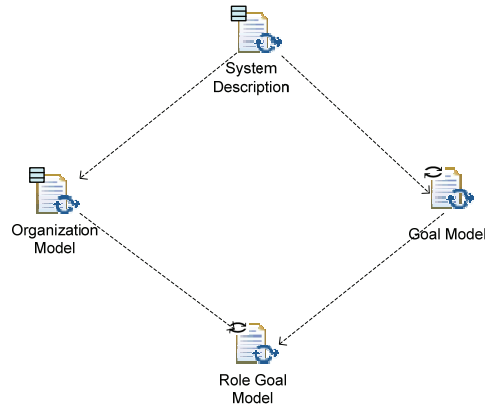


Figure 6. Requirement Analysis Phase’s Work Product Dependencies

in a straightforward, sequential manner, this might not always be the case. Specifically, it is unclear how to document activities and tasks that might take differing approaches and thus would likely be incompatible within the same process. In addition, forcing O-MaSE into any predefined set of phases masks the flexibility of the general approach proposed in O-MaSE.

We also believe it to be the case that the DPDT was defined assuming that fragments would be defined at the Activity level. However, when we create custom O-MaSE compliant processes, we generally use fragments at the Task level. Since Tasks are broken down only within an Activity and exist as rows in a table, there is not a natural mechanism for referring to them other than to call them by name and refer to the Activity in which they are defined. This breakdown also makes it difficult to document tasks that can be used in more than one Activity. For example, we have a Model Protocols task that is nominally defined in the Architecture Design activity. However, we can also model protocols within the Solution Analysis activity as well. Actually, you can Model Protocols in five ways: between organizations and external actors, between external actors and roles, between external actors and agents, between roles and roles, and between agents. We could make five copies of the Model Protocols task and specify the Work Product inputs slightly differently in each case; however, this seems redundant. In our original O-MaSE definition, we have one task called Model Protocols that has several optional Work Product inputs.

Although we believe the DPDT specification is headed in the right direction by supporting the construction of custom agent-based processes, there is considerable work to do before the DPDT will make an impact on industrial acceptance. While the DPDT will allow fragments to be documented in a common format, this is not useful unless tools for creating, maintaining, and transforming fragments are developed. While APE is an initial step in this direction, additional effort should be pursued to further support industrial acceptance.

Specifically, taking the DPDT as a starting point, research should be performed to extend this work to (i) develop qualitative and quantitative methods for helping process designers in creating custom processes based on the functional, non-

functional, and general architectures of new systems; (ii) formalize process guidelines in order to avoid ambiguities between the metamodel and the method fragments used to assemble agent-oriented applications, (iii) provide a set of guidelines on how to integrate different agent-oriented metamodels.

References

1. Brinkkemper, S. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*. 38(4) 1996, pp 275-280.
2. Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardization to research. *Intl Jnl of Agent-Oriented Software Engineering*, 1(1), 91–121, 2007.
3. DeLoach, S.A., Oyenon, W., Matson, E.T. A capabilities based model for artificial organizations. *Autonomous Agents and Multiagent Systems*. 16(1), 13-56, 2008.
4. DeLoach, S.A., and Miller, M. A Goal Model for Adaptive Complex Systems. *International Journal of Computational Intelligence: Theory and Practice*. Volume 5, no. 2, 2010. (in press).
5. FIPA Design Process Documentation and Fragmentation Working Group. Design Process Documentation Template (08-06-2010). <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/>.
6. Garcia-Ojeda, J.C., DeLoach, S.A. Robby. agentTool process editor: supporting the design of tailored agent-based processes. In *Proc. of the 2009 ACM Symp on Applied Computing*, ACM: New York, 2009.
7. Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenon, W.H.,Valenzuela, J. O-MaSE: a customizable approach to developing multiagent development processes. In M. Luck (ed.), *Agent-Oriented Software Engineering VIII: The 8th Intl Workshop on Agent Oriented Software Engineering (AOSE 2007)*, LNCS 4951, 1-15, Springer: Berlin.
8. Jennings, N. R., Sycara, K., Wooldridge, M. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems* 1(1) 7-38, 1998.
9. Low, G., Beydoun, G., Henderson-Sellers, B., Gonzalez-Perez, C. Towards method engineering for multi-agent systems: a validation of a generic MAS metamodel. In *10th Pacific Rim Intl Conf on Multi-Agent Systems, PRIMA 2007, Bangkok, Nov 21-23, 2007*. A. Ghose, G. Governatori, R. Sadananda, Eds. LNAI 5044. Springer: Berlin, 255-267, 2009.
10. Luck, M. McBurney, P. Shehory, O. Willmott, S. *Agent technology: a roadmap for agent based computing*. AgentLink, Southampton, UK, 2005.
11. Molesini, A., Denti, E., Nardini, E., Omicini, A. Situated process engineering for integrating processes from methodologies to infrastructures. In *Proc. of the 2009 ACM Symposium on Applied Computing*, ACM: New York, 699-706, 2009.
12. OMG (2008) “Software & Systems Process Engineering Meta-Model Specification”, v2.0. Formal/2008-04-01, <http://www.omg.org/docs/formal/08-04-01.pdf>.
13. Seidita, V., Cossentino, M., Gaglio, S.: A repository of fragments for agent systems design. In *Proc. of the 7th Workshop from Objects to Agents (WOA 2006)*, Catania, Italy, pp. 130–137, 2006.

Applying Process Document Standarization to INGENIAS

Alma Gómez-Rodríguez¹ and Juan C. González-Moreno¹

Departamento de Informática (University of Vigo)
Ed. Politécnico, Campus As Lagoas,
Ourense E-32004 (SPAIN),
{alma,jcmoreno}@uvigo.es
<http://gwai.ei.uvigo.es/>

Abstract. The increasing interest on Agent Oriented Software Engineering in the last years is mainly due to its suitability for the design and implementation of huge, complex, distributed systems. In this field, special attention has been paid to development processes, because of direct correlation between the quality of the product and the process followed to obtain it. At the moment, there is neither a formal specification to define the activities to develop, the participants and the deliverables, nor guidance about the elements to introduce in the model or how these relate with each other. The use of standard notations may make it easier to describe the process, the resources and the mandatory deliverables. The IEEE FIPA Design Process Documentation and Fragmentation working group has proposed a template in order to cover this gap. This paper provides a first attempt at considering the results obtained by applying the proposed template to a well known development process proposed by the INGENIAS methodology for the construction of a multi-agent system.

1 Introduction

The software quality assurance discipline considers that the development process is very important because of the direct relation between process quality and final product quality. In particular, in Agent Oriented Software Engineering (AOSE) many methodologies and their associated processes of development have been proposed in the latest years [1–4]. All of them introduce all the conceptual abstractions that must be taken into account in any MultiAgent System (MAS) development.

Nevertheless, it is necessary to pay attention to the introduction of standards for the formal definition of these processes and methodologies. The use of standards provides better understanding and simplifies the task of sharing information among several groups of developers. At this moment, there is an ongoing work impelled by FIPA that proposes a template for standardizing methodological definition in AOSE field. The template provides a way of describing processes as well as some guidelines of how to use it. The detailed definition of the template is available at [5].

Following the key lines included in the template, this paper addresses the definition of a well known development process in the AOSE field. The process defined using the template is the one originally proposed by INGENIAS methodology [6–8] which is based on the Rational Unified Process (RUP)[9] also known as the Unified Development Process(UDP).

The remainder of the paper is organized according to the different sections of the template proposed by FIPA. This means that next section starts with an introduction to the methodology, with indication of its most relevant features. Section 3 details one of the phases defined by INGENIAS for MAS development and after, we introduce the definition of work-products dependencies. And finally, the paper ends with the conclusions obtained from template usage.

	Analysis	Design
Phases		
Inception	To generate use cases and identify actions of these use cases with the corresponding Interaction Model To outline the system architecture with an Organisation Model To generate Environment Models which reflects Requirement elicitation	To generate a prototype using RAD tools such as ZEUS or AgentTool
Elaboration	To refine use cases To generate Agent Models that detail the elements of the system architecture To continue with the Organisation Models, identifying workflows and tasks To obtain Task and Goal Models to highlight control constraints (main goals, goal decomposition) To refine the Environment Model including new elements	To focus the Organisation Model on workflow To refine Tasks and Goal Models reflecting the dependencies and needs identified in workflows and the relationships with system's goals To show how tasks are executed using Interaction Models To generate Agent Models which show required mental state patterns
Construction	To study the remaining use cases	To generate new Agent models or refining existing ones To study social relationships in order to refine the organisation

Table 1. Phases and tasks for Ingenias Development Process

2 INGENIAS Process Documentation: Introduction

The INGENIAS methodology covers the analysis and design of MAS and it is intended for general use; that is, with no restrictions on application domain. It



Fig. 1. Lifecycle for INGENIAS Methodology

has shown its capability and maturity as the supporting specification for the development of Multi-Agent Systems (MAS). The methodology provides the INGENIAS Development Kit (IDK), which contains a graphical editor for MAS specifications. Besides, the INGENIAS Agent Framework (IAF) [10], which is integrated in the IDK, enables a full model-driven development and transforms automatically specifications into code in the Java Agent DEvelopment (JADE) Framework.

Following the Rational Unified Process (RUP)[9], INGENIAS methodology distributes the tasks of analysis and design in three consecutive phases (see Fig. 1): *Inception*, *Elaboration* and *Construction*, with several iterations (where iteration means a complete cycle of development, which includes the performance of some analysis, design, implementation and proofs tasks). The sequence of iterations leads to the procurement of the final system.

The process of such development process is often represented by its authors in a tabular form (see Table 1). In the table, the three development phases are presented jointly with two different types of workflows for *Analysis* and *Design*. The methodology pays few attention, compared to RUP, to *Implementation* and *Test workflows*, because it provides some tools which automatically generate code, in parallel with system's specification. Attending this facility, these workflows are considered not to be modeled as a fundamental part of the process.

INGENIAS tries to follow a Model Driven Development (MDD) [11] approach, so it is based on the definition of a set of meta-models that describe the elements that could be used to specify a MAS following five viewpoints [12]:

1. **Agent:** It specifies the definition, control and management of each agent mental state
2. **Interaction:** The model is used to describe the agents' interactions
3. **Organization:** It details MAS architecture
4. **Environment:** This viewpoint is used to model the environment of the MAS
5. **Task and goals:** This meta-model present a detailed view of the tasks and goals assigned to each agent

The development process is supported by a set of tools, which are generated from the meta-models specification by means of a meta-modeling processor (which is the core of the IDK). MAS modeling is facilitated by a graphical editor and a verification tool. The methodology has been used in several examples from different domains, such as PC management, stock market, word-processor assistant, and specially collaborative filtering information systems.

Detailed references of the methodology from their authors can be found in [6, 7, 12].

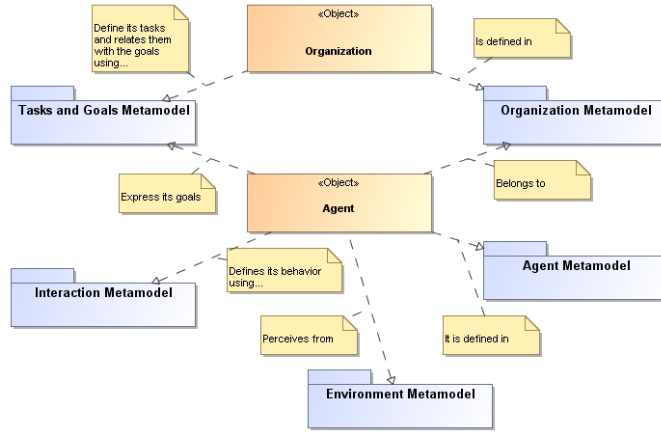


Fig. 2. Global meta-model of INGENIAS Methodology

2.1 The INGENIAS Process lifecycle

As pointed in the introduction INGENIAS methodology distributes the tasks of analysis and design in three consecutive phases: *Inception*, *Elaboration* and *Construction*. Each phase may have several iterations (where iteration means a complete cycle of development)¹.

Following the idea proposed by RUP to take the system architecture as guideline for development, INGENIAS propose the use of the Organization Model as basis for the MAS definition and construction (see Table 1)

2.2 The INGENIAS Meta-model

From the point of view of INGENIAS, a meta-model defines the primitives and the syntactic and semantic properties of a model. Following this idea the methodology provides five meta-models that define five different views of the system.

An important characteristic of INGENIAS meta-models is that they are quite detailed (fine grained). This is due to that they are intended to be a precise definition of the system, and also because each meta-model is the basis for the automatic code generation provided by the *INGENIAS Development Kit* (IDK).

As an example of how meta-models are detailed in INGENIAS, Fig. 3 shows the graphical definition of the Agent Meta-model. An agent is identified as an autonomous entity, with particular goals and a unique identity. Three fundamental elements are identified for each agent: *the roles* the agent must

¹ the definition of INGENIAS can be found in <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs/INGENIAS.pdf>

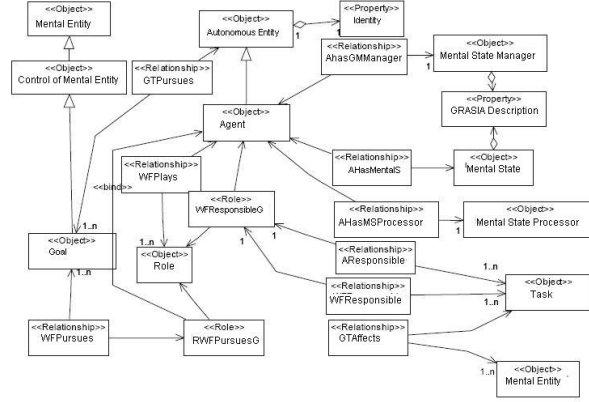


Fig. 3. Agent meta-model proposed by INGENIAS Methodology

play, *the tasks* the agent must accomplish and its *mental state*. The relationships among them show how an agent can pursue its goals and how it achieves that goals executing a particular tasks. This meta-model is selected because the entities (*Agents* and *Roles*) are also included in other MAS meta-models.

2.3 Definition of MAS meta-model elements

In table 2, the basic elements taken from the meta-model are introduced. As the meta-models of INGENIAS are very detailed, only the most important concepts have been defined. For further details, the original documentation of the methodology must be reviewed [6, 7, 12]

3 INGENIAS Process Documentation: Inception Phase

Following the recommendation of the template each phase must be specified in a section, due to space limitations this paper is focused on the first phase proposed by the methodology. According to INGENIAS, meta-models are a key issue in the MAS development, but these models must be associated to the activities done to obtain them. This integration is the key point covered on the specification introduced in this section.

Figure 4 gives a general view of the INGENIAS Inception Phase. The methodology considers that the development process is initiated from the document describing the problem, so that, this can be considered the initial input of the process. From this document, the Inception phase introduces several activities that are described in Fig. 4. Regarding the Analysis workflow at this level the activities are:

Concept	Definition	Cross-References
Agent	An agent entity is an autonomous entity with identity, purposes and that performs activities to achieve its goals	Autonomous Entity
Application	An application is a wrapper to computational system entities. Computational represents a system having an interface and a concrete behavior	
Autonomous Entity	Root concept that represents an entity with identity and that pursues goals	Goal
Goal	According to the BDI model, a goal is a desired state that an agent wants to reach. In planning, a goal is represented by a world state. Here a goal is an entity by itself, however it can be related with a representation of the world state using satisfaction relationships with tasks. This relationships contains references to descriptions of mental states of agents, so they refer to the image of the world that agent have	Agent
Interaction	Interaction represents an exchange between two or more agents or roles. There can be only one initiator and at least one collaborator. An interaction also details the goal that pursues. This goal should be related with the goals of the participants.	Agent, Role & Goal
MentalState	A mental state represents the information an agent has in a certain moment. A MentalState is an aggregate of mental entities.	Agent
Organisation	An organisation is a set of agents, roles and resources that get together to achieve one or several goals. Inside an organisation there are not other organisations, just groups. You can think of an organisation as an enterprise. Internally it is composed by departments that may be restructured without affecting the external image of an enterprise.	Agent
Resource	Resource describes a resource according to TAEMS notation. Opposite to TAEMS, there is no distinction between consumable and non-consumable resources.	
Role	A role is a self-contained grouping of functionalities. When an agent plays a role we want to express that you have to execute tasks associated to a role and participate in the same interactions that role	Agent
Task	Tasks is the encapsulation of actions or non-distributable algorithms. Tasks can use Applications and resources. Tasks generate changes in the mental state of the agent that executes them. Changes consist of: (a) modifying, creating or destroying mental entities; or (b) changes in the perception of the world by acting over applications (applications act over the world producing events, that are perceived by the agent). Though tasks can be also assigned to roles, at the end, it will belong to an agent	Role
Workflow	A workflow is an abstraction to a process that has been automatized using activities and identifying responsibility relationships	

Table 2. Definition of MAS meta-model Elements

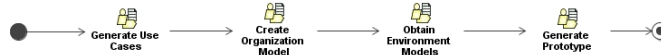


Fig. 4. Activities and workflows of Inception phase proposed by INGENIAS Methodology

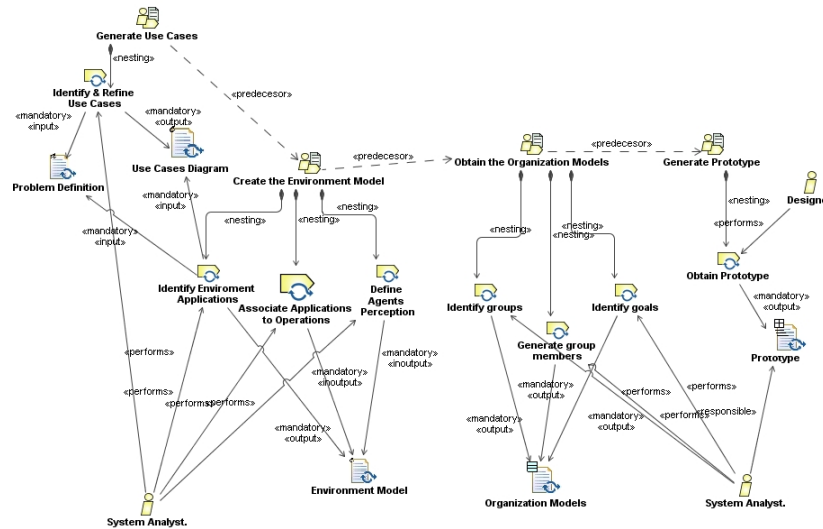


Fig. 5. Detailed tasks of Inception activities

- Generate Use Cases
- Initiate the architecture using the Organization Model
- Generate the Environment Model

In what respects to Design only the construction of a rapid prototype must be addressed.

All these activities and the tasks associated to each of them are shown in Fig. 5. From this figure, we can identify the different tasks proposed by INGENIAS for Inception and the produced work-products. Moreover, the roles responsible of each task as well as the kind of responsibility they assume are also shown.

3.1 Process roles

The template says that the roles that are responsible of each task must be identified. Nevertheless, INGENIAS methodology makes no explicit reference to the roles implied in the development. We consider roles identification proposed by the template very helpful because it solves a problem previously detected when

using the methodology in real environments. In some cases, the team members have difficulties to know what activity or task they must do and what their responsibilities are according to the process.

To state the roles involved in this phase, it has been taken into consideration that INGENIAS follows RUP development and it has been considered also the activities to be done and the level of abstraction of such activities. From this analysis, we propose only two roles to participate in this phase: the *System Analyst* and the *Designer*.

The System Analyst is responsible or performs the most part of the activities proposed in this phase. In particular, he will:

- *Identify the Use Cases and construct and refine the Use Cases Diagram.* From the initial description of the problem to solve, the analyst must obtain the use cases that will guide after the creation of the Interaction Model.
- Define the Environment Model, showing the interaction of the system with its environment. This will imply to: *identify applications* (in INGENIAS, all the software and hardware that interact with the system and can't be designed as an agent will be considered an application); *associate operations to particular applications* and *define agents perception on applications*.
- *Obtain the Architectural view of the System using the Organization Model.* This means to generate a structural definition of the system by *identify groups* in the organization, *generate group members* and *identify goals*.

The second role identified: the *Designer* must be responsible of *generating the prototypes*. According to INGENIAS literature, this will be done using a rapid application development tool such as ZEUS, Agent Tool or others.

3.2 Activity Details

Following the template recommendation [5], this section details the activities previously outlined for Inception Phase.

Generate Use Cases.- The generation and refining of Use Cases has been identified as a unique task. The goal of this task will be to identify the intended functioning of the system. Knowing the functionalities the system must provide, will allow to identify interaction collaborators and initiators and also to discover the nature of such interactions that will affect the type of control applied to the agent: *planning, cooperation, contract-net or competition*.

Generate the Environment Model.- The Environment Model tries to show the elements that constitute the environment of the system, and in consequence, the perceptions of the agents. The elements defined in this model are of three basic kinds: *agents, resources and applications*.

Figure 6 shows the task to be accomplished for obtaining an Environment Model of the system to construct. These tasks are further explained in Table 3.

Initiate the architecture.- One of the key activities in Inception Phase is to start the definition of system architecture. This is done by constructing the Organization Model, which reflects mainly the system's workflows.

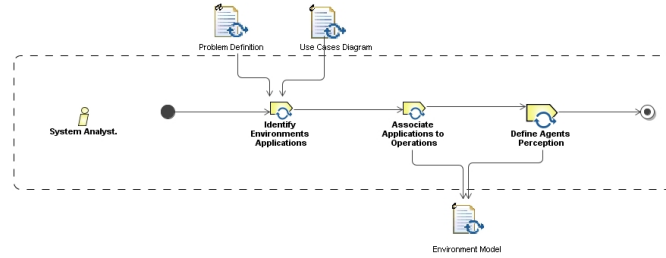


Fig. 6. Obtaining the Environment Model in the Inception Phase of INGENIAS

Activity	Task	Description	Roles Involved
Generate the Environment Model	Identify the Environment Applications	All the software and hardware that interact with the system and that can not be designed following an agent oriented approach will be considered an application	System Analyst
Generate the Environment Model	Associate the Applications and Operations	Operations are associated to the applications defined by requirements. These operations have a signature, a precondition and a postcondition. The identification of operations is a conventional engineering task.	System Analyst
Generate the Environment Model	Define Agents Perception	The main aim of this task is to define agents perception on environment applications, at this moment of process it is enough to relate agents and applications	System Analyst

Table 3. Task of Activity Generate the Environment Model of Inception Phase of INGENIAS

In Figure 7 the basic tasks related with the procurement of Organization Model in Inception activity are shown. These activities try to obtain an organizational view of the system, attending its structural, functional and social aspects. The detailed definition of tasks are addressed in Table 4.

Construction of a prototype.- The generation of a prototype is a unique and simple task, that is supposed to be done using a RAD tool.

3.3 Work Products

The last aspect to be covered to complete the specification of a phase following the template recommendation [5] is to detail the Work products used. The INGENIAS Inception Phase produces as result four basic work-products: a *Use Cases diagram*, an *Environment Model*, one or more *Organization Models*

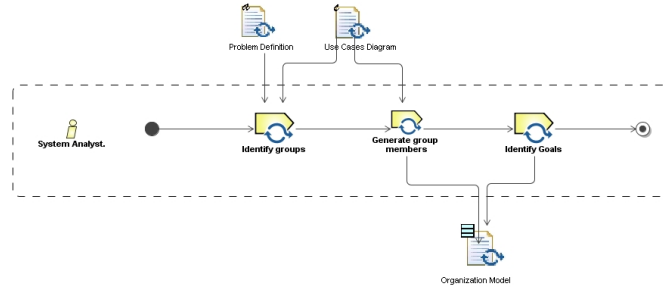


Fig. 7. Obtaining the Organization model in the Inception Phase of INGENIAS

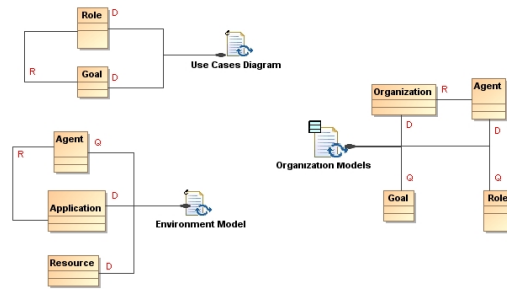


Fig. 8. Structure of Inception Work-products

and a *prototype of the system* to be built. The relationships among the models and the the meta-model elements are shown in Figure 8. Organization model, for instance, defines the organization meta-model element and the agents and uses the roles and goals previously defined. In this particular case, organization concept includes also the groups within the system (see organization definition in table 2). On the other hand, the Environment Model defines the internal and external applications the system interacts with, as well as the resources available.

4 INGENIAS Process Documentation: Work-product dependencies

Following the template [5], a final aspect that must be detailed after the specification of the process phases is the Work-product dependencies. Figure 9 introduces a global view of INGENIAS work-products, as well as their dependencies. As shown in the Figure, *Agent Model* depends on *Organization* and *Environment Models*, while the *Interaction Model* shows dependencies from *Agent* and *Task/Goal Models* among others.

Activity	Task	Description	Roles Involved
Obtain the Organization Model	Identify groups	The groups in the system must be identified. In this way the participants in a particular work flow will be organized.	System Analyst
Obtain the Organization Model	Generate group members	Members (agents, roles, resources and applications) are assigned to groups creating the corresponding relationships. If needed, the groups can be decomposed in order to reduce complexity.	System Analyst
Obtain the Organization Model	Identify groups	The organization has a set of goals that must justify collaboration between agents. The goals identified in this task will after be assigned to individual agents or roles in the Task and Goals Model.	System Analyst

Table 4. Task of Activity Initiate Architecture of Inception Phase of INGENIAS

5 Conclusions and Further Work

Most times a methodology proposes a particular development process in its description. This process may be common to different methodologies, but it is not specified using a common notation. This gap is being covered by the IEEE FIPA Design Process Documentation and Fragmentation working group with the proposal of a template for its definition. This paper provides a first attempt at the application of the proposed template to model the original development process of INGENIAS methodology.

The standard has been very useful for the definition and the results have been satisfactory. Thanks to the use of the the template jointly with the standard notation, we have been able to improve some how the definition of INGENIAS RUP based process. For instance, we have identified roles and responsibilities for each of the tasks that were not previously defined. Moreover, the dependencies among work-products indicate some relationships that must be taken into account when adapting the methodology to Agile development processes.

Acknowledgements This work has been supported by the project *Novos entornos colaborativos para o ensino* supported by Xunta de Galicia with grant 08SIN009305PR.

References

1. Cuesta, P., Gómez, A., González, J., Rodríguez, F.J.: The MESMA methodology for agent-oriented software engineering. In: Proceedings of First International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS'2002). (2002) 87–98
2. Bernon, C., Cossentino, M., Pavón, J.: Agent-oriented software engineering. *Knowl. Eng. Rev.* **20** (2005) 99–116

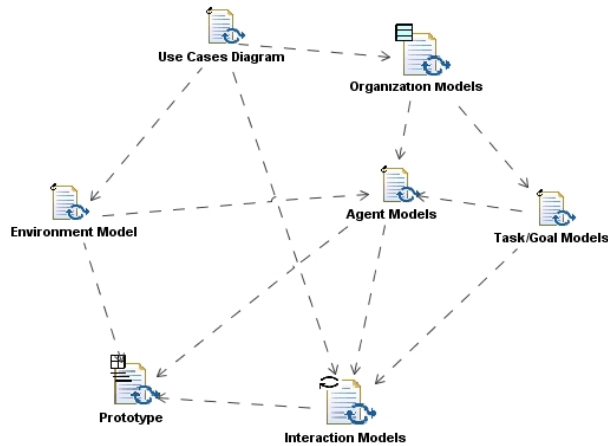


Fig. 9. Dependencies among INGENIAS Work-products

3. Federico Bergenti, Marie-pierre Gleizes, F.: Methodologies And Software Engineering For Agent Systems: The Agent-oriented Software Engineering Handbook. Springer (2004)
4. Henderson-Sellers, B., Giorgini, P.: Agent-oriented methodologies / Brian Henderson-Sellers, Paolo Giorgini. Idea Group Pub., Hershey, PA (2005)
5. IEEE FIPA Design Process Documentation and Fragmentation: IEEE FIPA Design Process Documentation and Fragmentation Homepage. <http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/> (2009)
6. Gómez-Sanz, J.: Modelado de Sistemas Multi-agente. PhD thesis, Universidad Complutense de Madrid. Facultad de Informática (2002)
7. Gómez-Sanz, J.J., Pavón, J.: Meta-modelling in agent oriented software engineering. In: Advances in Artificial Intelligence - IBERAMIA 2002, 8th Ibero-American Conference on AI, Seville, Spain, November 12-15, 2002, Proceedings. Volume 2527 of Lecture Notes in Computer Science. (2002) 606–615
8. Pavón, J., Gómez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. Multi-Agent Systems and Applications III **2691** (2003) 394–403
9. Rational Software: Rational Unified Process: White Paper (1998)
10. Gómez-Sanz, J.: Ingenias Agent Framework. Development Guide V. 1.0. Technical report, Universidad Complutense de Madrid. (2008)
11. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. Software, IEEE **20** (2003) 36–41
12. Grupo de Investigación en Agentes Software: Ingeniería y Aplicaciones. INGENIAS Section. <http://grasia.fdi.ucm.es/main/?q=es/node/61> (2010)

Exploring the Boundaries: when Method Fragmentation is not Convenient

Chiara Leonardi¹, Luca Sabatucci¹, Angelo Susi¹, and Massimo Zancanaro¹

Fondazione Bruno Kessler IRST, Via Sommarive, 18 I-38050 Trento, Italy
{cleonardi,sabatucci,susi,zancana}@fbk.eu

Abstract. This paper presents an approach to explore the coupling of User-Centred Design and Tropos methodologies. The two methodologies have been employed in a real project aiming at developing smart environment for nursing home to support medical and assistance staff. In particular Tropos has been used for modeling (and reason about) the domain and the system, whereas User-Centred Design has been useful for establishing an interface for communicating with stakeholders. The integration was challenging due to the epistemological differences between the two design approaches.

1 Introduction

Goal-oriented Requirements Engineering (GORE) plays a fundamental role in the development of *user intensive systems*, enabling reasoning about the domain features with the aim of identifying conflicts and of checking for validity of functional and non-functional requirements. Nevertheless, we experienced the need for an effective way to center the design on the users of the system. The strength of Goal-Oriented techniques in modelling the domains can be still enhanced by coupling the engineering perspective with a creative perspective typical of User-Centred Design (UCD) approaches. Basic principles of this integration are: (i) early focus on users, tasks and environment, (ii) the active involvement of users in the design process, (iii) allocation of functions between user and system, (iv) the incorporation of user-derived feedback into system design, (v) iterative design whereby a prototype is designed, tested and modified.

Both approaches ground their processes in information about the people that are directly or indirectly involved by the technology that has to be developed. Yet, they not only have different set of techniques and incompatible vocabularies but also they are based on two diverging epistemological foundations. UCD practitioners shun from any formal method at risk of compromising the actual use of the knowledge gained in the field. On the other side, RE practitioners often loose contacts with “real” people because formalizations cannot easily be shared with them: user analysis thus becomes a single-player game rather than a meaningful dialogue with stakeholders.

Context. The need of reconciliating the two approaches raised up from the work in a large research project aimed at developing a smart environment in

nursing home as support to medical and assistance staff¹. Coming from some experiences in situational method engineering [1, 2] we supposed to use method fragment composition and meta-model unification for creating an ad-hoc design process for our aims. Anyway the attempt to define a common vocabulary for engineers and sociologists team members was the source for long philosophical discussions that terminated with the feeling that other ways should be walked. In particular the main problem was to find an agreement on identifying precise relations between terms coming from different vocabularies. In addition, the situation became more complicated when we tried to understand phases and activities to perform (and work product to produce); whereas Tropos life-cycle is clear and well-defined, UCD practitioners refuse to decide a-priori the activities to perform in a project and their order. Anyway, the attempt was not useless: we reached the awareness that the problem is not only dialectical, but epistemological; the two research teams intend their process and language from different points of view. It was clear that a combination of the two approaches was necessary but that combination should avoid reducing either one approach and the other. We perceived the paradigm should change from fragment composition to fragment collaboration.

Contribution. The contribution of this paper is the analysis of foundations for the integration of the engineering perspective of Tropos with the qualitative prospective of the UCD approach. While goal oriented requirement engineering provides accountable procedures and formal or semi-formal methods for eliciting requirements and providing systematic and complete system description, the UCD process encompasses less formal practices aimed at envisioning sparking ideas by inspirational techniques ranging from ethnographic fieldwork for understanding users, to design storm to inspire 'blue sky' concepts. We propose a framework for mediating these two different approaches without compromising their very nature: in the differences, it lies the power of the integration and its risks. The framework is based on a novel concept for creating a synergy between methodologies (or parts) that is the definition of communication protocols between methods. This concept is based on the exploration of methodological and linguistic boundaries and the definition of channels for sharing and tracking design data among practitioners of the involved methodologies.

2 Challenges

Traditionally, two main trends can be identified for composing research approaches of different nature: from one hand, there is the tendency to privilege a disciplinary perspective and, on the other hand, the effective effort to integrate different epistemologies [3]. In the first case one disciplinary approach is usually modified to be *assimilated* into the other approach: while the risk is to limit the potential of the approach itself; the advantage is to work in a situation of 'methodological purity'. In the case of disciplinary *integration*, practitioners should accept to work in a situation of methodological pluralism: the goal is not

¹ ACube project, funded by the Autonomous Province of Trento. <http://acube.fbk.eu/>

to transform or to assimilate a specific approach to make it fit into another one, but rather to bridge the gap between different research traditions and take advantages of their mutual strengths.

This distinction can be borrowed from social science to be applied in software engineering. Indeed, the Situational Method Engineering [4, 5] is grounded on the assimilation approach: constructing ad-hoc software engineering processes by reusing fragments of existing design processes; the basis for the assimilation technique is the method fragment [6], a self contained component that can be used as building block for the process composition. Techniques for fragment manipulation (extraction, selection, and composition) are still open points, and even if there is a disagreement about the level of precision, it is clear that fragment specification requires a language for describing at least the process and its products. Some recent approaches [7] make use of the SPEM notation for describing the process as a workflow and meta-models as linguistic keys for bridging activities and artifacts coming from different methodologies [8].

The preconditions for applying situational method engineering is to analyze core elements of a methodology for building model descriptions of activities and artifacts. There are cases in which these preconditions do not apply, and a formalization of the process is not feasible without the risk to lose all advantages of the process. In our case, whereas Tropos phases and diagrams can be formalized by using meta-models and SPEM diagrams [8, 7], UCD practitioners are very resistant in providing any kinds of structure for framing their theories and techniques; they shun from any formal method at risk of compromising the actual use of the knowledge gained in the field. They claim the freedom is the key for flexibility of procedures and for quickly adapting to the context. In addition, the language they use exploits ambiguity as a design opportunity and not as a problem: the everyday world is inherently ambiguous, and allowing this ambiguity to be reflected in design has the advantage to encourage people to interpret situations, by establishing deeper and personal relations with the meaning offered by situations [9].

For these reasons UCD artifacts have typically a descriptive form that preserves every information about the domain, ranging from users' motivations to the empathy versus the product. Therefore, it is not reasonable (and productive) to generate a meta-model without reducing the expressiveness of these artifacts. In these cases the integration approach may be useful, as a replacement of the assimilation approach; this because it does not require transforming the methods specific to each research tradition but it is based on creating preconditions for a beneficial dialogue between the two [10]. Maintaining the different epistemological and methodological traditions is grounded on managing the dialectic issues concerning the concurrent usage of different research paradigms.

By exploring the boundaries between Tropos and UCD, two sub-challenges have emerged and are discussed in the following.

2.1 Epistemological challenge

The first issue is to consider epistemological foundations and validity criteria of both the approaches, to manage differences without weakening and distorting the two research paradigms. While Tropos is grounded on a *positivist* research tradition [11], several methods employed in UCD derive from a *constructivist* perspective.

Positivism is an epistemological perspective which holds that knowledge is based on sensorial experience and positive verification. One of the key features of positivism is the ability of demonstrating the logical structure and coherence of a concern by axiomatization. Tropos is classified as a positivist approach — even if the debate on the positivist nature of many RE methods has recently been criticized [12] — by providing a precise frame for the modeling activity and the reasoning process.

Constructivism is a different epistemological perspective which holds that the Ontological Reality is utterly incoherent as a concept, since there is no way to verify how one has finally reached a definitive notion of Reality: scientific knowledge is built by scientists and not discovered from the world. In this context, there is no single valid methodology and researchers play an active role in defining the reality. UCD is grounded on this research tradition: hence the scarce formality of methods, the subjective insights developed by practitioners, and the ambiguities in the analysis are, if correctly managed, not only accepted but actively perused [9, 13].

Each methodology has its own basic axioms that not only guide the research process, but also the way method is perceived and applied. In the spirit of integration rather than assimilation, the concrete procedure proposed in this paper does not dictate a choice or a priority among the two approaches but rather leaves analysts free to choose the most promising techniques in the two domains. Indeed, the process boundary to explore and to overcome is peculiar: making explicit and reason on these differences is the first step to exploit the complementary nature of UCD practices and RE approaches.

2.2 Linguistic boundary

Beside the methodological boundary, a linguistic boundary exists. The concurrent usage of both approaches requires that a common language exists in order to make a dialogue possible. Several concepts exist in both Tropos and UCD that suggest an integration is possible and profitable. Examples of these are the pairs of goal/need, actor/persona, task/activity; yet these terms have slight different meanings in the two methodologies that hinder the composition process.

The integration of the two methodologies must pass through a reconciliation of terms. Two alternatives were possible: (i) to create a unified meta-model of the integrated process, or (ii) to tie up terms with similar meanings while keeping them separate. The first way was fascinating but it failed because of the difficulty to identify a meta-model for the UCD process. Just as an example, during the attempt to formalize a term like 'persona', we had the feeling to loose

the flexibility and the expressivity of the instrument. Thereafter, the definition of a communication protocol for allowing an exchange of data between the two processes revealed preferable even if it required an additional effort for creating a framework in which data of different nature can be easily interchanged.

The proposed framework maintains the original nature of the instruments and it introduces new methodological and conceptual tools for tracing all data transformations along the process. In the following, some differences are identified between Tropos and UCD techniques. For example, in our case the field study leads to the identification of a number of institutional roles for our stakeholders. Then, Tropos engineers used these data for defining relevant actors of the domain, their main goals and dependencies. Afterwards, the analysis tried to generalize information in order to discover high level interdependencies among these actors. On the other side, UCD practitioners focused on the behaviour of individual workers, during their daily job, keeping track of attitudes and personal motivations that may influence the final value of the design. From the description of stakeholder daily activities (including routines, methodologies, but also unexpected situations to front) the Tropos engineers extracted goal/task decomposition and resource usage. The UCD designer, again, focused more on problems, stressing situations, lacks of methodologies and expectations over the system under design. These activities were continuously intermingled until the awareness of the knowledge of the domain was considered deep enough.

3 Methodology Integration

The *Tropos methodology* [14] is a goal-oriented design process that relies on a set of concepts, such as actors, goals, plans, resources, and dependencies to formally represent the knowledge about a domain and the system requirements. An actor represents an entity that has strategic goals and intentionality within the system or the organizational setting. Goals represent states of affairs an actor wants to achieve. A Plan is a means to realize a goal. Actors may depend on other actors to attain some goals or resources or for having plans executed.

Tropos distinguishes five phases in the software development process: Early Requirements, where the organizational domain is described, Late Requirements, where the system-to-be is introduced in the organization, System Architecture Design, System Design and System Implementation.

User Centered Design is a design philosophy that exploits a number of different techniques within a iterative design process. Tenets of UCD are: early focus on users, tasks and environment, the active involvement of users in the design process, allocation of functions between user and system, the incorporation of user-derived feedbacks into system design, iterative design by which a prototypes is designed, tested and modified. UCD exploits a series of well-defined methods and techniques coming from social sciences and psychology for analysis, design, and evaluation technologies. Contextual inquiries, personas and scenarios - that we adopted in our project - are widely used when researchers aim at obtaining a rich picture of a context (organizational, social, physical), at easily communicat-

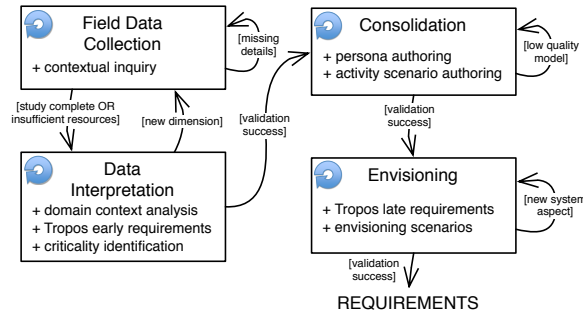


Fig. 1: Phases of the process and life-cycle

ing it to stakeholders in order to envision acceptable and innovative technological solutions.

The framework for the integration of the two methodologies considers both epistemological and language boundaries discussed in the previous section. The result is an integrated methodology where the component processes maintain their own identity, even if their activities are interleaved and an intensive exchange of data is supported by specific communication protocols.

3.1 Integration of lifecycles

The process model is represented in Figure 1, that shows phases, activities and conditions for moving along the steps. Every macro-phase of the process is represented as a box with a title and activities are placed inside. The execution order of activities in a phase is not specified: they are concurrently managed and iterated as well as some conditions are met. Generally, each phase terminates with a validation according to assigned criteria, in which typically also stakeholders are involved.

The process begins with the investigation of the domain in order to understand the organizational setting and to derive possible needs and services that the system could provide to users. Several methods exist to analyse the domain: recently ethnographic methods, such as contextual inquiry, demonstrated their capacity to satisfy the needs for a deep but at the same time rapid understanding of complex domain. Data interpretation provides a first classification and abstraction in order to create a believable model of the domain, but avoiding to lose important details typical of a narrative analysis. In our process, data interpretation is concurrently carried out in a twofold way: one is the domain context analysis and one is the Tropos early identification. The data consolidation acts as a filter in order to focus on relevant characteristics to consider in the following phases. Finally, the envisioning phase lets the analysis team to reason on the system-to-be in order to expand designers' perspective, to look at the problems from different points of view, to figure out how their ideas can work in

a real context, to identify design criticalities, and to generate requirements. The process ends with the validation of requirements with stakeholders, essential for moving to the next design phase.

3.2 Exploring Methodological Boundaries

The modeling activity in the Tropos methodology follows a 'positivist' approach, indeed, Tropos algorithms and meta-model [8] provide well-defined descriptions for the design activity. A typical shortcoming of positivist approaches is that they can not easily provide general techniques for interpreting the domain, and transforming perception data into model elements. In the case of Tropos, for instance, it is the analyst's responsibility to decide how to model the domain, managing trade-offs and choices: the process does not provide general guidelines about what actors and goals to include in the model, how to handle and/or decomposition, and so on.

On the other side UCD is a 'constructivist' approach, thus it avoids prescribing a process to follow, but it provides criteria for achieving project objectives and supporting the designer decisions. For instance, the contextual inquiry phase, in which the designer gathers detailed data needed for the design and discovers implicit aspects of work that would normally be invisible. This activity may be conducted by using different techniques (interviews, direct observation, questionnaires, and so on) to use in isolation or to interleave according to the needs emerging from the context.

Data captured by contextual inquiry is greatly useful for leading decisions during the early requirement; between these two activities there is a methodological boundary that may be used to create a method synergy. The contextual inquiry produces a huge documentation concerning observed users/customers and their needs. If opportunely analyzed and filtered this data can feed the Tropos entity identification, by providing criteria for motivating the introduction of new elements and tracing the source.

Nevertheless, another methodological boundary exists in the opposite direction: filtered data, modeled with Tropos, is an input for UCD designers in order to feed the following consolidation phase. An example is the Tropos early requirement that produces a model of the domain, organization dependencies and stakeholders' strategies for goal commitments. This model can be profitably used by UCD designers in order to summarize relevant aspects of the domain preliminary to the envisioning of the new system.

3.3 Exploring Linguistic Boundaries

A linguistic boundary is due to a mismatching in the dictionary used in the two methodologies. This aspect is specifically evident in the integration between a well-specified language (Tropos) and a language that is intentionally verbose and sometimes ambiguous (UCD). The identification of these linguistic boundaries is important for the reconciliation of incompatible concepts and for creating the framework for data sharing.

An example of linguistic boundary exists between the Tropos 'task' and the UCD 'activity' terms. A Tropos 'task' is defined as the conceptualization of a plan that provides the means for the operationalization of a goal. An example of task is [caregivers monitor guests' behavior]. The UCD 'activity' concept captures additional information about the context in which it is carried out, including the user point of view and the empathy aspect. An instance of activity description is extracted from an interview to a caregiver:

"...during my job it is important to continuously observe patients' behavior, but this is often an heavy activity to carry on together with other our duties. This is due to the high number of guests compared to the low number of professionals. This working overhead causes we are incapable of concentrating on the human aspect of our job as well as we would do ..."

Maintaining and tracking this difference along the unified process is fundamental for the following design phases, but it requires a reconciliation: it requires to explore how a Tropos task is related to an UCD activity. The solution we explored is to connect the two concepts with a different kind of relationships respect to classical ones used in meta-modeling. We introduce a loose relationship among linguistic elements that is based on collaboration protocols. This is discussed in the following section.

3.4 Defining Collaboration Protocols

In a situation of methodological pluralism, in which at least two design teams collaborate, a collaboration protocol relates linguistic elements that need to be reconcile. A protocol defines crossing terms, steps, guidelines and instruments for translating and tracking design data from one methodology to the other.

The exploration of methodological boundaries provides hooks in which a collaboration is possible and beneficial, whereas linguistic boundaries identify elements that must be reconcile in order to realize the collaboration. An instance of collaboration protocol is defined for the boundary existing when moving from the Tropos early identification to the UCD consolidation activity, which — in our project — was conducted with activity scenarios and personas authoring.

The Tropos *Early Requirement* activity depicts the strategic and organizational views of the domain. During this phase the relevant stakeholders are identified, along with their respective objectives; stakeholders are represented as actors, while their objectives are represented as goals. Goal and plan models allow the designer to analyze goals and plans from the perspective of a specific actor. This phase results from the analysis of social and system actors, as well as of their goals and dependencies for goal achievement.

The use of *Scenarios* in RE is pretty established as an instrument to describe instances of behavior of the system, but their use ranges for several purposes and it is aimed at very different concerns [15]. We used activity scenarios, which are stories about people carrying out activities; they describe a context in which personas act with the aim of summarizing, clarifying and reasoning on the

collected information; these scenarios are narrative description of the behavior of personas in critical contexts of the domain [16]. Another difference respect to classical software engineering scenarios is the use of personas. *Personas* are powerful instruments for creating descriptive models of system-to-be users [17]. Mikkelsen and Lee [18] introduced user archetypes that describe classes or types of user of a product, further refined by Cooper [17] that introduces “personas” as composite archetypes based on behavioral data gathered from many actual users encountered in ethnographic study. They provide a tangible representation of the user to act as a believable agent in the setting of scenario. Summarizing, personas are hypothetical but significant user archetypes for which to motivate the design; they are defined as [19–22]: (i) attitudes, experiences, aspirations; (ii) general expectations the persona may have about the experience of using the product; (iii) behaviors that persona will expect from the product; (iv) how the persona think about basic elements or units of data.

A linguistic boundary was identified between the Tropos concept of ‘actor’ and the UCD concept of ‘persona’. Whereas both of them identify users of the system-to-be, an actor is a powerful instrument to abstract a role in the organization, while a persona is an archetype of user, sufficiently concrete to provide the understanding of the empathy emerged from ethnographic study and personal motivations within a scenario. The cognitive and emotional dimensions are important factors persona try to catch for helping the designer to take decisions in the design process, characteristics that are missing in an actor.

The collaboration protocol for this couple — methodological/linguistic — of boundaries is based on the identification of criticalities that tie up the organization model with the concrete context in which actors play their roles. The *Criticality Identification* bridges the Tropos early requirement analysis with the following persona and scenario authoring, by connecting linguistic terms like actor and persona.

A criticality is an exceptional situation to front in the organization for which the system is designed. The criticality is discovered in the Tropos goal-models, by analyzing and/or decomposition and by the conflict analysis. A criticality is identified as a view on the organization model that focuses on highlighting actors, goals and tasks when a critical situation occurs. The description is enriched with information about the context in which the problem may occur and the impact on the standard stakeholder activities.

The aim of this protocol is to highlight every possible breakdown or problem that may occur in the organization that hinders the achievement of goals; this information — given to UCD designers — leads the construction of scenarios that, subsequently, have a specific significance for reasoning of system requirements in the creative sessions. Criticalities are initially classified and prioritized on the base of their relevance in the domain. Subsequently for each relevant criticality at least one scenario is authored and a cast of personas is engaged. The aim of the scenario is to highlight concrete instances in which the problem occurs, and to reveal stakeholders’ behavior in the circumstance.

4 Discussion

The first point we want to discuss in this paper is whether fragmentation activity is always possible or — as well as in the case of 'constructivist' approaches — it is a risk to reduce the advantages of methodology synergy. The difficulty to frame a design process within a precise formalization may hinder the applicability of situation method engineering techniques.

In our framework we maintained the two epistemologically different methodologies, by creating some communication channels for the teams of engineers and sociologists to easily communicate and share information. This activity required a deep analysis of the two approaches, in order to identify where the two methodologies present similarities and where they were conceptually different. At the beginning, for this purpose, the teams spent time in defining a common vocabulary. During these meetings the participants identified pairs of terms that may be re-conciliated (actor/persona, goal/need, task/activity), but they failed in identifying a precise relation between them even if relations have been investigated. The problem were not only dialectical, but epistemological: the two teams have different sensibility and a different vision about the problem and how to solve it. For instance, an actor identifies the 'abstraction' of a role in an organization, whereas a persona is an 'archetype' of users for which the system is going to be designed; the actor is featured with institutional goals that hold for every person will play the role, whereas each person is unique due to his/her personal attributes. This way the failure in the definition of a unified vocabulary raised up the need for the exploration of the boundary between component methodologies. Boundaries have to be interpreted as an additional value for the integration, because they allow for defining how to share design data even if talking different languages.

The second point of this discussion is the systematization of the approach we exploited. The goal is to reconcile the use of communication protocols with existing situational method engineering techniques for constructing methodologies. In our opinion it is possible to consider a communication protocol as a specific fragment, built ad-hoc for the specified situation. Considering, for instance, the criticality identification activity used to tie up the Tropos early requirement with personas/scenarios authoring: this activity did not exist neither in Tropos nor in UCD. The concept of critical aspect and the technique for identifying criticalities in the domain have been created ad-hoc for linking Tropos activities/concepts with UCD ones. Now, we are investigating whether a communication protocol can be framed inside the situational method engineering by introducing a loose methodological and linguistic link for integrating fragments. It is worth noting this is a loose relationship, that is different from a 'strong' structural relationship because it does not introduce destructive modifications. For instance, aggregation is the classic meta-model link among elements, and it is typically used when blending two meta-models (or portions) [6]. Another advantage of the 'communication' link is that it does not require a full formalization of the process and the products concerning the fragment. It may work even when the process model is

partially defined or in other cases — as UCD — in which the process frequently changes with the context and the language is flexible but ambiguous.

References

1. M. Cossentino, L. Sabatucci, and V. Seidita, “A collaborative tool for designing and enacting design processes,” in *SAC*, S. Y. Shin and S. Ossowski, Eds. ACM, 2009, pp. 715–721.
2. M. Cossentino, L. Sabatucci, V. Seidita, and S. Gaglio, “An agent oriented tool for method engineering,” in *EUMAS*, ser. CEUR Workshop Proceedings, B. Dunin-Keplicz, A. Omicini, and J. A. Padget, Eds., vol. 223. CEUR-WS.org, 2006.
3. A. Pickard and P. Dixon, “The applicability of constructivist user studies: how can constructivist inquiry inform service providers and systems designers,” *Information Research*, vol. 9, no. 3, pp. 9–3, 2004.
4. S. Brinkkemper, “Method engineering: engineering of information systems development methods and tools,” *Information and Software Technology*, vol. 38, no. 4, pp. 275–280, 1996.
5. B. Henderson-Sellers and J. Ralyté, “Situational Method Engineering: State-of-the-Art Review,” *Journal of Universal Computer Science*, vol. 16, no. 3, pp. 424–478, 2010.
6. M. Cossentino, S. Gaglio, A. Garro, and V. Seidita, “Method fragments for agent design methodologies: from standardisation to research,” *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 1, pp. 91–121, 2007.
7. V. Seidita, M. Cossentino, and S. Gaglio, “A Repository of Fragments for Agent Systems Design,” in *Proc. Of the Workshop on Objects and Agents (WOA’06)*, Catania, Italy, September 2006.
8. A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini, “The tropos metamodel and its use,” *Informatica*, vol. 29, no. 4, pp. 401–408, 2005.
9. W. Gaver, J. Beaver, and S. Benford, “Ambiguity as a resource for design,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM New York, NY, USA, 2003, pp. 233–240.
10. R. Weber, “The rhetoric of positivism versus interpretivism: A personal view,” *MIS Quarterly*, vol. 28 (1), pp. 3–12, 2004.
11. C. Potts and W. Newstetter, “Naturalistic inquiry and requirements engineering: reconciling their theoretical foundations,” in *Proc. of the Third IEEE International Symposium on Requirements Engineering*, 1997, pp. 118–127.
12. C. Hinds, “The case against a positivist philosophy of requirements engineering,” *Requirements Engineering*, vol. 13, no. 4, pp. 315–328, 2008.
13. T. Wolf, J. Rode, J. Sussman, and W. Kellogg, “Dispelling design as the black art of CHI,” in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, p. 530.
14. L. Penserini, A. Perini, A. Susi, and J. Mylopoulos, “High variability design for software agents: Extending Tropos,” *TAAS*, vol. 2, no. 4, 2007.
15. C. Rolland, C. Ben Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois *et al.*, “A proposal for a scenario classification framework,” *Requirements Engineering*, vol. 3, no. 1, pp. 23–47, 1998.
16. P. Wright, “What’s in a scenario?” *ACM SIGCHI Bulletin*, vol. 24, no. 4, p. 12, 1992.

17. A. Cooper, R. Reimann, and D. Cronin, *About face 3: the essentials of interaction design*. Wiley India Pvt. Ltd., 2007.
18. N. Mikkelsen and W. Lee, “Incorporating user archetypes into scenario-based design,” in *Proc. UPA*, 2000.
19. P. Junior and L. Filgueiras, “User modeling with personas,” in *Proceedings of the 2005 Latin American conference on Human-computer interaction*. ACM, 2005, p. 282.
20. J. Pruitt and J. Grudin, “Personas: practice and theory,” in *Proceedings of the 2003 conference on Designing for user experiences*. ACM New York, NY, USA, 2003, pp. 1–15.
21. Y. Chang, Y. Lim, and E. Stolterman, “Personas: from theory to practices,” in *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*. ACM, 2008, pp. 439–442.
22. J. Nieters, S. Ivaturi, and I. Ahmed, “Making personas memorable,” in *CHI’07 extended abstracts on Human factors in computing systems*. ACM, 2007, p. 1824.

**Pre-Proceedings of the Third International
Workshop on LAnguages, methodologies and
Development tools for multi-agent systemS**

LADS @ MALLOW 2010

Organised by

Mehdi Dastani, Utrecht University, The Netherlands
Amal El Fallah Seghrouchni, University of Paris VI, France
Jomi F. Hübner, Federal University of Santa Catarina, Brazil
João Leite, New University of Lisbon, Lisbon

Held with The Multi-Agent Logics, Languages, and Organisations
Federated Workshops (MALLOW 2010),
August 30th - September 2nd, Lyon, France

Preface

These are the pre-proceedings of the third international workshop on languages, methodologies and development tools for multi-agent systems (LADS'010). LADS'010 workshop aims to address both theoretical and practical issues related to developing and deploying multi-agent systems. In particular, it will constitute a rich forum where leading researchers from both academia and industry share their experiences on formal approaches, programming languages, methodologies, tools and techniques that support the development and deployment of multi-agent systems. From theoretical point of view, LADS'010 aims to address issues related to theories, models, and approaches that are needed to facilitate the development of multi-agent systems ensuring their predictability and verifications. From practical point of view, the workshop aims at stimulating research and discussion on how multi-agent system specifications and designs can be effectively implemented and tested. LADS'010 workshop promises to provide interesting discussion and exchange of ideas concerning theories, methodologies, techniques and principles that are important for multi-agent programming technology. The programme of the workshop consists of three sessions covering models, theories and tools for multi-agent systems. More details on the programme can be found at: <http://www.cs.uu.nl/lads2010>.

The co-chairs of this workshop would like to thank all authors, programme committee members, and additional reviewers for their outstanding contribution to the success of LADS'010. The co-chairs would also like to thank all the sponsors and Springer. We are particularly grateful to MALLOW 2010 organisers, Olivier Boissier, Amal El Fallah Seghrouchni, Salima Hassas, and Nicolas Maudet, for their technical support and for hosting the workshop.

LADS'010 Programme Co-chairs

Mehdi Dastani
Amal El Fallah Seghrouchni
Jomi F. Hübner
João Leite

August 31, 2010

Organisation

Organising Committee

Mehdi Dastani	Utrecht University, The Netherlands
Amal El Fallah Seghrouchni	University of Paris VI, France
Jomi F. Hübner	Federal University of Santa Catarina, Brazil
João Leite	New University of Lisbon, Lisbon

Steering Committee

Mehdi Dastani	Utrecht University, The Netherlands
Amal El Fallah Seghrouchni	University of Paris VI, France
João Leite	New University of Lisbon, Lisbon
Paolo Torroni	University of Bologna, Italy

Program Committee

Marco Alberti	New University of Lisbon, Portugal
José Júlio Alferes	New University of Lisbon, Portugal
Matteo Baldoni	University of Torino, Italy
Juan A. Botía	Murcia University, Spain
Lars Braubach	University of Hamburg, Germany
Yves Demazeau	Institut IMAG, Grenoble, France
Juergen Dix	Clausthal University, Germany
Paolo Giorgini	University of Trento, Italy
Koen Hindriks	Delft University, The Netherlands
Shinichi Honiden	NII, Tokyo, Japan
Wojtek Jamroga	Clausthal University, Germany
Peep Küngas	SOA Trader, Ltd., Tallin, Estonia
Brian Logan	University of Nottingham, UK
Alessio Lomuscio	Imperial College London, UK
Viviana Mascardi	University of Genova, Italy
John-Jules Meyer	Utrecht University, The Netherlands
Alexander Pokahr	University of Hamburg, Germany
Alessandro Ricci	University of Bologna, Italy
Patrick Taillibert	Thales Airborne Systems, Elancourt, France
Paolo Torroni	University of Bologna, Italy
Leon van der Torre	University of Luxembourg, Luxembourg
M. Birna van Riemsdijk	Delft University, The Netherlands
Pinar Yolum	Bogazici University, Istanbul, Turkey
Yingqian Zhang	Delft University, The Netherlands

Additional Reviewers

Natasha Alechina
Cristina Baroglio
Tristan Behrens
Akin Gunay
Ozgur Kafali
Yasuyuki Tahara

Table of Contents

ACRE: Agent Communication Reasoning Engine	7
<i>David Lillis, Rem Collier</i>	
OperettA: Organization-Oriented Development Environment	14
<i>Virginia Dignum, Huib Aldewereld</i>	
A Dialogic Dimension for the MOISE+ Organizational Model	21
<i>Alexandre Hübner, Graçaliz Dimuro, Antonio Carlos da Rocha Costa, Viviane Mattos</i>	
From Signed Information to Belief in Multi-Agent Systems	27
<i>Laurent Perrussel, Emiliano Lorini, Jean-Marc Thévenin</i>	
Towards efficient multi-agent abduction protocols	34
<i>Gauvain Bourgne, Katsumi Inoue, Nicolas Maudet</i>	
Validation of Agile Workflows using Simulation	41
<i>Kai Jander, Lars Braubach, Alexander Pokahr, Winfried Lamersdorf</i>	
JaCa-Android: An Agent-based Platform for Building Smart Mobile Applications	48
<i>Andrea Santi, Marco Guidi, Alessandro Ricci</i>	
Exploiting Agent-Oriented Programming for Building Advanced Web 2.0 Applications	55
<i>Mattia Minotti, Alessandro Ricci, Andrea Santi</i>	
Using HDS for Realizing Multi-Agent Applications	62
<i>Federico Bergenti, Enrico Franchi, Agostino Poggi</i>	
Author Index	69

ACRE: Agent Conversation Reasoning Engine

David Lillis

School of Computer Science and Informatics
University College Dublin

Email: david.lillis@ucd.ie

Rem W. Collier

School of Computer Science and Informatics
University College Dublin

Email: rem.collier@ucd.ie

Abstract—Within Multi Agent Systems, communication by means of Agent Communication Languages has a key role to play in the co-operation, co-ordination and knowledge-sharing between agents. Despite this, complex reasoning about agent messaging and specifically about conversations between agents, tends not to have widespread support amongst general-purpose agent programming languages.

ACRE (Agent Communication Reasoning Engine) aims to complement the existing logical reasoning capabilities of agent programming languages with the capability of reasoning about complex interaction protocols in order to facilitate conversations between agents. This paper outlines the aims of the ACRE project and gives details of the functioning of a prototype implementation within the AFAPL2 agent programming language.

I. INTRODUCTION

Communication is a vital part of a Multi Agent System (MAS). Agents make use of communication in order to aid mutual cooperation towards the achievement of their individual or shared objectives. The sharing of knowledge, objectives and ideas amongst agents is facilitated by the use of Agent Communication Languages (ACLs). The importance of ACLs is reflected by the widespread support for them in agent programming languages and toolkits, many of which have ACL support built-in as core features.

In many MASs, communication takes place by way of individual messages without formal links between them. An alternative approach is to group related messages into conversations: “task-oriented, shared sequences of messages that they observe, in order to accomplish specific tasks, such as a negotiation or an auction” [1].

This paper presents the Agent Conversation Reasoning Engine (ACRE). The principal aim of the ACRE project is to integrate interaction protocols into the core of existing agent programming languages. This is done by augmenting their existing reasoning capabilities and support for inter-agent communication by adding the ability to track and reason about conversations. Currently at the stage of an initial prototype, ACRE has been integrated with the AFAPL2 Agent Programming Language [2], which runs on the Agent Factory platform [3]. The longer-term goals of ACRE include its use within other mainstream programming languages.

The principal aim of this paper is to outline the goals of the ACRE project and to present the integration of the prototype system into AFAPL2.

This paper is laid out as follows: Section II outlines some related work on agent interaction. Section III then provides an overview of the aims and scope of the ACRE project. Following this, details of the integration of ACRE into the Agent Factory framework are given in Section IV. The relationships between message performatives and agent goals are discussed in Section V, followed by an example of a simple one-shot auction implemented via ACRE in Section VI. Finally, Section VII outlines some conclusions along with ideas for future work.

II. RELATED WORK

In the context of Agent Communication Languages, two standards have found widespread adoption. The Knowledge Query and Manipulation Language (KQML) was the firstly widely-adopted format for agent communication [4]. An alternative agent communication standard was later developed by the Foundation for Intelligent Physical Agents (FIPA). FIPA ACL utilises what it considers to be a minimal set of English verbs that are necessary for agent communication. These are used to define a set of performatives that can be used in ACL messages [5]. These performatives, along with their associated semantics, are defined in [6].

Recognising that one-off messages are limited in their power to be used in more complex interactions, FIPA also defined a set of interaction protocols [7]. These are designed to cover a set of common interactions such as one agent requesting information from another, an agent informing others of some event and auction protocols.

Support for either KQML or FIPA ACL communication is frequently included as a core feature in many agent toolkits and frameworks, native support for interaction protocols is less common. The JADE toolkit provides specific implementations of a number of the FIPA interaction protocols [8]. It also provides a Finite State Machine (FSM) behaviour to allow interaction protocols to be defined. Jason includes native support for communicative acts, but does not provide specific tools for the development of agent conversations using interaction protocols. This is left to the agent programmer [9, p. 130]. A similar level of support is present within the Agent Factory framework [10].

There do exist a number of toolkits, however, that do include support for conversations. For example, the COOrdination

Language (COOL) uses FSMs to represent conversations [11]. Here, a conversation is always in some state, with messages causing transitions between conversation states. Jackal [12] and KaOS [13] are other examples of agent systems making use of FSMs to model communications amongst agents. Alternative representations of Interaction Protocols include Coloured Petri Nets [14] and Dooley Graphs [15].

III. ACRE OVERVIEW

ACRE is aimed at providing a comprehensive system for modelling, managing and reasoning about complex interactions using protocols and conversations. Here, we distinguish between a *protocols*’ and *conversations*. A protocol is defined as a set of rules that dictate the format and ordering of messages that should be passed between agents that are involved in prolonged communication (beyond the passing of a single message). A conversation is defined as a single instance of multiple agents following a protocol in order to engage in communication. It is possible for two agents to engage in multiple conversations that follow the same protocol.

Such an aim can only be realised effectively if a number of features are already available. These include:

- **Protocol definitions understandable by agents:** Interaction protocols must be declared in a language that all agents must be able to understand and share. This also has the advantage that the protocol definition is separated from its implementation in the agent, thus providing a programmer with a greater understanding of the format the communication is expected to take. ACRE uses an XML representation of a finite state machine for this purpose.
- **Shared ontologies:** A shared vocabulary is essential to agents understanding each other’s communications. A shared ontology defines concepts about which agents need to be capable of reasoning.
- **Plan repository:** With the two above features in place, an agent may reason about the sequence of messages being exchanged, as well as the content of those messages. This reasoning will typically result in an agent deciding to perform some action as a consequence of receiving certain communications. In this case, it is useful to have available a shareable repository of plans that agents may perform so that new capabilities may be learned from others.

The presence of these features aid greatly in the realisation of ACRE’s aims. The principal aims are as follows:

- **External Monitoring of Interaction Protocols:** At its simplest level, conversation matching and recognition of interaction protocols allows for a relatively simple tool operating externally to any of the agents. This can intercept and read messages at the middleware level and is suitable for an open MAS in which agents communicate via FIPA ACL. This is a useful tool for debugging purposes, allowing developers to monitor communication to ensure that agents are following protocols correctly. This is particularly important where conversation management has been implemented in an ad-hoc way, with incoming

and outgoing messages being treated independently and without a strong notion of conversations.

- **Internal Conversation Reasoning:** On receipt of a FIPA ACL message, it should be possible to identify the protocol being followed by means of the `protocol` parameter defined in the message (for the specification of the parameters available in a FIPA ACL message see [16]). Similarly, the initiator of a conversation should also set the `conversation-id` parameter, which is a unique identifier for a conversation. By referring to the the protocol identifier, an agent can make decisions about its response by consulting the protocol specification. Similarly, the conversation identifier may be matched against the stored history of ongoing conversations. ACRE aims to use this information to analyse the status of conversations and generate appropriate goals for the agent to successfully continue the conversation along the appropriate lines for the protocol that is specified. The use of goals follows [17]. Goals represent the motivations of the participants in a conversation. Thus the agents’ engagement in a particular conversation is decoupled from the individual messages that are being exchanged, allowing greater flexibility in reasoning about their reactions and responses.
- **Organisation of Incoming Messages:** It is possible that an agent communicating with agents in another system may receive messages that do not specify their protocol and/or conversation identifier. In this case, it is useful for the agent to have access to definitions of the protocols in which it is capable of engaging so as to match these with incoming messages so as to categorise the messages.
- **Agent Code Verification:** The ultimate aim of ACRE is to facilitate the verification of certain aspects of agent code. In particular, given integration of conversation reasoning into a programming language, it should be possible to verify whether or not an agent is capable of engaging in a conversation following a particular protocol.

IV. AGENT FACTORY

Agent Factory is an extensible, modular and open framework for the development of multi agent systems [3]. The primary agent programming language packaged with Agent Factory is AFAPL2 [2], although it also includes support for other agent programming languages such as ALPHA [18] and AgentSpeak [9].

This principal aim of this paper is to outline the integration of ACRE with AFAPL2. AFAPL2 is an agent programming language that was initially based on the Agent0 language, with notions of belief and commitment at its core [19]. Its capabilities have been augmented since, however, with the addition of such features as goal reasoning [20] and roles [10].

The existing goal-reasoning capabilities of AFAPL2 (outlined in [20]) required some extension in order to be usable for the purposes of ACRE.

AFAPL2 contains two types of activities (code that allows an agent to perform some task): actions and plans. An action is

a simple activity that is implemented by way of a single Java class, known as an actuator. Actions are designed to be used as primitive activities that can be grouped together to carry out more complex tasks. A plan is such a grouping, making use of plan operators (such as operators to carry out several actions in sequence or in parallel) to combine actions. Each activity has three components:

- A *precondition* that specifies the circumstances in which the activity may be executed. This is expressed in terms of beliefs that the agent must have when attempting to execute the activity.
- A *postcondition* that indicates the anticipated mental state on successful completion of the activity. This is expressed in terms of beliefs the agent will expect to have once the activity has completed.
- The *body* indicates how the activity can be carried out: for actions this is a Java class name whereas for plans this is the expression of how the actions are combined for a more complex activity.

In the existing implementation of goal-handling, goals are achieved by comparing them with the postconditions of the activities that the agent is capable of performing. Figure 1 shows an example definition of a plan designed to check whether a host (identified by an IP address contained in the `?ip_addr` variable) is responding to ping requests (the actual code implementing the plan is omitted). The precondition `BELIEF(true)` is always satisfied. The postcondition `BELIEF(pingStatus(?ip_addr, ?status))` indicates that on successful execution of this plan, the agent will expect to have a belief about the status of the IP address that it attempted to check.

```

PLAN checkPingStatus(?ip_addr) {
  PRECONDITION BELIEF(true);
  POSTCONDITION BELIEF(pingStatus(?ip_addr, ?status));
  ...
}

```

Fig. 1. AFAPL2 Plan Definition (plan body omitted)

`GOAL(pingStatus(192.168.1.1, ?status))` indicates that the agent aims to have a belief about the status of the host with the IP address `192.168.1.1`. This interpretation of the goal would be contained in the relevant ontology. Here, `?status` is a variable (indicated by the `? sigil`) that can match against anything. Thus it is not a goal to bring about a particular status; rather just to find out what that status is.

An agent having this goal would identify the `checkPingStatus` plan to be a candidate plan for its achievement.. This is the case for two reasons. Firstly, its postcondition matches the goal, meaning that the agent will anticipate its goal being achieved by a successful execution of this plan and secondly because its precondition is satisfied by the current belief set of the agent (since an agent will always believe `true` to be true). In deciding on the appropriate course of action, the goal reasoning engine will identify all such candidate activities and execute one. In the event that no candidate activities can be found, the goal is dropped as unachievable.

A significant drawback with this method of reasoning is that if no activity is available that can directly result in a goal state being brought about, no further effort is made to achieve it. However, this does not necessarily mean that the agent is incapable of achieving its goal. In the event of an activity being identified whose postcondition is expected to satisfy the goal but whose precondition is not satisfied by the current state of the agent, the modified goal reasoning engine examines other activities to evaluate whether any are available that can satisfy that precondition. An example of this reasoning process is given in Section VI.

V. MAPPING PERFORMATIVES TO GOALS AND BELIEFS

In AFAPL2, the existing method of handing message receipts is simply to adopt a belief that the message has been received, leaving it as an exercise to the application programmer to deal with this event. One reason behind this method is that there is currently no support for messages to be linked into conversations. In contrast, ACRE can analyse the conversations and protocols about which the agent is aware and generate more appropriate goals and beliefs whenever messages are received and sent.

The goals or beliefs that are generated depend on the context within which a message is sent. For example, a `propose` message is used to indicate that the sender proposes to perform some action under certain conditions. There are, however, more than one reason why an agent may receive such a message. In one situation, the proposal is unsolicited (for example to initiate a FIPA Propose Interaction Protocol [21]). In this case, the message has no prior context and is unrelated to any previous experience of the recipient. By its nature, a propose message requires a response and so the recipient agent must evaluate the proposal and communicate whether or not it is willing to accept the proposal. As such, this situation will result in the adoption of a goal indicating that this type of evaluation should take place.

In contrast, a proposal may have been solicited by the recipient. The message may be matched to an existing conversation, either by means of an explicit conversation ID or by matching its content against that expected by the relevant protocol. By analysing this conversation, the agent can identify whether or not a call for proposals was previously sent out. In sending such a call, the agent will have been pursuing some other goal and so the adoption of an additional goal to handle the proposal is not desirable. Instead, a belief is adopted to indicate that the proposal has been received.

This approach also allows the agent to engage in separate but related conversations with different agents concurrently, as is shown in the example in Section VI.

Another example of the run-time conversation reasoning is on the receipt of an `accept-proposal` message. In this case, the treatment is different because of the future messages that the relevant protocol may or may not require to be sent in response. Under some protocols, an `accept-proposal` message is the final message in the conversation (e.g. the FIPA Propose Interaction Protocol [21] or the Vickrey Auction shown in Section VI). Here, a goal should be adopted merely

to perform the task that has been proposed and accepted. No further communication is required.

In other cases, such as within a FIPA Contract Net Interaction Protocol [22], the recipient of the `accept-proposal` message is required to communicate the result of performing the stated action back to the sender. In this case, the goal to be satisfied is twofold: firstly to perform the action and then communicate the result of this action to the sender. In reality, only one goal is necessary, as it is impossible to communicate the result of an action that has not been committed. This should be reflected in the preconditions of any activity that communicates the result of an action.

In the case of the sender of a message, it is not necessary to generate these goals. Here, the message is sent by the agent as a result of it having a goal that must be satisfied.

VI. EXAMPLE: VICKREY AUCTION

In order to demonstrate how the ACRE system works, we use a Vickrey Auction Interaction Protocol. Figure 2 illustrates the protocol using Agent UML [23].

A Vickrey auction is a non-iterated auction, in that each bidder submits only a single bid, which is either accepted or rejected. It is also a sealed-bid auction, in that bids are communicated only to the auctioneer. In a Vickrey Auction, the winner of the auction is the bidder who submits the highest bid, though the ultimate price paid is equal to the second-highest bid.

In this example, one agent is assumed to desire that a task be performed by another agent and requests other agents to submit proposals for the performance of this task. This agent is referred to as the “Auctioneer”. The auction is initiated by the Auctioneer sending a `cfp` message to a number of potential “Bidder” agents. Each bidder considers the call for proposals and decides whether or not to participate in the auction. Having done so, it indicates its decision to the Auctioneer either by submitting a bid (via a `propose` message) or by explicitly declining to do so (using a `refuse` message).

After receiving all of these responses, the Auctioneer must decide which is the winner of the auction and communicate its decision to each of the Bidders. This is done by sending a `accept-proposal` message to the successful bidder and a `reject-proposal` message to those that are unsuccessful.

A. ACRE Implementation Example

As noted in the above section, a Vickrey auction is typically initiated by an agent that wishes to have some task performed by another agent. This will generally be indicated by the agent adopting a goal to have the task performed.

In this example, we consider a MAS consisting of agents that are situated in a virtual grid world that contains items that the agents are required to collect. We begin the case study in a situation where one agent (which will become the Auctioneer agent) has discovered the location of an item and wishes to have it collected. This is reflected by the adoption of a goal, which is shown in Figure 4.

The addition of this goal to the agent’s mental state will cause the goal reasoning engine to evaluate the options open

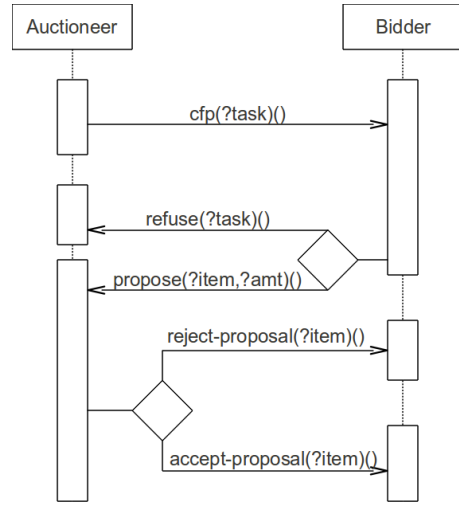


Fig. 2. AUML Diagram for a Vickrey-style auction



Fig. 4. Initial goal to trigger a Vickrey Auction

to it to satisfy this goal. One option may be to execute a plan such as that defined in Figure 5. This is a plan that allows the agent to carry out the task (i.e. collect the referenced item) itself, without the need for engaging in conversation with other agents. However, it may alternatively be the case that the agent is not capable of performing the collection itself (if, for example, it is a coordinator of other agents). In such a scenario, it may be necessary to engage with other agents in order to find another that will be capable of (and willing to) carry out the task instead. The holding of an auction is one common way of solving such a problem.

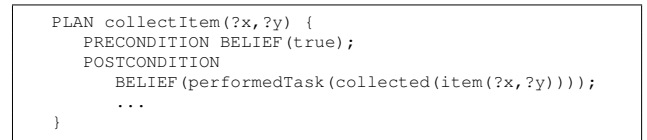


Fig. 5. Plan Definition to allow an agent collect items (plan body omitted)

Sample code to implement an Auctioneer agent is presented in Figure 3. This include two plans used in the implementation of a Vickrey Auction. In addition to the two plans, the agent also includes an *AuctionModule*, which contains the code to reason about the bids that have been received and decide upon a winner. Two actuators are also present: one (`addBid`) to add a received bid to the *AuctionModule* and the other (`endAuction`) to trigger the ending of the auction and cause a winner to be decided upon. Finally, a perceptor is also present (`auctionPerceptor`) that monitors the state of the *AuctionModule* and adopts beliefs based thereon. These include beliefs about who the winners and losers of the auction are, following the end of the auction.

As outlined in Section IV, the goal reasoning engine firstly seeks an activity (either an action or a plan) whose postcon-


```

IMPORT com.agentfactory.afapl2.core.agent.FIPACore;
IMPORT agent.ACREAgent;

PLAN cfpTaskSolver(?task) {
  PRECONDITION BELIEF(haveProposal(bidfor(?task,?bid),?agentID,?cid));
  POSTCONDITION BELIEF(performedTask(?task));

  BODY
    SEQ (
      FOREACH ( haveProposal(bidfor(?task,?amount),?agentID,?cid),
        addBid(?task,?agentID,?amount,?cid),
      ),
      endAuction,
      FOREACH ( BELIEF(status(?task,?agentID,winner)),
        accept-proposal(?agentID,?task)
      ),
      FOREACH ( BELIEF(status(?task,?agentID,loser)),
        reject-proposal(?agentID,?task)
      ),
      ADOPT(performedTask(?task))
    );
}

PLAN solicitProposals(?task) {
  PRECONDITION BELIEF(neighbour(agentID(?aname,?aaddr)));
  POSTCONDITION BELIEF(haveProposal(bidfor(?task,?bid),agentID(?aname,?aaddr),?cid));

  BODY
    FOREACH ( BELIEF(neighbour(?agentID)),
      SEQ (
        cfp(?agentID,bidfor(?task)),
        OR (
          AWAIT(BELIEF(haveProposal(?bid,agentID(?aname,?aaddr),?cid))),
          AWAIT(BELIEF(haveRefusal(?task,agentID(?aname,?aaddr),?cid))),
          SEQ( DELAY(20), ADOPT(BELIEF(timeout(?agentID))))
        )
      )
    );
}

LOAD_MODULE AuctionModule module.AuctionModule;

PERCEPTOR auctionPerceptor {
  CLASS perceptor.AuctionPerceptor;
}

ACTION endAuction {
  CLASS actuator.EndAuctionActuator;
}

ACTION addBid( ?task, ?agentID, ?amount, ?cid ) {
  actuator.AddBidActuator;
}

```

Fig. 3. AFAPL2 Auctioneer Agent

dition satisfies the goal. In this example, the postcondition of the `cfpTaskSolver` plan will match the goal. This postcondition contains the variable `?task`, which is matched against the goal. This has the effect that the plan will be invoked with `collected(item(20,25))` set as the value for the `?task` variable.

However, this plan by itself will not be capable of bringing about successful achievement of the goal. This is because it also has a precondition that indicates that in order for the plan

to succeed, the agent must already believe that it has received at least one other proposal from another agent to perform the task. As this is not the case, the goal reasoner must identify another activity that will bring about that precondition.

The `solicitProposals` plan has a postcondition that satisfies the precondition of `cfpTaskSolver` and is executable if the agent is aware of at least one neighbouring agent that it can invite to the auction. Thus the strategy the Auctioneer will employ will be to firstly execute

```

IMPORT agent.ACREAgent;

PLAN cfpProposal(?task, ?initiator, ?cid) {
  PRECONDITION BELIEF(canBid(?task, ?amount, ?cid));
  POSTCONDITION BELIEF(respondedToCfp(bidfor(?task), ?initiator, ?cid));

  BODY
    propose(?initiator, bid(?task, ?amount));
}

PLAN cfpRefusal(?task, ?initiator, ?cid) {
  PRECONDITION BELIEF(noBid(?task, ?cid));
  POSTCONDITION BELIEF(respondedToCfp(bidfor(?task), ?initiator, ?cid));

  BODY
    refuse(?initiator, bid(?task));
}

ACTION generateBid( ?task, ?cid ) {
  PRECONDITION BELIEF(conversation(?cid, acre-vickrey));
  POSTCONDITION BELIEF(canBid(?task, ?amount, ?cid));

  CLASS is.lill.acre.actuator.GenerateBidActuator;
}

```

Fig. 6. AFAPL2 Bidder Agent

`solicitProposals` and then `cfpTaskSolver` in the expectation that the goal will be satisfied afterwards (by another agent performing the task).

The body of `solicitProposals` firstly considers all of the agents it has knowledge of (the `FOREACH` plan operator will execute the following code in parallel for every belief in the agent's belief set that matches `BELIEF(neighbour(?agentID))`, where `?agentID` can be bound in turn to the contents of each belief). For each of these agents it firstly sends a message to initiate the auction (the `cfp` action is part of the standard `FIPACore` agent that is imported at the top of the file). It then either waits until one of the following events has occurred: a) it believes it has received a proposal from the bidder, b) it believes that it has received a rejection from the bidder or c) some timeout period elapses, following which it adopts a belief to that effect. Once one of these things has occurred, the plan has completed.

It is important to note at this stage that the postcondition of `solicitProposals` may not be satisfied by its execution. The postcondition is designed to indicate the intended result of the plan, rather than enumerating all of its possible outcomes. In this case, the purpose of the plan is to solicit bids from other agents as part of an auction. Although it is possible that all agents could refuse to participate or fail to respond, it is not logical for an agent to issue a call for proposals in the hope that this will occur. From a goal-reasoning point of view, if no bids are received then the precondition of `cfpTaskSolver` is not satisfied and the goal is considered to be unsolvable. This is a logical outcome since the agent has no capability of performing the task itself and has failed to find another agent that is willing to do so on its behalf.

The beliefs about the receipt of a proposal or refusal are generated by `ACRE` reasoning about the conversations. This is an example of the situation presented in Section V where

`ACRE` is aware that the proposal or refusal are in response to a call for proposals that was issued by the Auctioneer and so generates a belief rather than a goal.

If at least one bid is received then the precondition of `cfpTaskSolver` is satisfied and that plan may be executed. In this plan, the Auctioneer evaluates each of the proposals it has received and adds it to the `AuctionModule` that takes care of the decision-making with regard to the winner of the auction. Once all of the bids have been added, the auction can be ended. The auction perceptor will cause a set of beliefs about the auction to be adopted. These are used to send `accept-proposal` messages to the winner and `reject-proposal` messages to each of the losers of the auction.

In the context of the auctioneer, one advantage of this approach is that these plans are not limited to use within a Vickrey Auction. For example, a Contract Net Protocol [22] is also initiated by sending a `cfp` message and awaiting a response by means of either a `propose` or `refuse` message.

Figure 6 contains the AFAPL2 code for the Bidder agents. These agents must respond to the `cfp` message sent by the Auctioneer to initiate the auction. This is done by means of `ACRE` posting an appropriate goal for the agent's goal reasoner to solve. In this example, the goal is satisfied by the identical postconditions of the `cfpProposal` and `cfpRefusal` plans. However, there is no activity available with a postcondition that matches the precondition of `cfpRefusal`. On the other hand, `cfpProposal`'s precondition can be satisfied by executing the `generateBid` action (providing that its precondition that the conversation, represented by the variable `?cid` is of the type "acre-vickrey"). This is executed, followed by `cfpProposal`, assuming a belief that the agent is in a position to bid has been created.

The `generateBid` action may, however, cause the agent

to decline to make a bid (indicated by adopting a belief of the type `noBid`). This would mean that the precondition for the `cfpProposal` plan has not been satisfied and so it cannot be executed to satisfy the goal. At this point, the goal reasoner will re-evaluate the goal against the current belief set of the agent and, on finding the belief that the agent will not make a bid, now sees that the precondition of `cfpRefusal` is already satisfied by the current mental state of the agent. Thus this plan is called instead, causing a `refuse` message to be sent to the Auctioneer.

This example demonstrates one drawback of the use of postconditions in AFAPL2 to indicate the desired outcomes of activities. In this case, planning would be better facilitated by the express inclusion of `noBid` as a belief that will be adopted as an alternative outcome of the `generateBid` plan. As no express support is available for the enumeration of byproducts of activities (or the beliefs associated with a plan failing in its purpose).

No particular code is required to handle the response from the Auctioneer. In the event of the receipt of a `accept-proposal` message, this indicates that the Bidder is required to carry out some task that it has proposed to do, and so ACRE will adopt a goal to that effect. A `reject-proposal`, on the other hand, does not require any further action from the Bidder, so ACRE will merely adopt a belief to that effect that can be reasoned about by the agent.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents a prototype of the ACRE conversation reasoning system and specifically its integration into the AFAPL2 agent programming language. Although currently limited to AFAPL2, it is intended that ACRE will be used in conjunction with several other agent programming languages.

Although full integration with several languages is desirable, it may be necessary to adapt ACRE's workings to the specific needs and capabilities of particular languages. For example, not all languages support the use of preconditions and postconditions of activities to facilitate reasoning about them. On the other hand, support for ACL standards is widespread and so the grouping of messages into conversations is part of ACRE that is likely to be more widely applicable in its current form.

The availability of cross-platform communication tools such as ACRE, together with shared ontologies and protocol definitions can only aid interoperability between distinct agent platforms, toolkits and programming languages.

REFERENCES

- [1] Y. Labrou, "Standardizing agent communication," *Multi-Agents Systems and Applications (Advanced Course on Artificial Intelligence)*, pp. 74–97, 2001. [Online]. Available: <http://portal.acm.org/citation.cfm?id=567252>
- [2] C. Muldoon, G. O'Hare, R. W. Collier, and M. O'Grady, *Towards Pervasive Intelligence: Reflections on the Evolution of the Agent Factory Framework*. Boston, MA: Springer US, 2009, ch. 6, pp. 187–212. [Online]. Available: <http://www.springerlink.com/content/g813865gq77731p1>
- [3] R. Collier, G. O'Hare, T. Lowen, and C. Rooney, "Beyond Prototyping in the Factory of Agents," in *Multi-Agent Systems and Application III: 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003)*, Prague, Czech Republic, 2003.
- [4] T. Finin, R. Fritzson, D. McKay, and R. McEntire, "KQML as an Agent Communication Language," in *Proceedings of the Third International Conference on Information and Knowledge Management*, Gaithersburg, MD, 1994, pp. 456–463.
- [5] S. Poslad, P. Buckle, and R. Hadingham, "The FIPA-OS Agent Platform: Open Source for Open Standards," in *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (PAAM2000)*, Manchester, 2000, p. 368.
- [6] Foundation for Intelligent Physical Agents, "FIPA Communicative Act Library Specification," 2002. [Online]. Available: <http://www.fipa.org/specs/fipa00037/>
- [7] —, *FIPA Interaction Protocol Library Specification*, Std., 2000. [Online]. Available: <http://www.fipa.org/specs/fipa00025/>
- [8] F. Bellifemine, G. Caire, T. Trucco, and G. Rimass, "Jade Programmer's Guide," 2007. [Online]. Available: <http://jade.tilab.com/doc/programmersguide.pdf>
- [9] R. H. Bordini, J. F. Hübner, and M. J. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. Wiley-Interscience, 2007. [Online]. Available: <http://books.google.com/books?hl=en&lr=&id=AJHD4GkIQs0C&pgis=1>
- [10] R. Collier, R. Ross, and G. M. P. O'Hare, "A Role-Based Approach to Reuse in Agent-Oriented Programming," in *AAAI Fall Symposium on Roles, an Interdisciplinary Perspective (Roles 2005)*, Arlington, VA, USA, 2005.
- [11] M. Barbuceanu and M. S. Fox, "COOL: A language for describing coordination in multi agent systems," in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 1995, pp. 17–24.
- [12] S. Cost, T. Finin, Y. Labrou, X. Luan, Y. Peng, I. Soboroff, J. Mayfield, and A. Boughannam, "Jackal: a Java-based Tool for Agent Development," in *Working Papers of the AAAI-98 Workshop on Software Tools for Developing Agents*. AAAI Press, 1998.
- [13] J. M. Bradshaw, S. Dutfield, P. Benoit, and J. D. Woolley, "KAoS: Toward an industrial-strength open agent architecture," *Software Agents*, pp. 375–418, 1997.
- [14] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng, "Modeling agent conversations with colored petri nets," in *In: Workshop on Specifying and Implementing Conversation Policies, Third International Conference on Autonomous Agents (AGENTS '99)*, Seattle, 1999, pp. 59–66. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.6521>
- [15] H. Parunak, "Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis," in *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS)*, 1996.
- [16] Foundation for Intelligent Physical Agents, "FIPA ACL Message Structure Specification," 2002. [Online]. Available: <http://www.fipa.org/specs/fipa00061/>
- [17] L. Braubach and A. Pokahr, "Goal-Oriented Interaction Protocols," in *MATES '07: Proceedings of the 5th German Conference on Multiagent System Technologies*, vol. 4687. Leipzig, Germany: Springer, 2007, pp. 85–97.
- [18] R. Collier, R. Ross, and G. M. P. O'Hare, "Realising Reusable Agent Behaviours with ALPHA," in *Proceedings of the 3rd German Conference on Multi-Agent System Technologies (MATES 05)*, Koblenz, Germany, 2005, pp. 210–215.
- [19] Y. Shoham, "Agent0: An agent-oriented programming language and its interpreter," *Journal of Object-Oriented Programming*, vol. 8, no. 4, pp. 19–24, 1991.
- [20] M. Dragone, D. Lillis, R. W. Collier, and G. M. P. O'Hare, "Practical Development of Hybrid Intelligent Agent Systems with SoSAA," in *Proceedings of the 20th Irish Conference on Artificial Intelligence and Cognitive Science*, Dublin, Ireland, August 2009.
- [21] Foundation for Intelligent Physical Agents, "FIPA Propose Interaction Protocol Specification," 2002. [Online]. Available: <http://www.fipa.org/specs/fipa00036>
- [22] Foundation For Intelligent Physical Agents, "FIPA Contract Net Interaction Protocol Specification," 2002. [Online]. Available: <http://www.fipa.org/specs/fipa00029>
- [23] B. Bauer, J. Müller, and J. Odell, "Agent UML: A Formalism for Specifying Multiagent Software Systems," *Int. Journal of Software Engineering and Knowledge Engineering*, vol. 11, no. 3, pp. 207–230, 2001.

OperettA: Organization-Oriented Development Environment

Virginia Dignum

Delft University of Technology, The Netherlands, m.v.dignum@tudelft.nl

Huib Aldewereld

Utrecht University, The Netherlands, huib@cs.uu.nl

Abstract—The increasing complexity of distributed applications requires new modeling and engineering approaches. Such domains require representing the regulating structures explicitly and independently from the acting components (or agents). Organization computational models, based on Organization Theory, have been advocated to specify such systems. In this paper, we present the organizational modeling approach OperA and a graphical environment for the specification and analysis of organizational models, OperettA. OperA provides an expressive way for defining open organizations distinguishing explicitly between the organizational aims, and the agents who act in it.

I. INTRODUCTION

The engineering of applications for complex and dynamic domains is an increasingly difficult process. Requirements and functionalities are not fixed a priori, components are not designed nor controlled by a common entity, and unplanned and unspecified changes may occur during runtime. There is a need for representing the regulating structures explicitly and independently from the acting components (or agents). Organization computational models, based on Organization Theory, have been advocated to specify such systems.

Traditionally, Multi-Agent Systems (MAS) stress the autonomy and encapsulation characteristics of agents. In such agent-centric view, interactions between agents are mostly seen as speech acts whose meaning may be described in terms of the mental states of an agent. As systems grow to include hundreds or thousands of agents, it is necessary to move from an agent-centric view of coordination and control to an organization-centric one. Organizations provide stable means for coordination that enable the achievement of global goals. In this sense, organization models play a critical role in the development of larger and more complex MAS [5].

Comprehensive analysis of several agent systems has shown that different design approaches are appropriate for different domain characteristics [6]. In particular, agent organization frameworks are suitable to model complex environments where many independent entities coexist with explicit normative and organizational structures and global specification of control measures is necessary.

Models for agent organizations must, on the one hand, be able to specify global goals and requirements but, on the other hand, cannot assume that participating actors will always act according to the needs and expectations of the system design. Concepts as organizational rules [21], norms and institutions

[9], [10], and social structures [15] arise from the idea that the effective engineering of organizations needs high-level, actor-independent concepts and abstractions that explicitly define the organization in which agents live [22]. These are the rules and global objectives that govern the activity of an organization.

The OperA model [4] proposes an expressive way for defining agent organizations distinguishing explicitly between the organizational aims, and the agents who act in it. OperA enables the specification of organizational structures, requirements and objectives, and at the same time allows participants to have the freedom to act according to their own capabilities and demands. OperA has been applied in many domains, including knowledge management, work practice analysis, and serious games and social simulation. Space constraints do not allow for a comparison of OperA with other approaches. For this effect, we refer the reader to [2].

In this paper, we present OperettA, a graphical environment for the specification, validation and analysis of organizational models, based on the OperA formalism [4]. This organization specification tool builds heavily on mechanisms from Model Driven Engineering (MDE), which enables the introduction and combination of different formal methods hence enabling the modeling activity through systematic advices and model design consistency checking.

The paper is organized as follows: first, in section II we introduce and briefly explain the OperA framework. In section III the specification of organizational models in OperA is detailed. In section IV we introduce the OperettA Environment. In section V we provide some design guidelines for organization models. Finally, section VI gives our conclusions.

II. ORGANIZATION MODELING: THE OPERA FRAMEWORK

Organizational models should enable the explicit representation of structural and strategic concerns and their adaptation to environment changes in a way that is independent from the behavior of the agents. Organization models, combining global requirements and individual initiative, have been advocated to specify open systems in dynamic environments [11], [4]. We take formal processes and requirements as a basis for the modeling of complex systems that regulate the action of the different agents.

The deployment of organizations in dynamic and unpredictable settings brings forth critical issues concerning the

design, implementation and validation of their behavior [16], [13], [20], and should be guided by two principles.

- Provide sufficient representation of the institutional requirements so that the overall system complies with the norms.
- Provide enough flexibility to accommodate heterogeneous components.

Therefore, organizational models must provide means to represent concepts and relationships in the domain that are rich enough to *cover* the necessary contexts of agent interaction while keeping in mind the *relevance* of those concepts for the global aims of the system.

The OperA model [4] proposes an expressive way for defining open organizations distinguishing explicitly between the organizational aims, and the agents who act in it. That is, OperA enables the specification of organizational structures, requirements and objectives, and at the same time allows participants to have the freedom to act according to their own capabilities and demands. At an abstract level, an OperA model describes the aims and concerns of the organization with respect to the social system. These are described as organization’s externally observable *objectives*, that is, the desired states of affairs for the organization.

The OperA framework consists of three interrelated models. The **Organizational Model** (OM) is the result of the observation and analysis of the domain and describes the desired behavior of the organization, as determined by the organizational stakeholders in terms of objectives, norms, roles, interactions and ontologies. The OM provides the overall organization design that fulfills the stakeholders requirements. Objectives of an organization are achieved through the action of agents, which means that, at each moment, an organization should employ the relevant agents that can make its objectives happen. However, the OM does not specify how to structure groups of agents and constrain their behavior by social rules such that their combined activity will lead to the desired results. The **Social Model** (SM) maps organizational roles to agents and describes agreements concerning the role enactment and other conditions in social contracts. Finally, the **Interaction Model** (IM) specifies the interaction agreements between role-enacting agents as interaction contracts. IM specification enable variations to the enactment of interactions between role-enacting agents.

The OperettA framework is developed to specify organization models, according to the OperA OM, which will be described in more detail in section III, using as example the conference organization scenario taken from [8]. In section IV-B we describe the use of MDE principles to implement this framework.

III. THE ORGANIZATION MODEL

A common way to express the objectives of an organization is in terms of its expected functionality, that is, what is the organization expected to do or produce. In OperA, the Organization Model (OM) specifies the *means* to achieve such objectives. That is, OM describes the structure and global

characteristics of a domain from an organizational perspective, where global goals determine roles and interactions, specified in terms of *Social* and *Interaction Structures*. E.g., how should a conference be organized, its program, submissions, etc.

Moreover, organization specification should include the description of concepts holding in the domain, and of expected or required behaviors. Therefore, these structures should be linked with the norms, defined in *Normative Structure*, and with the ontologies and communication languages defined in the *Communication Structure*.

A. The Social Structure.

The social structure of an organization describes the roles holding in the organization. It consists of a list of role definitions, *Roles* (including their objectives, rights and requirements), such as PC-member, program chair, author, etc.; a list of role groups’ definitions, *Groups*; and a *Role Dependencies* graph.

Abstract society objectives form the basis for the definition of the objectives of roles. *Roles* are the main element of the *Social Structure*. From the society perspective, role descriptions should identify the activities and services necessary to achieve society objectives and enable to abstract from the individuals that will eventually perform the role. From the agent perspective, roles specify the expectations of the society with respect to the agent’s activity in the society. In OperA, the definition of a role consists of an identifier, a set of role objectives, possibly sets of sub-objectives per objective, a set of role rights, a set of norms and the type of role. An example of role description is presented in table I.

<i>Id</i>	PC_member
<i>Objectives</i>	paper_reviewed(Paper,Report)
<i>Sub-objectives</i>	{read(P), report_written(P, Rep), review_received(Org, P, Rep)}
<i>Rights</i>	access-confmanager-program(<i>me</i>)
<i>Norms & Rules</i>	PC_member is OBLIGED to understand English IF paper_assigned THEN PC_member is OBLIGED to review paper BEFORE given deadline IF author of paper_assigned is colleague THEN PC_member is OBLIGED to refuse to review asap

TABLE I
PC member ROLE DESCRIPTION.

Groups provide means to collectively refer to a set of roles and are used to specify norms that hold for all roles in the group. Groups are defined by means of an identifier, a non-empty set of roles, and group norms. An example of a group in the conference scenario is the organizing team consisting of the roles *program chair*, *local organizer*, and *general chair*.

The distribution of objectives in roles is defined by means of the *Role Hierarchy*. Different criteria can guide the definition of *Role Hierarchy*. In particular, a role can be refined by decomposing it in sub-roles that, together, fulfill the objectives of the given role.

This refinement of roles defines *Role Dependencies*. A dependency graph represents the dependency relations between roles. Nodes in the graph are roles in the society.

Arcs are labelled with the objectives for which the parent role depends on the child role. Part of the dependency graph for the conference society is displayed in figure 1. For example, the arc between nodes PC-Chair and PC-member represents the dependency between *PC-Chair* and *PC-member* concerning *paper-reviewed* ($PC - Chair \succeq_{paper_reviewed} PC - Member$). The way objective g in a dependency relation $r_1 \succeq_g r_2$ is actually passed between r_1 and r_2 depends on the coordination type of the society, defined in the Architectural Templates. In OperA, three types of role dependencies are identified: *bidding*, *request* and *delegation*.

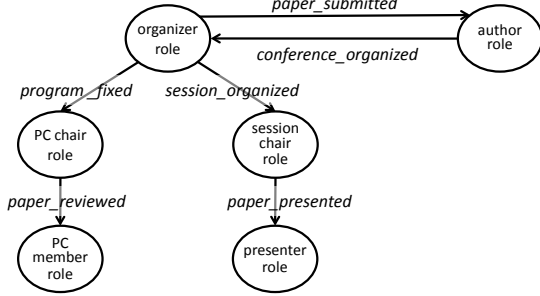


Fig. 1. Role dependencies in a conference.

B. The Interaction Structure.

Interaction is structured as a set of meaningful scenes that follow pre-defined abstract scene scripts. Examples of scenes are the registration of participants in a conference, which involves a representative of the organization and a potential participant, or paper review, involving program committee members and the PC chair. A *scene script* describes a scene by its players (roles), its desired results and the norms regulating the interaction. In the OM, scene scripts are specified according to the requirements of the society. The results of an interaction scene are achieved by the joint activity of the participating roles, through the realization of (sub-) objectives of those roles. A scene script establishes also the desired *interaction patterns* between roles, that is, a desired combination of the (sub-) objectives of the roles. Table II gives an example of a scene script.

Scene	Review Process
Roles	Program-Chair (1), PC-member(2..Max)
Results	$r_1 = \forall P \in \text{Papers: reviews_done}(P, rev1, rev2)$
Interact. Pattern	PATTERN(r_1): see figure 2
Norms & Rules	Program-Chair is PERMITTED to assign papers PC-member is OBLIGED to review papers assigned before deadline

TABLE II
SCRIPT FOR THE *Review Process* SCENE.

OperA interaction descriptions are declarative, indicating the global aims of the interaction rather than describing exact activities in details. Interaction objectives can be more or less restrictive, giving the agent enacting the role more or less freedom to decide how to achieve the role objectives and

interpret its norms. Following the ideas of [17], [14], we call such expressions *landmarks*, defined as conjunctions of logical expressions that are true in a state. Landmarks combined with a partial ordering to indicate the order in which the landmarks are to be achieved are called a *landmark pattern*. Figure 2 shows the landmark pattern for the *Review Process*. Several different specific actions can bring about the same

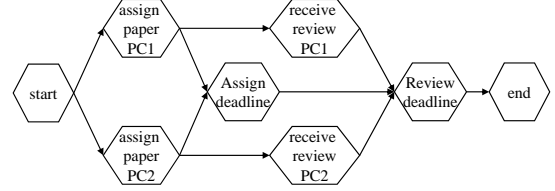


Fig. 2. Landmark pattern for *Review Process*.

state, that is, landmark patterns actually represent families of protocols. The use of landmarks to describe activity enables the actors to choose the best applicable actions, according to their own goals and capabilities. The relation between scenes is

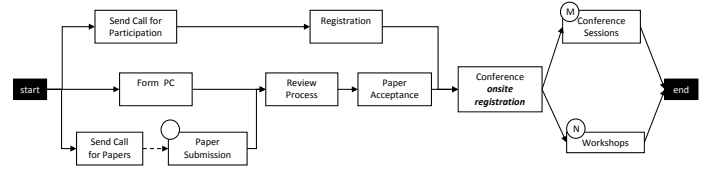


Fig. 3. Interaction Structure in the Conference scenario.

represented by the *Interaction Structure* (see figure 3). In this diagram, *transitions* describe a partial ordering of the scenes, plus eventual synchronization constraints. Note that several scenes can be happening at the same time and one agent can participate in different scenes simultaneously. Transitions also describe the conditions for the creation of a new instance of the scene, and specify the maximum number of scene instances that are allowed simultaneously. Furthermore, the enactment of a role in a scene may have consequences in following scenes. *Role evolution relations* describe the constraints that hold for the role-enacting agents as they move from scene to scene.

C. The Normative Structure.

At the highest level of abstraction, norms are the *values* of a society, in the sense that they define the concepts that are used to determine the value or utility of situations. For the conference organization scenario, the desire to share information and uphold scientific quality can be seen as organization values. However, values do not specify *how*, *when* or in *which* conditions individuals should behave appropriately in any given social setup.

In OperA, norms are specified in the Normative Structure using a deontic logic that is temporal, relativized (in terms of roles and groups) and conditional. For instance, the following norm might hold: “*The authors must submit their*

contributions before the deadline”, which can be formalized as: $O_{author}(submit(paper) \leq Deadline)$

Furthermore, in order to check norms and act on possible violations of the norms by the agents within an organization, abstract norms have to be translated into actions and concepts that can be handled within such organizations. To do so, the definition of the abstract norms are iteratively concretized into more concrete norms, and then translated into specific rules, violations and sanctions.

Concrete norms are related to abstract norms through a mapping function, based on the counts-as operator as developed in [1]. For example, in the context of Org , $submit(paper)$ can be concretized as: $send_mail(organizer, files) \vee send_post(organizer, hard_copies) \rightarrow_{Org} submit(paper)$

D. The Communication Structure.

Communication mechanisms include both the representation of domain knowledge (*what* are we talking about) and protocols for communication (*how* are we talking). Both content and protocol have different meanings at the different levels of abstraction (e.g. while at the abstract level one might talk of *disseminate*, such action will most probably not be available to agents acting at the implementation level). Specification of communication content is usually realized using ontologies, which are shared conceptualizations of the terms and predicates in a domain. Agent communication languages (ACLs) are the usual means in MAS to describe communicative actions. ACLs are wrapper languages in the sense that they abstract from the content of communication.

In OperA, the Communication Structure describes both the content and the language for communication. The content aspects of communication, or domain knowledge, are specified by *Domain Ontologies* and *Communication Acts* define the language for communication, including performatives and protocols.

IV. OPERETTA ENVIRONMENT

In order to support developers designing and maintaining organization models, tools are needed that provide an organization-oriented development environment. The requirements for such a development environment are the following.

- 1) **Organizational Design:** The tool should provide means for designing organizational models in an ‘intuitive’ manner. The tool should allow users to create and represent organizational structures, define the parties involved in an organization, represent organizational and role objectives, and define the pattern of interactions typically used to reach these objectives.
- 2) **Organizational Verification:** The tool should provide verification means and assistance in detecting faults in organizational designs as early as possible, to prevent context design issues from being translated to the other levels of system specification.
- 3) **Ontology Design:** The tool should be able to specify, import, and maintain domain ontologies. Domain ontologies specifying the knowledge for a specific domain

of interaction should be able to be represented, existing ontologies containing such information should be able to be included (and provide inputs for organizational concepts, such as role or objective names). Ontologies should be maintainable and updatable.

- 4) **Connectivity to System Level:** The output of the organizational design tool is intended for use by system level tools, namely MAS environments and agent programming languages. The output of the tool thus needs to provide easy integration and connection between the organization and system level.
- 5) **User-Friendly GUI:** A user-friendly graphical interface is to be provided for users to create and maintain organizational models easily. Help and guidelines are useful for beginners to use the tool.
- 6) **Availability:** The tool should be available under open source license and for use by other projects.

We have developed the Operetta development environment as an open-source solution on the basis of these requirements. Operetta enables the specification and verification of OperA OMs, which satisfies requirements 1 and 2. Operetta combines multiple editors into a single package. It provides separate editors on different components of organizational models; i.e., it has different (graphical) editors for each of the main components of an organizational model as defined in the OperA framework. These specialized editors correspond to the OperA OM structures: social, interaction, normative and communicative. The Operetta Ontology Manager enable the specification and import of domain ontologies, as in requirement 3.

The Operetta tool is a combination of tools based on the Eclipse Modeling Framework (EMF) [18] and tools based on the Graphical Modeling Framework (GMF) integrated into a single editor. Developed as an Eclipse plug-in, Operetta is fully open-source and follows the MDE principles of tool development. In the following we look at the editors provided by Operetta, and how Operetta connects to MAS solutions, thus satisfying requirement 4. A graphical interface (requirement 5) for Operetta has been developed and is currently being user-tested within the ALIVE project [12]. Finally, in accordance to the last requirement, Operetta is available opensource at sourceforge¹.

A. Operetta Components

The main element of Operetta is the OperA Meta-Model (see figure 4 for an overview of the tools in Operetta and their functionalities). The meta-model, created with the EMF tools, provides the (structural) definition of what organizational models should look like. This meta-model is extended with the default EMF edit and editor plug-ins to provide model accessors and the basic tree-based editor for the creation and management of OperA models. The basic editor has been extended with graphical interfaces for editing parts of the organization model: the social diagram editor, and the

¹<http://ict-alive.svn.sourceforge.net/>

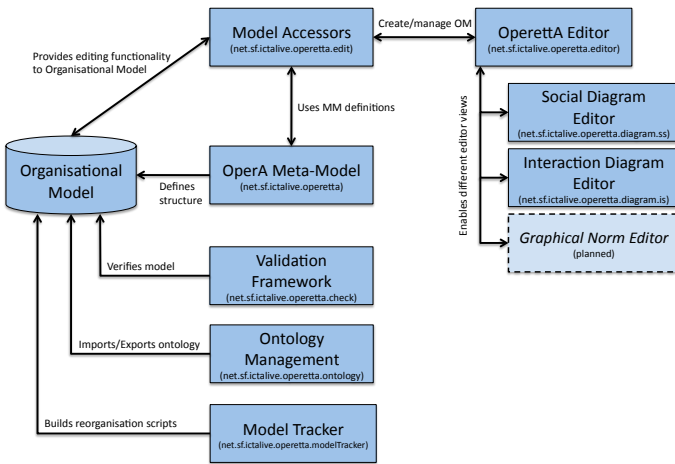


Fig. 4. Operetta Tool Components.

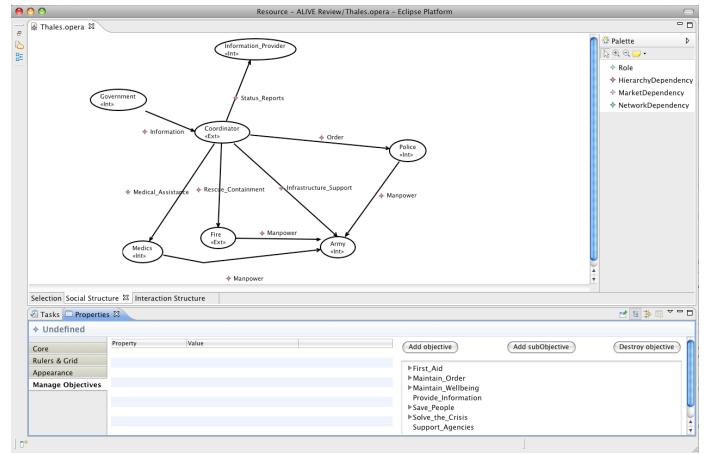


Fig. 5. Operetta Social Diagram Editor.

interaction diagram editor. A third graphical editor is planned for editing and managing formulas and norms.

Next to the graphical editing extensions, Operetta contains three other plug-ins for additional functionality. The Validation Framework provides an improvement over the default validation of EMF-based tools to provide validation of additional restrictions. An ontology managing plug-in is included as well to allow the ontology developed with Operetta to be exported to OWL, as well as allowing for importing existing ontology into the organization to boot-strap the organization design. Finally, Operetta contains a Model Tracker plug-in that can generate re-organization descriptions based on changes made in the Operetta organization editors.

We discuss the graphical editors and additional plug-ins in more details in the following.

1) *Social Diagram Editor*: This graphical editor provides a view of the Social Structure element of OMs. It allows the graphical creation of organizational Roles and Dependencies, thus specifying the social relations between important parties that play a part in the organization. The Social Diagram Editor also provides editing capabilities to specify and manage Role related Objectives, to provide context for the different Roles in an organization. Figure 5 depicts the Social Diagram Editor of Operetta that is used to enter organizational roles and dependencies between roles. Role objectives are created and managed via the objectives editor shown in the bottom part of the figure.

2) *Interaction Diagram Editor*: Similar to the Social Diagram Editor, the Interaction Diagram Editor provides a graphical view of the Interaction Structure element of OMs. This editor allows for the specification and management of the interaction elements of the organization; that is, it is for the specification and management of the different interactions that take place in the organization in order to achieve the different (role) objectives specified in the social part of the OM. The specification of the interaction is done in terms of scenes and transitions (the connection and synchronization

points between scenes). Together, these define the order in which objectives are to be reached and how the organization works (though specified on a high level of abstraction). The Interaction Diagram Editor allows for the graphical creation and maintenance of scenes, transitions and arcs (links between scenes and transitions). The graphical editor provides a user-friendly overview of the structural aspect of the organization, defining how different interactions within the organization are supposed to help achieve the organizational objectives. Finally, Operetta allows for the specification and editing of scene properties (like the scene results, the players active in the scene, the landmark pattern, etc).

3) *Ontology Manager*: The Ontology Manager part of the Operetta tool is a plug-in for importing and exporting (domain) ontologies. The creation and maintenance of ontologies is done by external tools (like, for example, Protégé). Parts of the functionality of organizational ontology editing is included in the Operetta editors:

- Automatic creation of organizational ontology while designing the organization. As the designer is inputting the organizational model in Operetta, Operetta maintains an ontology of role names, objective names, and logical atoms that the designer uses to define the organization.
- Using an included (existing) ontology for the naming of organizational model elements; that is, if an (external) ontology is present in the organizational model it can be used to pick concept names for different parts of an organizational model (e.g., the name of a role can be picked from an existing ontology included in the model). The addition of the (external) ontology to an organizational model is done via the ontology manager.

The functionality of ontology editing in the Operetta tools is limited to organizational ontologies. The Ontology manager plug-in extends Operetta with the following capabilities:

- Importing an ontology from a file (e.g., RDF or OWL [19]). Ontologies about the domain or organization that is to provide the context of a system might be already available. These ontologies tend to be stored in some

conventional ontology file-format. The Ontology Manager allows OperettA to import and use such ontologies.

- Exporting (generated) organizational ontologies to file. In order to align an use the organizational ontology created by OperettA, the Ontology Manager extends the OperettA tool with the capability to export the default ontology to an owl file.

The organizational ontology created by OperettA is stored in the Organizational Model. The ontological elements need to be available to the system level of design, and thus need to be included in the domain ontology. The integration of organizational concepts in a domain ontology is not trivial, as it should respect the structure of the domain ontology while adding organizational concepts as roles, objectives, etc. and the instances of these concepts; role names, objective names, etc. The alignment between the exported ontology and the domain ontology will have to be done by hand in an external editor. The inclusion of the ontology manager satisfies requirement 3.

4) *Model Tracker*: To support reorganization, OperettA is extended with a model tracker. This model tracker allows a designer to view the changes made on the organizational model since a last save (but not necessarily the previous one). By storing the changes to the organizational model in a history file, the model tracker can be used to generate scripts that express how an organization is changed. Reorganization scripts capture changes in a precise and concise manner, and can be used to communicate organizational changes to the system level.

5) *Validation Framework*: The validation plug-in of OperettA overwrites the basic validation provided by the EMF framework. Instead of just verifying constraints specified in the OperA meta-model, the validation has been extended with additional verification constraints to minimize organizational design mistakes. The overall purpose of the validation plug-in is to provide OM designers meaningful feedback to eliminate design errors as early as possible (in the design process). The validation plug-in is installed separately from OperettA, but after installation it can be invoked from within each of the different OperettA editing views. The validation plug-in seamlessly overwrites the standard EMF validation, making it the new default manner of validating OperettA models.

The validation plug-in works directly on the model instance to verify various modeling constraints, accessing the model via the meta-model definitions. Some examples of the constraints validated are checking that roles have a name, checking that role names are unique, checking that all roles have an objective, and so on. Less stringent constraints are checked as well, like, for example, whether roles are connected to other roles via dependencies; i.e., while it does not hold for every OM, in most models roles should be connected to other roles (that is, it should be depending upon (an)other role(s) or being depended upon by (an)other role(s)). Such “soft” constraints are presented to designer as a warning, intended to have the designer rethink their model and update if appropriate. The validation plugin fulfills requirement 3.

Property	Value
Activation Condition	Conjunction bid(item,price) ^ won(item)
Deadline	Atom one_week_after(auction)
Deontics	Role Deontic Statement O
Expiration Condition	Atom paid(price)
Maintenance Condition	Conjunction (~paid(price)) ^ (now < one_week_after(auction))
Norm ID	N1

Fig. 6. A Norm in OperettA.

6) *Norm Editor*: Norms are an important part of the organizational Model, providing lead ways on a high level of abstraction for the agents to follow. Norms can be inputted in the current version of OperettA via the basic EMF generated editor. This editor is not user-friendly. An example norm in OperettA is shown below in figure 6. The norm shown in this figure describes that buyers should have paid for the items they have won in an auction before one week has expired. The norm is input using several logical formulas; one for the activation condition expressing when the norm is active (the item is bid on and won), one for the expiration condition expressing that the norm is no longer active (the item has been paid for), one for the maintenance condition expressing the formula to be checked when the norm is active to see if violations have happened (the buyer has not paid and the week has not yet expired), and one for the deadline expressing a state of affairs before which the norm should have been fulfilled. A user-friendlier (graphical) interface for inputting and managing the norms of an organizational model is planned for a future version of OperettA. This extension, with the graphical editors for the social and interaction structures, fulfills requirement 5.

B. Connectivity to System Level

The OperettA tool has only off-line functionalities; it is used by designers to create the context of the system and their linked ontologies. It provides design and validation functionalities for the creation and management of OperA organizational models. For the connection to implementations OperettA depends on the Model Driven Engineering (MDE) approach by providing a meta-model of the modeling concepts.

MDE refers to the systematic use of models as primary artifacts throughout the Software Engineering lifecycle. The defining characteristics of MDE is the use of models to represent the important aspect of the system, be it requirements, high-level designs, user data structures, views, interoperability interfaces, test cases, or implementation-level artifacts such as code. The Model Driven Development promotes the automatic transformation of abstracted models into specific implementation technologies, by a series of predefined model transformations.

In essence, this means that the models created with the OperettA tool can be used for automated transformation towards applicable (models of) platforms; e.g. service-based implementations or multiagent systems. The only required step for such transformation is the definition of the transformations based on the OperettA meta-model concepts to the meta-model of the desired platform.

Finally, OperettA is based on the OperA formalism, which assumes that individual agents are designed independently from the organization, to model goals and capabilities of a given entity. Individual agents are the enactors of organizational role(s), as a means to realize their own goals [3]. As such it is necessary that OperettA can connect to such MAS frameworks. As a proof of concept, we have done experiments on the connection towards frameworks like Brahms and Repast, for the simulation of organizations in which normative properties of the organization can be verified for different populations with emergent behavior. As part of the ALIVE project [12] a connection was made with AgentScape to generate MAS from the organizational specification.

V. DESIGN GUIDELINES

In the previous we introduced organizational modeling and the OperettA Environment to support this. In this section we present a small overview on how one goes about designing an organization. After identifying that an organization presents the solution to the problem:

- 1) Identify (functional) requirements: First one determines the global functionalities and objectives of the society.
- 2) Identify stakeholders: The analysis of the objectives of the stakeholders identifies the operational roles in the society. These first two steps set the basis of the social structure of the OperA model.
- 3) Set social norms, define normative expectations: The analysis of the requirements and characteristics of the domain results in the specification of the normative characteristics of the society. This results in the norms in the normative structure.
- 4) Refine behavior: Using *means-end* and *contribution* analysis, a match can be made between what roles should provide and what roles can provide. This aspect contributes to refinement of role objectives and rights.
- 5) Create interaction scripts: Using the results from steps 3 and 4, one can now specify the patterns of interaction for the organization, resulting in the interaction structure.

More details about the methodological steps taken to create organizational models can be found in [7].

VI. CONCLUSIONS

In this paper, we present an organization-oriented modeling approach for system development. The OperA modeling framework can be used for different types of domains from closed to open environments and takes into consideration the differences between global and individual concerns. The OperettA tool supports software and services engineering based on the OperA modeling framework. It has been used in the European project ALIVE [12] that combines cutting edge coordination technology and organization models to provide flexible, high-level means to model the structure of interactions between services in an environment.

REFERENCES

- [1] Huib Aldewereld, Sergio Álvarez-Napagao, Frank Dignum, and Javier Vázquez-Salceda. Engineering social reality with inheritance relations. In *Proc. of the 10th Workshop Engineering Societies in the Agents' World (ESAW 2009)*. 2009.
- [2] L. Coutinho, J. Sichman, and O. Boissier. Modelling dimensions for agent organizations. In V. Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Information Science Reference, 2009.
- [3] M. Dastani, V. Dignum, and F. Dignum. Role assignment in open agent societies. In *AAMAS03*. ACM Press, July 2003.
- [4] V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. SIKS Dissertation Series 2004-1. Utrecht University, 2004. PhD Thesis.
- [5] V. Dignum. The role of organization in agent systems. In V. Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages ??-??. Information Science Reference, 2009.
- [6] V. Dignum and F. Dignum. Designing agent systems: State of the practice. *International Journal on Agent-Oriented Software Engineering*, 4(3), 2010.
- [7] V. Dignum, F. Dignum, and J.J. Meyer. An agent-mediated approach to the support of knowledge sharing in organizations. *Knowledge Engineering Review*, 19(2):147–174, 2004.
- [8] V. Dignum, J. Vazquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems: Second International Workshop ProMAS 2004*, volume 3346 of *LNAI*. Springer, 2005.
- [9] Virginia Dignum and Frank Dignum. Modeling agent societies: coordination frameworks and institutions. In A. Jorge P. Brazdil, editor, *Progress in Artificial Intelligence: Proc. of EPIA-2001*, LNAI 2258, pages 191–204. Springer, 2001.
- [10] M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In *ATAL-2001*, LNAI 2333, pages 348–366. Springer, 2001.
- [11] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *ICMAS'98*, pages 128–135. IEEE Computer Society, 1998.
- [12] European Commission FP7-215890. ALIVE, 2009. <http://www.ist-alive.eu/>.
- [13] Davide Grossi, Frank Dignum, Mehdi Dastani, and Lambèr Royakkers. Foundations of organizational structures in multiagent systems. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 690–697, New York, NY, USA, 2005. ACM.
- [14] S. Kumar, M. Huber, P. Cohen, and D. McGee. Towards a formalism for conversation protocols using joint intention theory. *Computational Intelligence Journal*, 18(2), 2002.
- [15] H.V.D. Parunak and J. Odell. Representing social structures in uml. In M. Wooldridge, G. Weiss, and P. Ciancarini, editors, *Agent-Oriented Software Engineering II*, LNCS 2222. Springer-Verlag, 2002.
- [16] L. Penserini, D. Grossi, F. Dignum, V. Dignum, and H. Aldewereld. Evaluating organizational configurations. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2009)*, 2009.
- [17] I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmbach. Designing conversation policies using joint intention theory. In *Proc. ICMAS-98*, pages 269–276. IEEE Press, 1998.
- [18] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework*. Eclipse Series. Addison-Wesley Professional, 2008.
- [19] W3C. Owl-s, 2004. <http://www.w3c.org/Submission/OWL-S>.
- [20] H. Weigand and V. Dignum. I am autonomous, you are autonomous. In M. Nickles, M. Rovatsos, and G. Weiss, editors, *Agents and Computational Autonomy*, volume 2969 of *LNCS*, pages 227–236. Springer, 2004.
- [21] F. Zambonelli. Abstractions and infrastructures for the design and development of mobile agent organizations. LNAI 2222, pages 245–262. Springer, 2002.
- [22] F. Zambonelli, N. Jennings, and M. Wooldridge. Organizational abstractions for the analysis and design of multi agent systems. LNAI 1957, pages 235–251. Springer, 2001.

A Dialogic Dimension for the $\mathcal{MOISE}+$ Organizational Model

A. Hübner, G.P. Dimuro and A.C.R. Costa

PPGMC, Centro de Ciências Computacionais

Universidade Federal do Rio Grande

96201-900 Rio Grande, RS, Brazil

Email: {ale.hubner, ac.rocha.costa,gracaliz}@gmail.com

V.L.D. Mattos

DEMAT, Instituto de Ciências Exatas

Universidade Federal Rural do Rio de Janeiro

32890-000 Seropédica, RJ, Brazil

Email: viviane.leite.mattos@gmail.com

Abstract—The aim of this work is to propose a fourth dimension for the $\mathcal{MOISE}+$ multiagent system organizational model, focused in the communication between roles. For that, the model $\mathcal{MOISE}+$ is extended with a dialogic dimension that defines the protocols used for communication between roles. In order to interlink this new dimension to the other dimensions of the $\mathcal{MOISE}+$ model, new relations are added to the deontic specification, which are responsible for indicating which protocols should or could be used to achieve the goals that constitute the roles' missions. The use of the extended model is illustrated, with a case study in the modeling of the process of creation of an episodic graduate course in a particular communitarian university in Brazil.

I. INTRODUCTION

It is possible to distinguish two structural levels in a Multiagent System (MAS), namely, the *organization* and the *population* structures. Basically, the population structure is composed by the agents themselves (and, of course, all the mechanism related to them, such as the ones for interaction and communication). The organization structure is related to the roles that may be played by the MAS population.

The PopOrg model [1], [2], [3], [4], for example, is a MAS organizational model that clearly separates those two structural levels.

On the other hand, in the PopOrg model, the notion of interaction between agents/roles is explained by means of *social exchanges* (i.e., exchange of services or objects [5], [6], [7]) between agents/roles in each structural level.

Since *communication* is one of the main tools that agents have to coordinate their actions in the exchanges that they perform at the population level (cf., e.g., [8]), the specification of the exchanges (and then, the communication) between roles may be a tool for the regulation of the exchanges between the agents that adopt such roles.

So, in the same way that the communication between agents is crucial to allow/regulate the exchanges that agents perform on their own, at the level of the MAS population, also the communication between roles can be a specification mechanism, at the organization level, to allow/regulate the exchanges between agents when they adopt those roles.

This becomes particularly important in some kinds of organizations where the exchange processes promoted in their

context demand an intensive communication flow between the organizational roles and/or groups of roles.

An example of such organization is the one of a university, considering, for example, its management processes, which are mainly coordinated by the communication between the roles (e.g, the president, a dean of the school, a head of department, a course coordinator, a professor, a department secretary), or between groups of roles (e.g., a department, a faculty, a scientific board, a school board, a research laboratory, a research group).

We have studied such management processes in a particular private communitarian university in Brazil, where it was observed those intensive communication flows guiding all such processes.

The MAS organizational model $\mathcal{MOISE}+$ [9], [10], [11], [12], [13], [14], [15], showed to be a practical organization model very suitable for our purposes, given in particular the very good set of tools that support it and help the design of MAS.

However, the $\mathcal{MOISE}+$ model presents only the following dimensions: the *structural* dimension (roles, role relations, inheritance relation, links, groups, etc.), the *funcional* dimension (global plans, missions, etc.), and the *deontic* dimension, which relates the other two dimension by stating the permissions and obligations of a role on a mission, thus lacking a dimension to specify communications between roles, which was essential in our application. Such lack of an explicit dialogic dimension in the $\mathcal{MOISE}+$ model, contrasted with the central position that dialogues play in our theoretical PopOrg model, motivated the present work.¹

Thus, the aim of this work is to propose the introduction of a *dialogic* dimension in the $\mathcal{MOISE}+$ model, focused in the communication between roles at the MAS organization level, not in the communication between agents at the MAS population level. As a case study, we took the process of creation of an episodic graduate course at that particular private communitarian university.

¹The definition of the $\mathcal{MOISE}+$ model was based on the \mathcal{MOISE} [16], [17] model, adding to it many facilities [9], [10], and offering a framework for the reorganization of multiagent systems [11]. Neither the original $\mathcal{MOISE}+$ nor in the $\mathcal{MOISE}+$ model, however, gave the status of a full dimension for the specification of the dialogues between roles.

The paper is organized as follows. Section II briefly presents basic concepts of MAS organization models, and, in particular, the *MOISE+* model, which is the one this paper is concerned with. Section III presents our proposal for introducing a dialogic dimension in the *MOISE+* model, allowing for the specification of the communication between roles. Section IV presents a case study related to the process of creation of an episodic graduate course at a particular private communitarian university in Brazil. Section V is the Conclusion.

II. MAS ORGANIZATIONAL MODELS AND *MOISE+*

In the literature, it is possible to find many approaches for MAS organization modelling [18]. Most of them offer a set of computational tools to support their use in the modeling of MAS. The development of these tools may consider either an *agent-centered* or a *system-centered* conception [9], [19], [20]. The former takes the agents as the engine for the organization formation, focusing the organizational agent-level deliberative mechanisms to interpret and reason about the specification of the organization. In the latter, the main concern is the organizational infrastructure, i.e., the organization exists a priori (defined by the designer or by the agents themselves) and the agents ought to follow it.

In general, the MAS organization models present a declarative language for the organization modelling and also an organization architecture (see, e.g., the Islander editor [21] and the Ameli agent-based platform [22] for electronic institutions).

The *MOISE+* model, which is an organization-centered model, also presents those facilities [20], [23]. In *MOISE+*, there is a language for specifying the MAS organization, which allows us to chose constraints and cooperation patterns to be imposed on the agents, in order to develop the *Organization Specification* (OS). An *Organization Entity* (OE) is then created as the agents adopt the roles specified in the organization specification, i.e., a set of agents builds an organization entity by adopting an appropriate organization specification in order to achieve its purpose.

The *MOISE+* model considers three organizational dimensions: the organization structure itself, its functions, and the deontic relation among them to explain how a MAS organization collaborates for its purpose:

- *Structural Dimension* (roles, groups, relations): A role is conceived as a set of behavioral constraints that an agent accepts since it joins a group in the organization. For example, in the case of the organization of a university, the agent that adopts the role of a *professor* has some kind of authority over the one that is playing the role of a *student*.
- *Functional Dimension* (goals, global plans, missions): It defines a set of global plans for the MAS to achieve its goals, which are structured in a social schema, as a goal decomposition tree, where each goal may be decomposed in sub-goals, and the responsibilities for the sub-goals are distributed in missions. The mission are attributed to the roles and constitute the commitments of the agents that adopt such roles. Once an agent is committed with a

mission, it is responsible to achieve the goals related to it. The mission may be attached to individual preferences, which are used in the case of establishing a preference order among missions.

- *Deontic Dimension* (obligations, permissions): It specifies the relations between the structural specification and the functional specification, establishing which missions each role is obliged or has the permission to realize.

The first two dimensions can be specified almost independently of each other, and, after, they are properly linked by the deontic dimension, which facilitates also the reorganization of the system.

A *MOISE+* organization specification is formed by a Structural Specification (SS), a Functional Specification (FS) and a Deontic Specification (DS). A *MOISE+* organization specification can be represented as a XML file, using an specific format, which can be manipulated by the *MOISE+* editor.

III. A DIALOGIC DIMENSION FOR THE *MOISE+* MODEL

As discussed in the Introduction, the specification of the interactions/exchanges between roles at the organization level may be an important tool for the regulation of the interactions/exchanges between agents that adopt those roles at the population level, and the communication is a fundamental tool that the roles/agents have in order to perform interactions/exchanges.

For example, in the GAIA methodology [24], whose underlying organization model allows to deal with adaptive multi-agent organizations [25], a role is defined by a set of four attributes: responsibility, permissions, activities and protocols. The protocols establishes the requirements for the interactions between roles (for example, to the role of *manager* may be associated the Contract Net protocol). Those protocols may be defined at the analysis phase. This association of protocols to roles generates an *interaction model*, which specifies the links between roles. Electronic institutions [21], [22] and the OperA model [26] are other organizational models where the specification of interactions also is a central feature.

On the other hand, the *MOISE+* organization model does not support a clear specification of how the interaction between roles may be conducted. In this paper, we propose the integration of a fourth dimension to the *MOISE+* model, namely the *dialogic* dimension, which allows for the specification of the communication between roles through protocols that should/may be used by them.

The idea of the inclusion of a dialogic dimension in the *MOISE+* model implies the addition of new relations in the deontic dimension, indicating which missions present goals that need communication between roles, and which protocols are required/permitted to be used while trying to achieve those goals.

The new organization configuration that we propose for the *MOISE+* model is shown in Fig. 1.

The protocols defined in the dialogic dimension are abstract, i.e., they do not specify the details of the communication

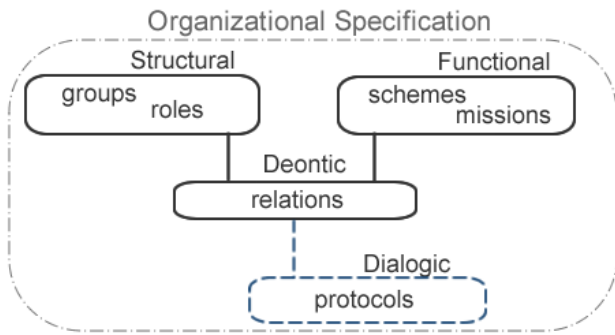


Fig. 1. The MOISE+ model extended with the Dialogic Dimension

operations to be used. The specification of how those abstract communication operations are to be realized by the communication primitives effectively available for the agents, when they adopt the roles involved in those communications, is defined separately, in a so-called *Dialogic Specification* (DLS), which is treated as a new separate part in the PopOrg specification, complementing the specification of how the population structure implements the organization structure.

The Deontic Specification of the MOISE+ is extended with the element `deontic-links`, which is responsible for defining which protocol is to be used by each role that has a goal whose achievement demands an interaction with another role (see XML Code 1; note that the `deontic-relation` element is original to the MOISE+ model).

XML Code 1. Communication in the deontic specification

```

<deontic-specification>
  <deontic-relation type="permission" role="role[x]"
    mission="m1" />
  <deontic-relation type="permission" role="role[y]"
    mission="m2" />
  <deontic-relation type="obligation" role="role[z]"
    mission="m3" />

  <deontic-links mission="m1" >
    <link type="obligation" goal="g1" protocol="p1"
      />
  </deontic-links>

  <deontic-links mission="m2" >
    <link type="obligation" goal="g2" protocol="p2"
      />
    <link type="permission" goal="g3" protocol="p3"
      />
  </deontic-links>
</deontic-specification>
  
```

A set of deontic-links like

```

<deontic-links mission="m2" >
  <link type="obligation" goal="g2" protocol="p2" />
  <link type="permission" goal="g3" protocol="p3" />
</deontic-links>
  
```

says that whenever a goal has the mission `m2`, it has the obligation of using protocol `p2` to achieve goal `g2` of `m2`, and the permission to use protocol `p3` to achieve goal `g3` of `m2`.

Although the communication protocols are defined abstractly in the dialogic specification, the parameters and per-

formatives of FIPA ACL [27] are used in order to structure the message in the communication specification, as can be seen in the XML Code 2 (a generic specification) and in XML Code 3 (an instantiated specification).

XML Code 2. A generic communication protocol

```

<dialogical-specification>
  <protocol-definitions>
    <protocol id="px" >
      <seq>
        <msg id="1" send="roleX" receiver="roleY"
          ">
          <content type="request" language="
            Prolog" says="requested(Request)"
            />
          <return reply-with="X" />
        </msg>
        <msg id="2" send="roleY" receiver="roleX"
          ">
          <content type="inform" language="
            Prolog" says="reply([X1 = V1, X2
            = V2, ... Xn = Vn])" />
          <return in-reply-to="X" />
        </msg>
      </seq>
    </protocol>
  </protocol-definitions>
</dialogical-specification>
  
```

XML Code 3. An instantiated communication protocol

```

<dialogical-specification>
  <protocol-definitions>
    <protocol id="p1" >
      <seq>
        <msg id="1" send="professor" receiver="
          student" >
          <content type="request" language="
            Prolog" says="?- location(you, (
            City,Country))" />
          <return reply-with="address" />
        </msg>
        <msg id="2" send="student" receiver="
          professor" >
          <content type="inform" language="
            Prolog" says="\+ City = pelotas,
            Country = brazil" />
          <return in-reply-to="address" />
        </msg>
      </seq>
    </protocol>
  </protocol-definitions>
</dialogical-specification>
  
```

In both codes, it is possible to observe the XML elements and attributes used in the implementation of a particular example of a dialogic specification:

- The element `<protocol>` has the attribute `id`, which is responsible for linking the dialogic specification with the deontic specification;
- The element `<msg>` may have from 2 to 4 attributes: `send/receiver` (indicates who send/receive the message), `propagate` (sends the message for a group), and `to` (indicates the final target of the message, when it is forwarded);

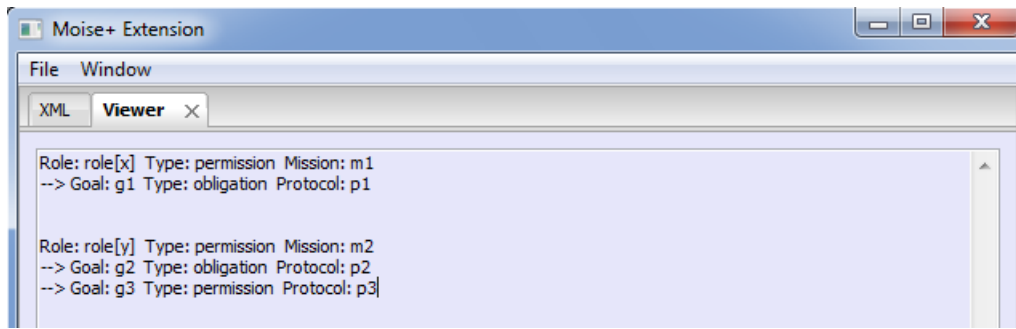


Fig. 2. The viewer tool

- The element `<content>` has 4 attributes: `type` (defines an interpretation for the message), `from` (indicated the first sender of the message), `says` (carries the content of the message), and `language` (specifies the language, which, in this case, is Prolog);
- The element `<return>` may have 1 or 2 attributes: `reply-with` (contains the identification label for an returning answer), and `in-reply-to` (contains the identification label of the received message).

Note that, for a particular application, a particular ontology for role communication would be specified.

In order to help the user, we implemented a viewer tool (Fig. 2), which joints the dialogic specification with de deontic specification, allowing to view, in a structured and organized way, all the protocols that the roles use to execute their duties.

IV. APPLICATION EXAMPLE

For the case study of this work, we selected one of the management processes that we found in the context of a particular private communitarian university in Brazil, namely, the management process of episodic graduate courses (the course that should occur just once), which can be divided into 4 stages: (i) creation, (ii) promotion and advertisement, (iii) classes and advising, and (iv) closing.

In the first stage, called the *creation* phase, which encompasses the conception and the formalization of the course, the role `professor` is the one who has the idea to propose the course.

Then, this *proponent professor* starts to collect related material, exchanging ideas with its colleagues (also with the role `professor`), and also talking with the role `dean of department` to which it proposes informally the creation of the course.

Observe that, at this phase, there is an intensive flow of communication between the group `faculty`, i.e., between the *proponent professor* and the other `professors`, and between the roles `professor` and `dean of department`.

The *proponent professor* also uses a lot of communication in order to ask for services and instructions, give and receive information/suggestions, to receive and discuss informal reports, etc., during the creation phase.

After an informal analysis if there is a good probability to have the course proposal approved in the higher management and scientific instances of the university, the *proponent professor* develops an schema of the course pedagogical project.

Then, the *dean of department* constitutes a group, the work team, which is composed by roles of `professors`. This work team is supposed to have meetings in order to elaborate the formal course pedagogical project.

After that, the *proponent professor* requests that the Control and Planning Consultancy to elaborate the financial analysis (costs, incomings) of the proposal. After that, the *proponent professor* formalizes its proposal, jointing the course pedagogical project with the respective financial analysis.

In the sequence, the *department secretary* opens a formal process, which is evaluated in the various management and scientific instances of the university, such as: Department Consultant Council, Graduate Board, Administration Board, and Superior Scientific Council.

After been approved in all those instances, the process goes to the second stage, which is the *promotion/advertisement* of the course.

If the course attracts a sufficient number of applications that guarantees that it will be economical viable, then it is finally approved, and it advances the other stages, namely, the *classes and advising*, and finally the *closing*.

In this paper, we show just the first stage of this process, namely, the creation process. After the conceptual modeling phase, where all the structure the university, related to this application, was depicted, identifying all the roles, groups of roles, relations, interactions between roles and between roles and groups, global plans, missions, etc., we developed the organization specification of a MAS for simulating the creation process, using the MOISE+ model.

Figures 3, and 4 show a sample of UML sequence diagrams, illustrating how the role communication protocols of the dialogic specification are visually designed.

After the visual design phase, the XML representation of the protocols are written. For example, the sequence diagram of Fig. 3 generates the protocol shown in the XML Code 4.

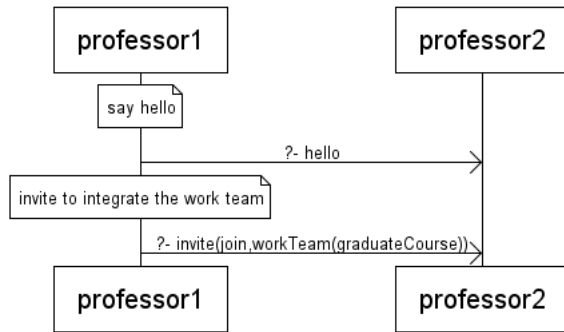


Fig. 3. Partial diagram of a role communication protocol

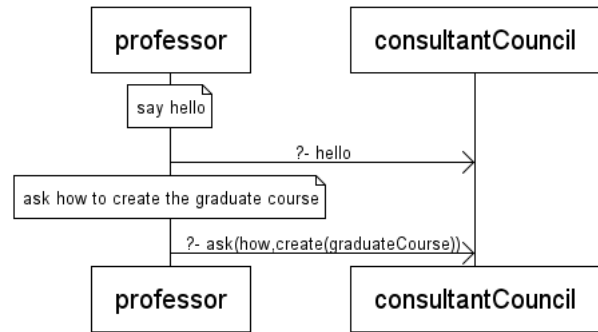


Fig. 4. Partial diagram of a role communication protocol

XML Code 4. Protocols of the Dialogic Specification

```

<dialogical-specification>
  <protocol-definitions>

    <protocol id="p1" >
      <seq>
        <msg id="1" send="professor1" receiver="
          professor2" group="faculty">
          <content type="inform" language="
            Prolog" says="?- hello" />
        </msg>
        <msg id="2" send="professor1" receiver="
          professor2" group="faculty">
          <content type="cfp" language="Prolog"
            says="?- join(workTeam(
              graduateCourse))" />
        </msg>
      </seq>
    </protocol>

  </protocol-definitions>
</dialogical-specification>

```

Figure 5 shows the deontic dimension, with the dialogical elements that were added for the specification of the protocols to be used in the interactions between roles in the creation phase of the management process of episodic graduate courses.

V. CONCLUSION

It is possible to find in the literature several organizational models for the modeling of multiagent systems. This work was concerned, in particular, with the MOISE+ model. The MOISE+ model is an improvement over the MOISE model that allowed its use in different contexts when modeling MAS systems. However, some elements were not considered in MOISE+ model, such as the specification of communication protocols.

In this paper, we discussed the importance, in some specific applications, of having tools for the specification of the interactions/exchanges that use communication between roles at the MAS organization level, which may help the regulation of the interactions/exchanges that use communication between the agents that adopt those roles at the MAS population level.

This work proposed an extension to the MOISE+ organizational model, which incorporated a dialogic dimension used to specify the communication between roles, where the protocols applied in the role communication are defined.

The dialogic dimension was connected to the deontic dimension by the adding new relations that are responsible for indicating which missions have goals that need role communication, specifying permission and obligations to use communication protocols. The dialogic dimension was modeled with the specification of the protocols using the XML language.

We developed an application related to the creation phase of the management process of an episodic graduate course in a particular private communitarian university. This case study was particularly interesting for the purpose of validating our proposal, since that this kind of organization and its management processes presented a large communication flow between the roles.

ACKNOWLEDGMENT

This work is part of a larger project (RS-SOC: Rede Estadual de Simulação Social), being run under the FAPERGS/CNPq/PRONEX context, where the political aspects of dialogical features of social organizations are being investigated. The work is supported by FAPERGS/CNPq/PRONEX (Proc. 10/0049-7) and CNPq (Proc. 483257/09-5, 307185/07-9, 304580/07-4). We thank Jomi Hübner for helping us with the MOISE+ tools and also for his valuable suggestions.

REFERENCES

- [1] A. C. da Rocha Costa and G. P. Dimuro, "Semantical concepts for a formal structural dynamics of situated multiagent systems," in *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, ser. LNAI, J. Sichman, P. Noriega, J. Padget, and S. Ossowski, Eds. Berlin: Springer, 2008, no. 4870, pp. 139–154.
- [2] —, "A basis for an exchange value-based operational notion of morality for multiagent systems," in *Progress in Artificial Intelligence, 13th Portuguese Conf. on Artificial Intelligence, EPIA 2007*, ser. LNAI, J. Neves, M. Santos, and J. Machado, Eds. Berlin: Springer, 2007, no. 4874, pp. 580–592.
- [3] —, "A minimal dynamical organization model," in *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, V. Dignum, Ed. Hershey: IGI Global, 2009, pp. 419–445.
- [4] —, "Introducing social groups and group exchanges in the PopOrg model," in *Proceedings of AAMAS 2009*, vol. 1. Budapest: IFAAMAS, 2009, pp. 1297–1298.
- [5] J. Piaget, *Sociological Studies*. London: Routledge, 1995.
- [6] G. Homans, *Social Behavior – Its Elementary Forms*. New York: Harcourt, Brace & World, 1961.
- [7] P. Blau, *Exchange & Power in Social Life*. New Brunswick: Trans. Publish., 2005.

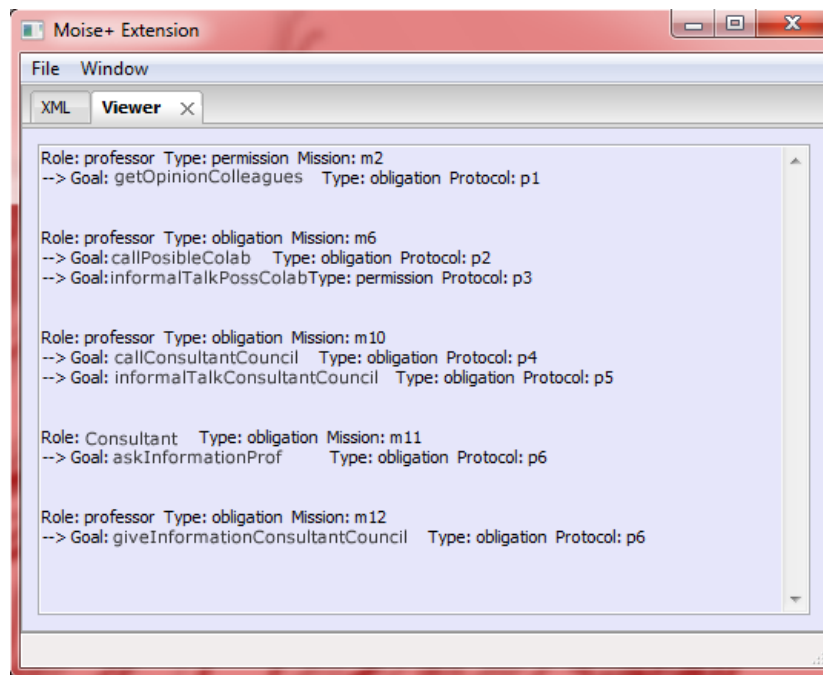


Fig. 5. The Extended Deontic Specification

- [8] M. Wooldridge, *An Introduction to MultiAgent Systems*. Chichester: Wiley, 2002.
- [9] J. F. Hübner, J. S. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in multiagent systems," in *Advances in Artificial Intelligence, Proceedings of the 16th Brazilian Symposium on Artificial Intelligence*, ser. LNCS, G. Bittencourt and G. Ramalho, Eds., vol. 2507. Springer, 2002, pp. 118–128.
- [10] —, "MOISE+: towards a structural, functional, and deontic model for MAS organization," in *Proc. of the First Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*. New York: ACM, 2002, pp. 501–502.
- [11] J. F. Hübner, O. Boissier, and J. S. Sichman, "Programming MAS reorganisation with MOISE+," in *Foundations and Practice of Programming Multi-Agent Systems*, ser. Dagstuhl Seminars Proc., J. Meyer, M. Dastani, and R. Bordini, Eds., no. 06261. IFBI, 2006.
- [12] J. F. Hübner, J. S. Sichman, and O. Boissier, "Developing organised multi-agent systems using the MOISE+ model: programming issues at the system and agent levels," in *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 3/4, pp. 370–395, 2007.
- [13] J. F. Hübner, R. H. Bordini, and G. Picard, "Jason and MOISE+: Organisational programming in the agent contest 2008," in *Dagstuhl Seminar on Programming Multi-Agent Systems*, ser. Dagstuhl Seminars Proc., R. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, Eds., no. 08361, 2008.
- [14] J. F. Hübner, "Moise+ framework and ora4mas," in *Proceedings of Lorentz Workshop on Rich Cognitive Models for Policy Design and Simulation*, W. Jager, C. Jonker, and V. Dignum, Eds., Leiden, 2009.
- [15] J. F. Hübner, O. Boissier, and J. S. Sichman, "Using jason and MOISE+ to develop a team of cowboys," in *Programming Multi-Agent Systems*, ser. LNAI, J. Meyer, M. Dastani, and R. Bordini, Eds. Berlin: Springer, 2006, no. 5442, pp. 238–242.
- [16] M. Hannoun, O. Boissier, J. S. Sichman, and C. Sayettatino, "MOISE: An organizational model for multi-agent systems," in *Proc. of Intl. Joint Conferences, 7th Ibero-American Conf. on Artificial Intelligence (IBERAMIA'00) and 15th. Braz. Symp. on Artificial Intelligence (SBIA'00)*, ser. LNAI, vol. 1952. Berlin: Springer, 2000, pp. 152–161.
- [17] M. Hannoun, O. Boissier, J. S. Sichman, and C. Sayettat, "Moise: An organizational model for multi-agent systems," in *Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI, IBERAMIA/SBIA 2000*, ser. LNAI, M. C. Monard and J. S. Sichman, Eds. Berlin: Springer, 2000, no. 1952, pp. 152–161.
- [18] V. Dignum, Ed., *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Hershey: IGI Global, 2009.
- [19] C. Lemaytre and C. B. Excelente, "Multi-agent organization approach," in *Proceedings of II Iberoamerican Workshop on DAI and MAS*, F. J. Garijo and C. Lemaytre, Eds., Toledo, 1998.
- [20] J. F. Hübner, J. S. Sichman, and O. Boissier, "Developing organised multi-agent systems using the moise+ model: Programming issues at the system and agent levels," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 3-4, pp. 370–395, 2007.
- [21] M. Esteva, D. de la Cruz, and C. Sierra, "ISLANDER: An electronic institutions editor," in *Proc. First international joint conference on autonomous agents and multiAgent systems, AAMAS 2002*, ser. LNAI, C. Castelfranchi and W. L. Johnson, Eds. Berlin: Springer, 2002, no. 1191, pp. 1045–1052.
- [22] M. Esteva, Rodríguez-Aguilar, J. A., B. Rosell, and J. L. Arcos, "AMELI: An agent-based middleware for electronic institutions," in *Proc. Third international joint conference on autonomous agents and multi-agent systems, AAMAS 2004*, N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, Eds. New York: ACM, 2004, pp. 236–243.
- [23] J. F. Hübner, J. S. Sichman, and O. Boissier, "S-MOISE+: A middleware for developing organised multi-agent systems," in *Coordination, organizations, institutions, and norms in multi-agent systems*, ser. LNAI, O. Boissier, V. Dignum, E. Matson, and J. S. Sichman, Eds. Berlin: Springer, 2006, no. 3913, pp. 64–78.
- [24] M. Wooldridge, N. R. Jennings, and D. Kinny, "The gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.
- [25] L. Cernuzzi and F. Zambonelli, "Dealing with adaptive multi-agent organizations in the gaia methodology," in *Proceedings of AOSE 2005*, 2005, pp. 217–228.
- [26] V. Dignum, "A model for organizational interaction: based on agents, founded in logic," Ph.D. dissertation, University of Utrecht, Utrecht, 2003.
- [27] Foundation for Intelligent Physical Agents, *FIPA 97 Specification Part 2: Agent Communication Language*, Oct. 1997, version 2.0. [Online]. Available: <http://www.fipa.org/spec/f8a22.zip>

From Signed Information to Belief in Multi-Agent Systems

Laurent Perrussel
IRIT – Université de Toulouse
laurent.perrussel@irit.fr

Emiliano Lorini
IRIT – Université de Toulouse
emiliano.lorini@irit.fr

Jean-Marc Thévenin
IRIT – Université de Toulouse
jean-marc.thevenin@irit.fr

Abstract—The aim of this paper is to propose a logical framework for reasoning about signed information. That is, as long as agents receive information in a multi-agent system, they keep track of the information source. The main advantage is that by considering a reliability relation over the sources of information, agents can justify their own current belief state. Agents believe at first information issued from the most reliable sources. Keeping track of belief’s origin also enables agents to improve communication by asking and gaining details about exchanged information. This is a key issue in trust handling and improvement: an agent believes some statement because it may justify the statement’s origin and its reliability.

I. INTRODUCTION

An agent embedded in a multi-agent system gets information from multiple origins; it captures information from its own sensors or, through some communication channels it may receive messages issued by other agents. Based on this set of basic information the agent then defines its beliefs and performs actions [1]. As long as it gets information, the agent has to decide what it should believe and also which beliefs are dropped [2], [3]. In order to decide which beliefs should hold, the agent needs some criteria. A common criterion consists of handling a reliability relation on its beliefs w.r.t. their origins [4], [5]. According to its opinion about the reliability of the information source, the agent decides to adopt or not the received piece of information. By keeping track of information and its origin, agents can justify their beliefs: agent a believes φ because agent b has provided φ and b is reliable [6]. This explicit representation helps agents to enrich their dialogs: they cannot only provide information but they may also mention the third party at the origin of information. Let us consider again agent a and information provided by b : a may then ask b the underlying source of φ and a may then ask to this source. Hence, this issue is a key one for trust characterization: keeping track of agents involved in information broadcasting enables agents to evaluate, from their own point of view, whether they are all reliable, i.e. believable [7].

The aim of this paper is to propose a modal framework for representing agent’s belief state and its dynamics by considering signed information, that is information associated to its source. If many work has been made in order to show how an agent can merge information issued from multiple origins [8], [9], very few work has focused on the explicit representation of the origins of information [10], [6] in the context of BDI-based systems with communication actions.

But we advocate that this explicit representation is necessary since it represents the underlying rationale of agents’ beliefs.

The dynamics is usually described in terms of performative actions based on KQML performatives [11] or speech acts [12], [13]. Hereafter, we propose to consider *tell* actions as private announcements from an agent (the sender of the message) to another agent (the receiver of the message). Private announcements enable to stress up how agents “restrict” their belief state as they receive information. More precisely, they shrink the space of information with their origins and then according to that space, they build up their beliefs.

The paper is structured as follows: In section II, we present the intuitive meaning of signed information and belief state. Next in section III, we present the technical details of the modal logic framework. In section IV, we then represent an intuitive and common policy for relating signed information and belief which consists in the adoption as belief of all consistent information. Next, in section V, we extend the logical system with actions of the form “agent a tells to agent b that a certain fact p is true”. We conclude the paper in section VI by summing up the contribution and considering some open issues.

II. SETTING THE FRAMEWORK

Handling the source of information leads to the notion of *signed statement*, that is some statement is true according to some source. From a semantics perspective, we want to be able to represent, w.r.t. some initial state of affairs, for each agent, what are the possible states that can be signed by each source. Agents build their own belief state using information signed by each source and the reliability of the source.

Example 1 *Suppose a car accident involving three cars which are blue (bc), red (rc) and yellow (yc). Now suppose a police detective who is interviewing the witnesses of the accident. Let po be the police detective. The first witness w_1 tells to the police detective that the blue car is responsible of the accident while the second one (w_2) states that the red car has caused the collision. Both of them tell to the police detective that yc is not responsible of the accident. In that context of information gathering, the police detective does not need to assume that the witnesses tell the truth or believe in information they provide. The police detective just needs to assume that w_1 provides or signs information $bc \wedge \neg rc \wedge \neg yc$ and w_2 provides or*

signs information $\neg bc \wedge rc \wedge \neg yc$. Next, based on these pieces of information, the police detective will build his opinion, i.e. his belief about the accident. The police detective faces contradicting information about the blue and red cars, but because the witnesses both agree about the yellow one, the police detective should believe that the yellow car is not the responsible of the collision. That is, the detective is willing to root his belief upon the set of signed statements he handles.

A. Representing signed statements

Signed statements can be represented through Kripke models using one accessibility relation per source of information. Let $\text{Sign}(b, p)$ be a modal operator stating that statement p is true according to source b . $\text{Sign}(b, p)$ is true in state w if p holds in all states reachable from w through a relation denoted S_b describing the possible information states issued from b .

Example 2 Let us consider the initial example. Information which might be signed by the two witnesses are bc and rc which leads to the signed statements $\text{Sign}(w_1, bc)$, $\text{Sign}(w_1, rc)$, $\text{Sign}(w_1, bc \wedge rc)$,... With respect to our example, hereafter we will focus on the two signed statements $\text{Sign}(w_1, bc \wedge \neg rc \wedge \neg yc)$ and $\text{Sign}(w_2, \neg bc \wedge rc \wedge \neg yc)$.

B. Interpreting signed statements

The aim is to represent formulas such as $\text{Bel}(a, \text{Sign}(b, \varphi_0))$ or, in a more general way $\text{Bel}(a, \varphi_0)$, which respectively stands for agent a believes that agent b signs φ_0 and agent a believes φ_0 . As for signatures, we use an accessibility relation denoted B_a to represent the possible belief states of agent a .

We assume that signed statements represent the rationales for beliefs. That is, if agent a believes φ_0 it is because some signed statement $\text{Sign}(b, \varphi_0)$ holds in every possible belief state of agent a and agent a is willing to commit to this signed statement. Let a , b and c be three agents and p be a propositional symbol; Figure 1 illustrates the possible belief states of agent a w.r.t. some initial state w_0 using accessibility relation B_a (if p holds in a state, p is mentioned between brackets). Agent a considers two possible belief states, w_1 and w_2 . In state w_1 , the two possible states given by S_b contain p which entails that p is signed by b . On the other hand, the two possible states given by S_c contain p and $\neg p$: no information can be signed by agent c . In all states related to w_2 with S_b and S_c , p is true. From this figure we can conclude that in state w_0

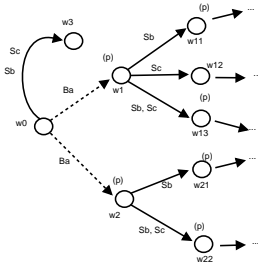


Fig. 1: Relating belief state and signatures

agent a believes that p is signed by b that is $\text{Bel}(a, \text{Sign}(b, p))$ while it does not believe that p or $\neg p$ is signed by c . Since p is signed by b and agent c says nothing about p , agent a should believe p : $\text{Bel}(a, p)$. Hence, it follows that in order to prevent adoption of inconsistent statements, hereafter we will assume that signed statements are always consistent (and thus relation S_a is serial).

Notice that the way we consider the link between beliefs and signed statements differs from the way this link is defined in [6]. That is, signed states are considered from each belief state while C. Liau [6] considers informational states and belief states in an independent way. This is due to the fact that informational states in [6] reflect communication actions while our notion of signed statement is more considered as an epistemic notion.

Example 3 Let us pursue our motivating example. As mentioned, we assume that the detective is willing to adopt as belief statements signed by the witnesses: $\text{Bel}(p_0, \text{Sign}(w_1, bc \wedge \neg rc \wedge \neg yc))$ and $\text{Bel}(p_0, \text{Sign}(w_2, \neg bc \wedge rc \wedge \neg yc))$. Since both witnesses agree on $\neg yc$, agent p_0 also adopts as belief $\neg yc$. Meanwhile, he cannot set his belief about the two other cars since p_0 faces contradicting signed statements.

C. Preferences over information sources

In order to know how to handle mutually inconsistent signed statements, agents consider extra information stating which signed statement they prefer. Agents may determine themselves their preferences by considering the sources of information [4], [9], temporal aspects or the topics of the statements [14].

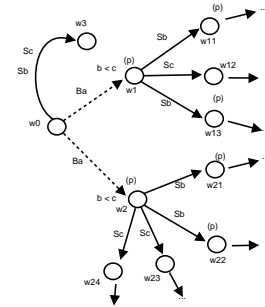


Fig. 2: Contradicting signed statements

In this paper, for the sake of conciseness and following numerous contributions such as [5], we propose to consider extra information about the reliability of sources of information as illustrated by Figure 2. That is, we assume that the agents consider information about only one topic. Consequently, handling competencies or different kinds of reliability (such as suggested in [15]) is out of the scope of the paper.

That is, if agent a believes that b is more reliable than c , then agent a adopts statement p as a belief even if agent c has signed $\neg p$. Suppose that reliability is represented with the

help of a pre-order relation \leq (or $<$): $a \leq b$ stands for a is at least as reliable as b . In semi formal terms, we get that:

$$\text{Bel}(a, (\text{Sign}(b, p) \wedge \text{Sign}(c, \neg p) \wedge b < c)) \Rightarrow \text{Bel}(a, p)$$

It follows that in each state, we do not only consider the value of propositional symbols but also a pre-order relation which characterizes a reliability order over information sources. Using extra-information on reliability and by considering signed statements rather than statements, the problem of belief change [2] is almost rephrased in terms close to the ones used in belief merging [16], [17]. Reliability order over sources of information enables us to stratify signed information and then by merging this stratified information in a consistent way the agents get “justified” beliefs [18].

Example 4 *Let us go on with our motivating example. Suppose agent po considers that the first witness is at least as reliable as the second one and he is himself willing to adopt as belief the signed statements issued by the two witnesses, i.e. we have the following belief:*

$$\text{Bel}(po, w_1 \leq w_2 \leq po)$$

Hence, according to the previous semi formal axiom schema previously given, the police detective should believe that the blue car (bc) has caused the accident. Notice that the willingness attitude is translated in terms of preferences: po has no opinion and considers as more important information provided by w_1 and w_2 .

D. Representing tell statements

Dynamics is viewed as restriction on agents’ belief states. We interpret the *tell* performative as a private announcement [19] rather than with help of actions and transitions between states. A private announcement consists of an information flow from one agent to a second one with a propositional statement as content. Figure 3 illustrates how agent a ’s belief state changes after agent c tells p . According to this example, after the performative $Tell(c, a, p)$, agent a has restricted its possible belief states to the states in which c signs p . In the initial situation (the left part of the figure), at w_0 , agent a believes $\text{Sign}(b, p)$, does not believe $\text{Sign}(c, p)$ and does not believe p (since p does not hold in w_2). After receiving agent c ’s message (right part of the figure), states where p is not signed by c are no longer possible states for agent a and thus, at w_0 , agent a believes $\text{Sign}(b, p)$, $\text{Sign}(c, p)$ and finally also believes p . That is, the performative $Tell(c, a, p)$ (agent c tells to agent a that p is true) is responsible for updating a ’s beliefs in such a way that a believes that c signs p . In other words, private announcements stress up the information gathering aspect: possible worlds accessible through relation B_a represent the ignorance of agent B_a and by shrinking this set of possible believable worlds, we represent how agent a gains information. Let us stress that this way of handling the dynamics entails as a drawback that agent’s belief cannot always be consistent: updating a model might lead to a model where seriality cannot be guaranteed.

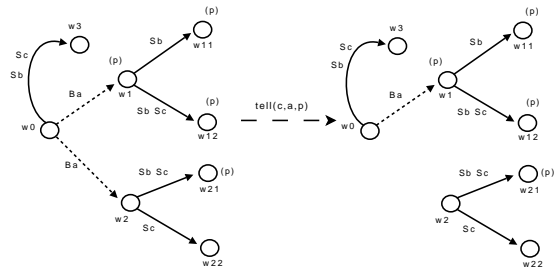


Fig. 3: Agent c tells p to agent a

Example 5 *In the context of our motivating example, the dynamics is represented by the sequence of interviews. For instance, agent po interviews at first w_1 , action represented by $Tell(w_1, po, bc \wedge \neg rc \wedge \neg yc)$ and then interviews the second witness ($Tell(w_2, po, \neg bc \wedge rc \wedge \neg yc)$). After these two announcements, the detective believes: $\text{Bel}(po, \text{Sign}(w_1, bc \wedge \neg rc \wedge \neg yc))$ and $\text{Bel}(po, \text{Sign}(\neg bc \wedge rc \wedge \neg yc))$.*

III. FORMAL FRAMEWORK

The proposed language for reasoning about signatures, beliefs and preferences is a restricted first order language which enables quantification over agent ids. In this section, we focus on these three notions, *tell* actions will be introduced later. Quantification allows agents to reason about anonymous signatures. For the sake of conciseness, we restrict signed statements to propositional statements. Let \mathcal{L}_0 be the propositional language built over a set of propositional symbols \mathcal{P} and \mathcal{L} be the logical language. Language \mathcal{L} is based on doxastic logic. Modal operator Bel represents beliefs: $\text{Bel}(a, \varphi)$ means agent a believes \mathcal{L} -formula φ . Modal operator Sign represents signed statements: $\text{Sign}(t, \varphi_0)$ means t (an agent id or a variable of the agent sort) signs propositional statement φ_0 . In order to represent agent’s opinion about reliability, we introduce the notation $a \leq b$ which stands for: agent a is said to be at least as reliable as b .

Definition 1 (Syntax of \mathcal{L}) *Let \mathcal{P} be a finite set of propositional symbols. Let \mathcal{A} be a finite set of agent ids. Let \mathcal{V} be a set of variables s.t. $\mathcal{A} \cap \mathcal{V} = \emptyset$. Let $\mathcal{T} = \mathcal{A} \cup \mathcal{V}$ be the set of agent terms. The set of formulas of the language \mathcal{L} is defined by the following BNF:*

$$\varphi ::= p \mid \neg \varphi \mid \varphi \wedge \varphi \mid \text{Sign}(t, \varphi_0) \mid \text{Bel}(a, \varphi) \mid \forall x \varphi \mid t \leq t'$$

where $p \in \mathcal{P}$, $t \in \mathcal{T}$, $\varphi_0 \in \mathcal{L}_0$, $a \in \mathcal{A}$ and $x \in \mathcal{V}$.

Writing $a < b$ stands for a is strictly more reliable than b : $a \leq b \wedge \neg(b \leq a)$. Writing $a \sim b$ means that a and b are equally reliable. Operators \rightarrow and \exists are used according to their usual meaning.

A. Semantics

The semantics of \mathcal{L} -formulas is defined in terms of possible states and relations between states [20]. Those relations respectively represent the notion of signatures and beliefs. In

each state, propositional symbols are interpreted and total pre-orders representing agents' reliability are set.

Definition 2 (Model) Let M be a model defined as a tuple:

$$\langle W, \bigcup_{i \in A} S_i, \bigcup_{i \in A} B_i, I, \preceq \rangle$$

where W is a set of possible states. $S_i \in W \times W$ is an accessibility relation representing signatures, $B_i \in W \times W$ is an accessibility relation representing beliefs. I is an interpretation function of the propositional symbols w.r.t. each possible state, $I : W \times \mathcal{P} \mapsto \{0, 1\}$. \preceq is a function which represents total pre-orders; these pre-orders are specific to each state, that is $\preceq : W \mapsto 2^{A \times A}$.

A variable assignment is a function v which maps every variable x to an agent id. A t -alternative v' of v is a variable assignment similar to v for every variable except t . For $t \in \mathcal{T}$, $\llbracket t \rrbracket_v$ belongs to A and refers to the assignment of agent terms w.r.t. variable assignment v , such that:

$$\text{if } t \in A \text{ then } \llbracket t \rrbracket_v = t \quad \text{if } t \in \mathcal{V} \text{ then } \llbracket t \rrbracket_v = v(t)$$

We define the satisfaction relation \models with respect to some model M , state w and variable assignment v as follows.

Definition 3 (\models) Let M be a model and v be a variable assignment: $v : \mathcal{V} \rightarrow A$. M satisfies an \mathcal{L} -formula φ w.r.t. a variable assignment v and a state w , according to the following rules:

- $M, v, w \models t \leq t'$ iff $(\llbracket t \rrbracket_v, \llbracket t' \rrbracket_v) \in \preceq(w)$.
- $M, v, w \models p$ iff $p \in \mathcal{P}$ and $I(w, p) = 1$.
- $M, v, w \models \text{Sign}(t, \varphi_0)$ iff $M, v, w' \models \varphi_0$ for all w' s.t. $(w, w') \in S_{\llbracket t \rrbracket_v}$.
- $M, v, w \models \text{Bel}(a, \varphi)$ iff $M, v, w' \models \varphi$ for all w' s.t. $(w, w') \in B_a$.
- $M, v, w \models \forall t \varphi$ iff for every t -alternative v' , $M, v', w \models \varphi$.

We write $\models \varphi$ iff for all M , w and v , we have $M, v, w \models \varphi$. The semantics for operators \neg , \rightarrow , \vee , \wedge and \exists is defined in the standard way. Let us now detail the constraints that should operate on the model. We only require that signature has to be consistent which entails that all relations S_i have to be serial. Belief operator is a $K45$ operator and thus all B_i are transitive and euclidian. Interwoven relations between signatures and beliefs are detailed in the next section.

1) *Constraining the Reliability Relations:* We assume that every agent holds belief about reliability without any uncertainty. That is, agent's beliefs about reliability can be represented as a total pre-order. However, it does not mean that we consider a fixed notion of reliability: we propose to handle multiple pre-orders by indexing reliability with worlds. That is, in each possible world or believable world, an agent considers how it ranks the agents. In that context, each rank is considered as a possible rank and thus it is natural that each of them should be total. However, we enforce a stronger notion

- (K_S) $\text{Sign}(a, \varphi_0 \rightarrow \psi_0) \rightarrow (\text{Sign}(a, \varphi_0) \rightarrow \text{Sign}(a, \psi_0))$
- (D_S) $\text{Sign}(a, \varphi_0) \rightarrow \neg \text{Sign}(a, \neg \varphi_0)$
- (K_B) $\text{Bel}(a, \varphi \rightarrow \psi) \rightarrow (\text{Bel}(a, \varphi) \rightarrow \text{Bel}(a, \psi))$
- (4_B) $\text{Bel}(a, \varphi) \rightarrow \text{Bel}(a, \text{Bel}(a, \varphi))$
- (5_B) $\neg \text{Bel}(a, \varphi) \rightarrow \text{Bel}(a, \neg \text{Bel}(a, \varphi))$
- (R_{\leq}) $t \leq t$
- (Tr_{\leq}) $t \leq t' \wedge t' \leq t'' \rightarrow t \leq t''$
- (T_{\leq}) $t \leq t' \vee t' \leq t$
- (To_{\leq}) $\text{Bel}(a, t \leq t') \vee \text{Bel}(a, t' \leq t)$
- (MP) From φ and $\varphi \rightarrow \psi$ infer ψ
- (G) From φ infer $\forall t \varphi$
- (N_S) From φ_0 infer $\text{Sign}(t, \varphi_0)$
- (N_B) From φ infer $\text{Bel}(a, \varphi)$

TABLE I: Logic \mathcal{L} axioms and inference rules

of totality which states that the aggregation of all believable ranks over agents (which are total) leads to a total preorder. This will then help the agent to integrate all signed statements. In other words, we require that the integration (or merging) of signed statements should be based on an underlying total preorder over statements (as it is commonly assumed in the belief revision and merging areas—see [2], [21], [17]). In terms of constraints on states and relations between them, it means that:

- 1) for all states w , $t \preceq(w) t'$ or $t' \preceq(w) t$ and,
- 2) suppose $w B_i w'$ and $t \preceq(w') t'$, then for all states w'' s.t. $w B_i w''$, $t \preceq(w'') t'$.

The first constraint enforces that pre-orders are total in all states; the second constraint expresses that totality should hold in all belief states. Moreover, preorder definition entails that reflexivity and transitivity hold.

B. Axiomatics

Let us now translate these constraints in terms of proof theory. Axiomatization of logic \mathcal{L} includes all tautologies of propositional calculus. Table I details the axioms and inference rules describing the behavior of belief, signed statement and reliability. Notice axiom schema (To_{\leq}) which reflects that reliability relations have to be believed as total. Let \vdash denotes the proof relation. We conclude by giving results about soundness and completeness.

Theorem 1 Logical system \mathcal{L} is sound and complete¹.

IV. LINKING SIGNATURES AND BELIEFS

There are multiple ways to switch from information to beliefs. These different ways may follow principles issued from the belief merging principle [16], [17], [5] or epistemic attitudes such as trust [7], [6]. As previously mentioned, we do not require that an agent has to believe that others

¹In this paper all proofs have been skipped; however a longer version of the paper with all proofs is downloadable at the URL <http://www.irit.fr/~Laurent.Perrussel/lads2010-long.pdf>.

believe in information they provide. This is a key issue when information is propagated from one agent to another. At some stage, an agent may just broadcast some information without committing to that information in terms of belief.

A common and rational way to proceed is to consider as belief all non mutually inconsistent signed statements. All signed statements are considered in an incremental way, that is “from the most reliable to the less reliable statements”. To describe the signed statements adoption stage, we first characterize agents which are equally reliable. Agents can be ranked since we always consider a total preorder; agents which are equally can be gathered in a same group. Each group can then be ranked. Let us at first characterize the most reliable set of agents; this set is denoted as C_1 :

$$a \in C_1 =_{def} \forall t (a \leq t)$$

The formula characterizing members of C_1 can then be used for characterizing membership to a set C_i such that $i > 1$.

$$a \in C_i =_{def} (\neg(a \in C_{i-1}) \wedge \forall t \neg(t \in C_{i-1})) \rightarrow (a \leq t)$$

Hence, all agents belonging to a set C_i are equally reliable and for all $a \in C_i, b \in C_j$ if $i <_{\mathbb{N}} j$ then $a \leq b$. Next, the following definition stands for each agent t^k belonging to some specific set C_i believes statement φ_0^k :

$$\bigwedge_{t^k \in C_i} \text{Sign}(t^k, \phi_0^k) =_{def} \bigwedge_{t^k \in A} (t^k \in C_i) \rightarrow \text{Sign}(t^k, \phi_0^k)$$

Using these shortcuts, we can now describe the merging process. The following axiom states that if a propositional statement φ_0 is believed by agent a if the conjunction of the statements signed by the agents belonging to the same set C_i entails φ_0 is believed by agent a (line 1), if statement $\neg\varphi_0$ is not already believed by a (line 2) and $\neg\varphi_0$ cannot be entailed with the help of statements signed by agents which are at least as reliable as agents belonging to C_i (line 3).

$$\begin{aligned} & (\text{Bel}(a, \bigwedge_{t^k \in C_i} \text{Sign}(t^k, \varphi_0^k)) \wedge \text{Bel}(a, \bigwedge \varphi_0^k \rightarrow \varphi_0) \wedge \\ & \quad \neg \text{Bel}(a, \neg\varphi_0) \wedge \\ & \quad (\bigwedge_{0 < j < i} \neg \text{Bel}(a, \bigwedge_{t^l \in C_j} \text{Sign}(t^l, \varphi_0^l) \wedge \bigwedge \varphi_0^l \rightarrow \neg\varphi_0))) \\ & \rightarrow \text{Bel}(a, \varphi_0) \quad \text{(IB)} \end{aligned}$$

In terms of semantics, it means that, w.r.t. some initial state w_0 , all belief states are related to some signed states. Hence, it requires to consider the state’s interpretation, that is to express the relation by using worlds, i.e. a state and its associated interpretation. We represent a world as a set of propositional symbols, symbols that hold in the associated state. Let w be a state and $[w]$ the associated world:

$$[w] = \{p \mid I(w, p) = 1\}$$

In a more general way, if W is a set of states, then $[W]$ denotes the set of associated worlds. At first, from the belief states, we rank agent ids based on reliability relations believed by the

agent. Suppose an agent a and a world w_0 ; using relation B_a , we extract the total preorder representing reliability relation believed by agent a at w_0 . Notice that the constraints shown section III-A1 ensures that this preorder is total and thus agents ids could be ranked for building a partition of set of agents. Let \mathcal{C} be a partition of \mathcal{A} such that in every set C_i of \mathcal{C} , all agents are equally reliable and for all $a \in C_i, b \in C_j$ if $i <_{\mathbb{N}} j$ then $a \prec b$. Second, from each set C_i , we consider common information, that is statements that are signed by every agents belonging to C_i . Let $[C_i]^{w_0}$ be the set of worlds commonly signed by all agents belonging to C_i and related to w_0 :

$$[C_i]^{w_0} = \bigcap_{a \in C_i} \{[w] \mid (w_0, w) \in S_a\}$$

Next all sets of worlds $[C_i]^{w_0}$ are merged in a consistent way, the resulting set of worlds is denoted as *reliable worlds*. By consistent way, we mean an incremental process which considers as reliable worlds at first the whole set of possible worlds $[W]$. Next for each part C_i , the set of reliable worlds is intersected with $[C_i]^{w_0}$ only if it does not lead to an empty set, i.e. an inconsistent result.

Definition 4 (Reliable worlds) *Let M be a model and w_0 a state such that $w_0 \in W$. The set of reliable worlds Ω^{w_0} is defined in an incremental way such that:*

- $\Omega^0 = [W]$
- $\Omega^i = \Omega^{i-1} \cap [C_i]^{w_0}$ if $\Omega^{i-1} \cap [C_i]^{w_0} \neq \emptyset$ and $i > 0$
- $\Omega^i = \Omega^{i-1}$ if $\Omega^{i-1} \cap [C_i]^{w_0} = \emptyset$ and $i > 0$

The resulting set Ω^{w_0} is equal to Ω^k such that $k = |\mathcal{C}|$.

Since the sets of worlds and agents are finite, we do not have to consider the infinite case. Reliable worlds represent information that should be actually believed. Let us consider agent a and an initial world w_0 ; from w_0 , we extract the belief states, and from these belief states, the set of reliable worlds. Beliefs of agent a are *rational* if all its believed worlds are included in its set of reliable worlds:

$$\bigcup_{(w_0, w) \in B_a} [w] \subseteq \bigcap_{(w_0, w) \in B_a} \Omega^w \quad \text{(IB)}$$

The following theorem relates formula **(IB)** and constraint **(IB)**. Let \vdash_{IB} denotes proof relation of the \mathcal{L} -system augmented with axiom schema **(IB)** and \models_{IB} be the satisfaction relation where for all models, constraint **(IB)** holds.

Theorem 2 $\vdash_{\text{IB}} \varphi$ iff $\models_{\text{IB}} \varphi$

V. ACQUIRING INFORMATION

In the previous section, we have detailed a policy for building belief based on signed information. This policy considers belief and signed information from a static point of view. Let us now consider a more dynamic view by introducing actions of the form “agent a tells to agent b that a certain fact φ_0 is true” (*alias* tell actions). This kind of action ensures that agent b will believes that agent a signs p , that is, a *tell* action is responsible for updating an agent’s beliefs about other agents’

signatures and, consequently, for the agent's acquisition of new information and for updating the agent's beliefs about objective facts. We note tell actions by $\text{Tell}(a, b, \varphi_0)$. Let \mathcal{L}_T be the extended language which embeds *tell* statements.

Definition 5 (Syntax of \mathcal{L}_T) *The set of formulas of the language \mathcal{L}_T is defined by the following BNF:*

$$\begin{aligned} \varphi ::= & p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \text{Sign}(t, \varphi_0) \mid \text{Bel}(a, \varphi) \mid \\ & \forall x\varphi \mid t \leq t' \mid [\text{Tell}(a, b, \varphi_0)]\varphi \end{aligned}$$

where $p \in \mathcal{P}$, $t \in \mathcal{T}$, $\varphi_0 \in \mathcal{L}_0$, $a \in \mathcal{A}$ and $x \in \mathcal{V}$.

In other terms, \mathcal{L}_T just extends \mathcal{L} with dynamic operators $[\text{Tell}(a, b, \varphi_0)]$. The intuitive meaning of statement $[\text{Tell}(a, b, \varphi_0)]\varphi$ is after a tells φ_0 to b , φ holds.

The truth conditions are those given above for the formulas p , $\neg\varphi$, $\varphi \wedge \varphi'$, $\text{Sign}(t, \varphi_0)$, $\text{Bel}(a, \varphi)$, $\forall x\varphi$, $t \leq t'$ and $[\text{Tell}(a, b, \varphi_0)]\varphi$. The truth condition for $[\text{Tell}(a, b, \varphi_0)]\varphi$ is defined in a way which is closed to the semantics of dynamic epistemic logic [19]. More precisely, after agent a tells to agent b information φ_0 , agent b removes from its belief state all states in which agent a does not sign φ_0 . Therefore, after agent a tells to agent b information φ_0 , agent b believes that agent a signs φ_0 . In our framework, a tell action of agent a (the sender) towards agent b (the receiver) that φ_0 is true is considered as a *private announcement* in the sense of [22], that is, after agent a tells to agent b information φ_0 , only agent b 's belief state should change whereas the belief states of the other agents are not changed. In other words, a *tell* action $\text{Tell}(a, b, \varphi_0)$ characterizes a *private communication* from a sender to a specific receiver of the sender's message, where the content of the speaker's message is nothing else than the content of the speaker's signature (i.e. $\text{Sign}(t, \varphi_0)$).

Definition 6 (Announcement Semantics) *Let $M = \langle W, \bigcup_{i \in \mathcal{A}} S_i, \bigcup_{i \in \mathcal{A}} B_i, I, \preceq \rangle$ be a model and let w be a state in W . We have:*

- $M, v, w \models [\text{Tell}(a, b, \varphi_0)]\varphi$ iff $M|_{\langle a, b, \varphi_0 \rangle}, v, w_1 \models \varphi$.

$M|_{\langle a, b, \varphi_0 \rangle} = \langle W^*, \bigcup_{i \in \mathcal{A}} S_i^*, \bigcup_{i \in \mathcal{A}} B_i^*, I^*, \preceq^* \rangle$ is defined as follows:

- $W^* = \{w_1 | w \in W\} \cup \{w_2 | w \in W\};$
- $B_b^* = \{(w_1, w'_1) | (w, w') \in B_b \text{ and } M, v, w' \models \text{Sign}(a, \varphi_0)\} \cup \{(w_2, w'_2) | (w, w') \in B_b\};$
- $B_i^* = \{(w_1, w'_1) | (w, w') \in B_i\} \cup \{(w_2, w'_2) | (w, w') \in B_i\}$ for all $i \in \mathcal{A}$ such that $i \neq b$;
- $S_i^* = \{(w_1, w'_1) | (w, w') \in S_i\} \cup \{(w_2, w'_2) | (w, w') \in S_i\}$ for all $i \in \mathcal{A}$;
- $\preceq^*(w_1) = \preceq^*(w_2) = \preceq(w)$ for all $w \in W$;
- $I^*(w_1, p) = I^*(w_2, p) = I(w, p)$ for all $w \in W$.

Basically, the effect of a 's action of telling to b that φ_0 is to shrink the set of belief accessible states for b to the states in which a signs φ_0 , while keeping constant the set of belief accessible states for all other agents. Note that a 's action of

$$\begin{aligned} (T_{AP}) \quad & [\text{Tell}(a, b, \varphi_0)]p \leftrightarrow p \\ (T_N) \quad & [\text{Tell}(a, b, \varphi_0)]\neg\varphi \leftrightarrow \neg[\text{Tell}(a, b, \varphi_0)]\varphi \\ (T_C) \quad & [\text{Tell}(a, b, \varphi_0)](\varphi \wedge \varphi') \leftrightarrow \\ & ([\text{Tell}(a, b, \varphi_0)]\varphi \wedge [\text{Tell}(a, b, \varphi_0)]\varphi') \\ (T_B) \quad & [\text{Tell}(a, b, \varphi_0)]\text{Bel}(b, \varphi) \leftrightarrow \\ & \text{Bel}(b, (\text{Sign}(a, \varphi_0) \rightarrow [\text{Tell}(a, b, \varphi_0)]\varphi)) \\ (T_{B_{\neq}}) \quad & [\text{Tell}(a, b, \varphi_0)]\text{Bel}(i, \varphi) \leftrightarrow \text{Bel}(i, \varphi) \quad \text{if } i \neq b \\ (T_S) \quad & [\text{Tell}(a, b, \varphi_0)]\text{Sign}(t, \varphi'_0) \leftrightarrow \text{Sign}(t, \varphi'_0) \\ (T_{\leq}) \quad & [\text{Tell}(a, b, \varphi_0)](t \leq t') \leftrightarrow (t \leq t') \\ (T_{\forall}) \quad & [\text{Tell}(a, b, \varphi_0)]\forall x\varphi \leftrightarrow \forall x[\text{Tell}(a, b, \varphi_0)]\varphi \end{aligned}$$

TABLE II: Logic \mathcal{L}_T axioms and inference rules

telling to b that φ_0 also keeps constant agents' signatures and the reliability order over agents.

Theorem 3 *If M is a \mathcal{L} -model then $M|_{\langle a, b, \varphi_0 \rangle}$ is also a \mathcal{L} -model.*

Theorem 4 *If M is a \mathcal{L} -model in which constraint IB holds then $M|_{\langle a, b, \varphi_0 \rangle}$ is also a \mathcal{L} -model in which constraint IB holds.*

Let us now focus on the axiomatics of the logic \mathcal{L}_T . Table II details the reduction axioms describing the behavior of the operator $[\text{Tell}(a, b, \varphi_0)]$. (T_{AP}) denotes the atomic permanence, (T_N) denotes negation handling, and (T_C) denotes conjunction handling. (T_B) describes the interplay between a *tell* action and the beliefs of the message receiver. $(T_{B_{\neq}})$ describes the interplay between a *tell* action and the beliefs of all agents different from the message receiver. In particular, $(T_{B_{\neq}})$ highlights the permanence of the beliefs of all agents different from the message receiver. (T_S) describes signature permanence, (T_{\leq}) describes preferences permanence, and (T_{\forall}) describes the interplay between *tell* action and quantification over variable assignments.

Theorem 5 *The schemata in table II are valid.*

We then state the theorem about completeness of the logic \mathcal{L}_T .

Theorem 6 *The logic \mathcal{L}_T is completely axiomatized by principles of the logic \mathcal{L} together with the schemata in Table II and the rule of replacement of proved equivalence.*

We write \vdash_T to denote the proof relation for the logic \mathcal{L}_T determined by the principles of the logic \mathcal{L} , the schemata in table II and the rule of replacement of proved equivalence.

For instance, the following theorem of the logic \mathcal{L}_T captures the essential aspect of the *tell* action. It says that, after agent a tells to agent b information φ_0 , agent b believes that agent a signs φ_0 :

$$\vdash_T [\text{Tell}(a, b, \varphi_0)]\text{Bel}(b, \text{Sign}(a, \varphi_0))$$

Once agent b starts to believe that agent a signs φ_0 (as an effect of a 's act of telling to b that φ_0), agent b might also start to

believe that φ_0 . As we have shown above, this depends on the reliability of agent a according to agent b and on principles linking signatures with beliefs such as principle **(IB)**.

Example 6 *Let us go back to our initial example and let us represent in the system \vdash_T , how agent po concludes that the blue car has caused the collision. At first, assume that **(IB)**. Second, assume the following preferences: $\text{Bel}(po, w_1 \leq w_2)$. Then it follows that after the two announcements (we focus on the blue car), preferences are unchanged:*

$$\vdash_T [\text{Tell}(w_1, po, bc)][\text{Tell}(w_2, po, \neg bc)]\text{Bel}(po, w_1 \leq w_2)$$

And the detective believes the received information

$$\vdash_T [\text{Tell}(w_1, po, bc)][\text{Tell}(w_2, po, bc)] \\ \text{Bel}(po, \text{Sign}(w_1, bc) \wedge \text{Sign}(w_2, \neg bc))$$

Finally, axiom **(IB)** entails that

$$\vdash_T [\text{Tell}(w_1, po, \neg bc \wedge rc)][\text{Tell}(w_1, po, bc \wedge \neg rc)]\text{Bel}(po, bc)$$

VI. CONCLUSION

In this paper we have shown how information and its source can be processed by an agent so that at first, it just acquires information from sensors or other agents and second, it builds its belief state by considering signed information. By splitting information and belief, an agent is able to handle clear rationales to construct its belief state both from a static and dynamic perspectives. From a static perspective we have applied our formal framework to characterize a possible attitude for agents in the process of building their belief state from the basic signed information they hold. From this perspective this work is close to what has been done in belief merging [16], [17], [5]. The key difference with existing work in the belief merging area is the introduction of merging in a modal based framework at first (this is also a common characteristic with [5]); second a clear distinction between belief and signed statement and third a dynamic view on belief construction. These last two characteristics differ in two ways from existing work [16], [17], [5]: (i) it is usually assumed that belief and information are almost similar; we have shown that we do not have to assume this hypothesis; (ii) beliefs are almost not viewed as a primitive concepts but rather as the result of some information processing which gives a flexible framework (e.g. axiom **(IB)**). Our work is also related to the work of [23] in which agents' mental attitudes and agent's ostensible (expressed) attitudes are distinguished and a formalism capturing this distinction is proposed. In particular, our notion of signed information is close to the notion of ostensible belief of Nickles et al. However, Nickles et al. do not consider reliability of information sources. Moreover, their approach does not deal with dynamics of information by means of communicative actions. The latter is a central aspect of our proposal (see Section V).

Concerning the dynamic perspective we have shown how the basic signed information held by an agent may change as it receives tell statements from another agent processed in a

similar way to private announcements in the sense of dynamic epistemic logic (DEL) [19], [22].

Our short term goal is to consider more sophisticated ways to set the reliability relations. That is, our aim is to consider agent skills [15] so that agent can consider multiple reliability relations at the same time. At this time, even if agent can consider multiple alternative reliability relations, they cannot mixed them. Our goal is to avoid this limit.

REFERENCES

- [1] A. Rao and M. Georgeff, "Modeling rational agents within a bdi-architecture," in *Proc. of KR'91*, 1991, pp. 473–484.
- [2] P. Gärdenfors, *Knowledge in flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.
- [3] A. Herzog and D. Longin, "Belief dynamics in cooperative dialogues," *J. of Semantics*, vol. 17, no. 2, 2000, vol. published in 2001.
- [4] A. Dragoni and P. Giorgini, "Revising beliefs received from multiple sources," in *Frontiers of Belief Revision, Applied Logic*. Kluwer, 1999.
- [5] L. Cholvy, "A modal logic for reasoning with contradictory beliefs which takes into account the number and the reliability of the sources," in *Proc of ECSQARU'05*, ser. LNCS, vol. 3571. Springer, 2005, pp. 390–401.
- [6] C. Liau, "Belief, information acquisition, and trust in multi-agent systems—a modal logic formulation," *Artificial Intelligence*, vol. 149, no. 1, pp. 31–60, 2003.
- [7] E. Lorini and R. Demolombe, "From Binary Trust to Graded Trust in Information Sources: A Logical Perspective," in *Trust in Agent Societies*, ser. LNAI. Springer-Verlag, 2008, vol. 5396, pp. 205–225.
- [8] W. van der Hoek and M. Wooldridge, "Towards a logic of rational agency," *Journal of the IGPL*, vol. 11, no. 2, pp. 133–157, 2003.
- [9] L. Perrussel and J. Thévenin, "(Dis)Belief Change based on Messages Processing," in *Proc. of CLIMA'IV*, 2004.
- [10] —, "A logical approach for describing (dis)belief change and message processing," in *Proc. of AAMAS'04*. IEEE C.S., 2004, pp. 614–621.
- [11] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language," in *Software Agents*, J. Bradshaw, Ed. MIT Press, 1997.
- [12] P. Cohen and H. Levesque, "Rational Interaction as the Basis for Communication," in *Intentions in Communication*, P. Cohen, J. Morgan, and M. Pollack, Eds. MIT Press, 1990, pp. 221–256.
- [13] —, "Communicative actions for artificial agents," in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, V. Lesser and L. Gasser, Eds. San Francisco, CA, USA: The MIT Press, 1995, pp. 65–72.
- [14] L. F. del Cerro, A. Herzog, D. Longin, and O. Rifi, "Belief reconstruction in cooperative dialogues," in *Proc. of AIMS'98*, ser. LNCS, F. Giunchiglia, Ed., vol. 1480. Springer, 1998, pp. 254–266.
- [15] L. Cholvy, "Automated reasoning with merged contradictory information whose reliability depends on topics," in *Proc. of ECSQARU'95*, ser. LNCS, C. Froidevaux and J. Kohlas, Eds., vol. 946, 1995, pp. 125–132.
- [16] P. Liberatore and M. Schaerf, "Arbitration (or how to merge knowledge bases)," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 1, pp. 76–90, 1998.
- [17] S. Konieczny and R. Pérez, "Propositional belief base merging or how to merge beliefs/goals coming from several sources and some links with social choice theory," *European Journal of Operational Research*, vol. 160, no. 3, pp. 785–802, 2005.
- [18] S. Benferhat and L. Garcia, "Handling locally stratified inconsistent knowledge bases," *Studia Logica*, vol. 70, pp. 77–104, 2002.
- [19] H. van Ditmarsch, W. van der Hoek, and B. Kooi, *Dynamic Epistemic Logic*, ser. Synthese Library. Springer, 2007, vol. 337.
- [20] R. Fagin and Y. M. J. Halpern, *Reasoning About Knowledge*. MIT Press, 1995.
- [21] H. Katsuno and A. Mendelzon, "Propositional knowledge base revision and minimal change," *Artificial Intelligence*, vol. 52, no. 3, pp. 263–294, 1991.
- [22] J. Gerbrandy and W. Groeneveld, "Reasoning about information change," *J. of Logic, Language and Information*, vol. 6, no. 2, 1997.
- [23] M. Nickles, F. Fischer, and G. Weiss, "Communication attitudes: A formal approach to ostensible intentions, and individual and group opinions," in *Proc. of LCMAS'05*, ser. Electronic Notes in Computer Science, vol. 157(4). Elsevier, 2005, pp. 95–115.

Towards Efficient Multi-Agent Abduction Protocols

Gauvain Bourgne
National Institute of Informatics
Tokyo, Japan
Email: bourgne@nii.ac.jp

Katsumi Inoue
National Institute of Informatics
Tokyo, Japan
Email: ki@nii.ac.jp

Nicolas Maudet
LAMSADE,
Paris Dauphine University, France
Email: nicolas.maudet@lamsade.dauphine.fr

Abstract—What happens when distributed sources of information (agents) hold and acquire information locally, and have to communicate with neighbouring agents in order to refine their hypothesis regarding the actual global state of this environment? This question occurs when it is not possible (*e. g.* for practical or privacy concerns) to collect observations and knowledge, and centrally compute the resulting theory. In this paper, we assume that agents are equipped with full clausal theories and individually face abductive tasks, in a globally consistent environment. We adopt a learner/critic approach. We present the Multi-agent Abductive Reasoning System (MARS), a protocol guaranteeing convergence to a situation “sufficiently” satisfying as far as consistency of the system is concerned. Abduction in a full clausal theory has however already a high computational cost in centralized settings, which can become much worse with arbitrary distributions. We thus discuss ways to use knowledge about each agent’s theory language to improve efficiency. We then present some first experimental results to assess the impact of those refinements.

I. INTRODUCTION

In multi-agent systems, the inherent distribution of autonomous entities, perceiving and acting locally, is the source of many challenging questions. To overcome the limitation of their own knowledge, usually local and incomplete, agents are driven to form some hypotheses and share information with other agents. Especially, abductive reasoning is a form of hypothetical reasoning deriving the possible causes of an observation. It can be used to complete an agent’s understanding of its environment by explaining its observations, or, more proactively, for planning, as one can try to find the possible actions that might cause the completion of a goal. However reasoning in a sound manner with distributed knowledge rises interesting problems, as one cannot ensure locally the consistency of an information. Moreover, the system often comes with severe communication restrictions, due to physical (*e. g.* the limited scope of a communication device) or reasoning (*e. g.* the mere impossibility to consider all the potential communications) limitations of agents populating it. For such situations, we presented in [1] a sound mechanism that is guaranteed to find an abductive hypotheses with respect to distributed full clausal theories whenever one exists. This Multi-agent Abductive Reasoning System, MARS, is based on a consequence finding tools named SOLAR [2], that serves as a main reasoning engine. We are concerned in this paper with the efficiency of this mechanism, and thus want to evaluate and improve its average computational and communicational cost.

Distributed abduction has been considered in recent years in the ALIAS system [3]. They distribute the abductive programming algorithm of [4], using abductive logic program to represent each agent’s theory. More recently, DARE [5] addressed a similar problem, but consider possible dynamicity of the system by allowing agents to enter or exit some proof cluster. In none of these works however is the issue of communication constraints explicitly raised. Another related work is the peer-to-peer consequence finding algorithm DeCA [6]. Based on a different method (splitting clauses), it is to our knowledge the only other work in this domain taking into account restrictions of communication between peers. It is however restricted to propositional theories. The work on partition-based logical reasoning presented [7] is of particular interest for our present study as it investigates efficient theorem proving in partitioned theories. It relies on communication languages describing the common symbol in the individual languages of pairs of agents. However, this approach and the previous one explore all the consequences of the distributed theories, whereas when we are only concerned with some *new* consequences of the theories with respect to some knowledge (namely the negated observations when computing a hypothesis through inverse entailment, or the hypothesis itself when ensuring its consistency). As a result, while inspirational, they cannot be directly applied to our approach.

The rest of this paper is as follows. Section II gives the necessary background on abduction and consequence finding. Then, Section III describe formally a multi-agent abduction problem, and present the MARS protocol, giving details about the communications exchanged over its execution. Efficiency is then discussed, and we describe two improvement on the previous protocol. These variants are then experimentally tested in Section IV, and we conclude in Section V.

II. ABDUCTIVE REASONING

A. Preliminaries

First, we review some notions and terminology to represent our problem in a logical setting. A *literal* is an atom or the negation of an atom. A *clause* is a disjunction of literals, and is often denoted by the set of literals. A clause $\{A_1, \dots, A_m, \neg B_1, \dots, \neg B_n\}$, where A_i and $\neg B_j$ are respectively positive and negative atoms is also written as $A_1 \vee \dots \vee A_m \leftarrow B_1 \wedge \dots \wedge B_n$. Any variable in a clause is assumed to be universally quantified at the front. A *clausal theory* is a finite set of clauses which can be identified with the

conjunction of the clauses. Let S and T be clausal theories. S *logically implies* T , denoted as $S \models T$, if and only if for every interpretation I such that S is true under I , T is also true under I . \models is called the *entailment relation*. For a clausal theory \mathcal{T} , a *consequence* of \mathcal{T} is a clause entailed by \mathcal{T} . We denote by $Th(\mathcal{T})$ the set of all consequences of \mathcal{T} . Let C and D be two clauses. C *subsumes* D , denoted $C \succeq D$, if there is a substitution θ such that $C\theta \subseteq D$. C *properly subsumes* D if $C \succeq D$ but $D \not\succeq C$. For a clausal theory \mathcal{T} , $\mu\mathcal{T}$ denotes the set of clauses in \mathcal{T} not properly subsumed by any clause in \mathcal{T} .

We can now introduce the notion of *characteristic clauses*, which represents “interesting” consequences of a given problem [8]. Each characteristic clause is constructed over a subvocabulary of the representation language called a *production field*, and represented as $\langle \mathcal{L} \rangle$, where \mathcal{L} is a set of literal closed under instantiation. A clause C *belongs* to $\mathcal{P} = \langle \mathcal{L} \rangle$ if every literal in C belongs to \mathcal{L} . For a clausal theory \mathcal{T} , the set of consequences of \mathcal{T} belonging to \mathcal{P} is denoted $Th_{\mathcal{P}}(\mathcal{T})$. Then, the characteristic clauses of \mathcal{T} wrt to \mathcal{P} are defined as $Carc(\mathcal{T}, \mathcal{P}) = \mu Th_{\mathcal{P}}(\mathcal{T})$, where μ is subsumption minimality¹. When a set of new clauses S is added to a clausal theory, some consequences are newly derived with this additional information. The set of such clauses that belong to the production field are called *new characteristic clauses* of S wrt \mathcal{T} and \mathcal{P} ; they are defined as $Newcarc(\mathcal{T}, S, \mathcal{P}) = Carc(\mathcal{T} \cup S, \mathcal{P}) \setminus Carc(\mathcal{T}, \mathcal{P})$.

B. Abductive hypothesis

The logical framework of hypothesis generation in abduction for the centralized case can be expressed as follows. Let \mathcal{T} be a clausal theory, which represents the *background theory*, and O be a set of literals, which represents *observations*. Also let \mathcal{A} be a set of literals representing the set of *abducibles*, which are candidate assumptions to be added to \mathcal{T} for explaining O . Given \mathcal{T} , O and \mathcal{A} , the abduction problem is to find a *hypothesis* H such that:

- (i) $\mathcal{T} \cup H \models O$ (accountability),
- (ii) $\mathcal{T} \cup H \not\models \perp$ (consistency), and
- (iii) H is a set of instances of literals from \mathcal{A} (bias).

In this case, H is also called an *explanation* of O (with respects to \mathcal{T} and \mathcal{A}). A hypothesis is *minimal* if no proposer subset of H satisfies the above three conditions (which is equivalent to subsumption minimality for ground clauses). A hypothesis is *ground* if it is a set of ground literals (literals containing no variable). This restriction is often employed in applications whose observations are also given as ground literals. In the following, we shall indeed assume that observations are grounded, and that we are only searching for minimal ground hypotheses.

C. Computation through hypothesis finding

Given the observations O , each hypothesis H of O can be computed by the principle of *inverse entailment* [8], [9],

¹meaning that μX represents the clauses of X that are not properly subsumed by any other clause of X .

which converts the accountability condition (i) to $\mathcal{T} \cup \{\neg O\} \models \neg H$, where $\neg O = \bigvee_{L \in O} \neg L$ and $\neg H = \bigvee_{L \in H} \neg L$. Note that both $\neg O$ and $\neg H$ are clauses since O and H are sets of literals. Similarly, consistency condition (ii) is equivalent to $\mathcal{T} \not\models \neg H$. Hence, for any hypothesis H , its negated form $\neg H$ is deductively obtained as a “new” theorem of $\mathcal{T} \cup \{\neg O\}$ that is not an “old” theorem of \mathcal{T} alone. Moreover, to respect the bias condition (iii), every literal of $\neg H$ has to be an instance of a literal in $\bar{\mathcal{A}} = \{\neg L \mid L \in \mathcal{A}\}$. Then the negation of minimal hypotheses are the new characteristic clauses of O with respect to \mathcal{T} and $\bar{\mathcal{A}}$, that is, $Newcarc(\mathcal{T}, \{\neg O\}, \bar{\mathcal{A}})$.

SOLAR [2] is a sophisticated deductive reasoning system based on SOL-resolution [8], which is sound and complete for finding *minimal* consequences belonging to a given language bias (a *production field*). Consequence-finding by SOLAR is performed by *skipping* literals belonging to a production field \mathcal{P} instead of resolving them. Those skipped literals are then collected at the end of a proof, which constitute a clause as a logical consequence of the axiom set. Using SOLAR, we can implement an abductive system that is *complete* for finding minimal explanations due to the completeness of consequence-finding. SOLAR is designed for *full clausal theories* containing non-Horn clauses, and is based on a *connection tableau* format [10]. In this format, many redundant deductions are avoided using various state-of-the-art pruning techniques [2], thereby hypothesis-finding is efficiently realized.

Once possible hypotheses have been computed, a ranking process can be applied to select a *preferred hypothesis* (e.g. hypothesis ranking such as in [11]). We will not dwell on this part here, and instead assumed that a *preference relation* \succeq_p over the hypothesis is given as a total order between sets of grounded literals.

III. DISTRIBUTED ABDUCTION

A. Problem setting

We propose here a new formalization of our problem as a *multi-agent abductive system*, which is defined as a tuple $\langle \mathcal{S}, \{\Gamma_t\}, \mathcal{A}, \succeq_p \rangle$, where:

- $\mathcal{S} = \{a_0, \dots, a_{n-1}\}$ is a set of agents. Each agent a_i has its own *individual theory* \mathcal{T}_i and its own *observations* O_i . It will also form its own preferred hypothesis H_i , though it can also adopt it from other agents. In fact, in the end of the process, all agents will share the same hypothesis.
- $\Gamma_t = \langle \mathcal{S}, E_t \rangle$ is the *communicational constraint graph* at time t , an undirected unlabeled graph whose nodes are the agents in \mathcal{S} and whose edges E_t represent the communicational links between the agent. An agent a_i can only communicate with another agent a_j at time t if $(a_i, a_j) \in E_t$.
- \mathcal{A} is the common set of *abducibles* that represents the language bias of the abductive process.
- \succeq_p is the *common preference relation*, a total order over hypotheses.

Theories and observations are considered to be certain knowledge. As such, they are assumed to be consistent,

meaning that $\bigcup_{i < n} \mathcal{T}_i \cup \bigcup_{i < n} \mathcal{O}_i \not\models \perp$. To ensure termination, it will also be assumed that $\text{Carc}(\bigcup_{i < n} \mathcal{T}_i, \langle \mathcal{L} \rangle)$ is finite, and that both hypotheses and observations are ground (*i.e.* contain no variable). Moreover, the system will be assumed to be *temporally connected*, meaning that at any time t , the graph $\Gamma_{t^+} = \langle \mathcal{S}, \bigcup_{t' \geq t} E_{t'} \rangle$ is a connected graph.

Our aim is then to ensure the formation of an abductive explanation of $\bigcup_{i < n} \mathcal{O}_i$ with respects to $\bigcup_{i < n} \mathcal{T}_i$ and \mathcal{A} . Given a group of agents $G = \{a_i, i \in J\} \subset \mathcal{S}$, we shall say that a hypothesis H is *group-consistent* with G iff it is consistent with the union of all the individual theory of the agents of the group, that is, iff $\bigcup_{i \in J} \mathcal{T}_i \cup H \not\models \perp$. Likewise, we shall say that H ensures *group-accountability* for G iff it can explain all observations of the agents of the group when it is associated with the union of their theories, that is iff $\bigcup_{i \in J} \mathcal{T}_i \cup H \models \bigcup_{i \in J} \mathcal{O}_i$. If $G = \mathcal{S}$, we shall say that the hypothesis is *mas-consistent* or that it ensures *mas-accountability*. Finally we shall say that a set of literals is *acceptable* for a group G iff it is a set of grounded literals of \mathcal{A} that is group-consistent with G and ensures group-accountability for G . The objective of a multi-agent abductive system is thus to find a hypothesis that is *acceptable* for the whole system.

While consistency or accountability of a hypothesis with respect to both $(\mathcal{T}_i, \mathcal{O}_i)$ and $(\mathcal{T}_j, \mathcal{O}_j)$ is not equivalent to consistency or accountability wrt $(\mathcal{T}_i \cup \mathcal{T}_j, \mathcal{O}_i \cup \mathcal{O}_j)$, we still can ensure some relation between them in classical logic. Specifically, group-inconsistency of H with G implies group-inconsistency of H with any superset of G , which ensures that hypothesis inconsistent with a sub-group of agents (possibly a single agent) can be ruled out as a potential solution. Moreover, group-accountability of H for both G and G' implies group-accountability of H for $G \cup G'$ (but not reciprocally), which ensures that accountability can be checked locally.

In order for a learner agent to propose a hypothesis to a critic, it is necessary that his agent can produce such a hypothesis. However, given only a few clauses of the whole clausal theory, it might not be able to find an explanation for the observations using only abducibles. Therefore, we shall allow an agent to build *partial hypothesis*, which contains some non-abducible literals. Those literals might be the unexplained observations, or preferably some other literals of the language that would explain it. While interacting with other agents, they will share knowledge to expand these hypotheses in order to progressively build a fully abducible one. Note that of course, a hypothesis respecting the bias condition will always be favored over one who does not.

We shall now present MARS, a mechanism for solving multi-agent abductive problems based on SOLAR.

B. Bilateral interaction

To deal with distributed hypothesis formation in multi-agent systems, we take a learner-critic approach, in which learner agents aim at producing a globally adequate hypothesis through internal computations and local interactions with other agents acting as critics. In our abductive setting, however, critic agent cannot ensure the consistency of the hypothesis

by itself, and needs to interact with the learner in order to find incoherence (computing the context of a hypothesis) and produce complete hypotheses (exchanging useful information by justifyin partial hypotheses). The underlying mechanism was presented and proved correct in [1]. Here, we shall introduce the actual protocol based on that procedure, recapitulating its main steps while giving an exact account of the communications involved.

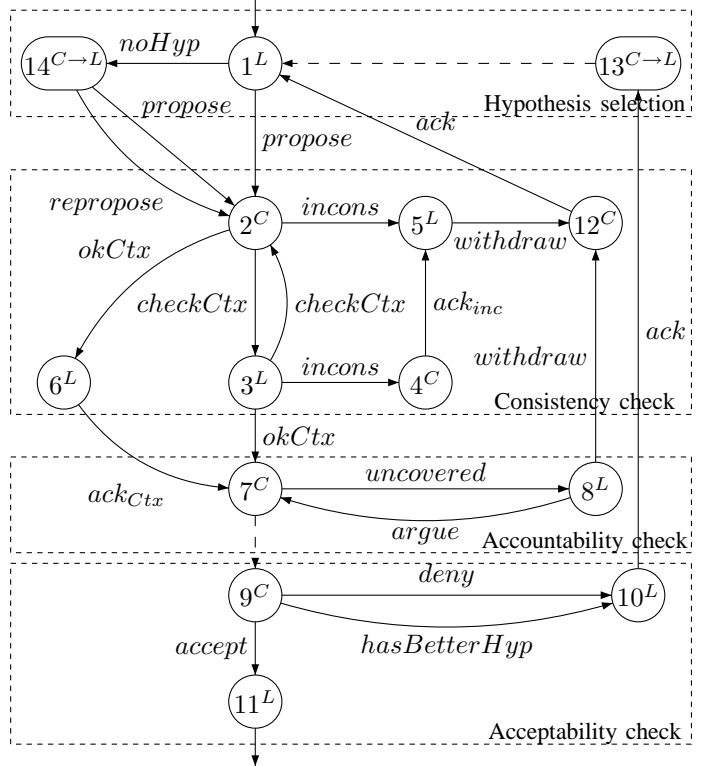


Fig. 1. Multi-Agent Learner-Critic Abductive Protocol.

Fig. 1 illustrate this protocol. Nodes indicate states of the agents (steps of the mechanism), with superscript L or C indicating whether it concerns the *Learner* agent or the *Critic* agent. Note that states 13 and 14 indicate a switching of the roles, as the critic becomes the learner. Labeled arcs indicate that a given message can be sent by an agent in a given state, making the other agent go to the target state upon reception. Dashed arcs indicate an internal change of state without communication. This mechanism is divided in four main steps that we shall now detail.

1) *Hypothesis selection*: An interaction is initiated by a learner agent a_0 , in state 1^L , proposing its hypothesis and its validity context to a critic agent a_1 ($\text{propose}(H_0)$). If learner's information has changed since it last computed its possible hypotheses, it will recompute them through inverse entailment, using $\bar{\mathcal{A}}$ as a production field. In case it cannot find a hypothesis this way, it will compute a *partial hypothesis* by using an *extended set of abducibles* (possibly the whole language). If the proposed hypothesis h_0 is a new one, the first context Ctx_0 will be computed as the new consequences of

$h_0 \cup \mathcal{T}_0$ wrt h_0 , that is $Newcarc(\mathcal{T}_0, h_0, \mathcal{P}_{\mathcal{L}})$ where $\mathcal{P}_{\mathcal{L}} = \langle \mathcal{L} \rangle$. Otherwise, the previously computed context will be used as initial context Ctx_0 . Then, when receiving such proposal, a_1 will start its critic, which consists of three steps: consistency check, accountability check and admissibility check. As the interaction continues, new hypotheses might have to be proposed. If the current learner cannot propose a hypothesis (which can only happen if it has blocked all its possible hypotheses during previous admissibility check), it will send *noHyp* to the other agent to switch the roles (state 14). If this one has also exhausted all its hypotheses, then it will unblock all its hypotheses and propose again the best one (*repropose*). Note that the new critic agent will also unblock all its hypotheses when receiving such a message.

2) *Consistency check*: When receiving a proposed hypothesis and context, the first step of the critique is to check the group-consistency of the hypothesis with both agents involved. A context is progressively built for a given hypothesis H to compute the new consequences of $H \cup \mathcal{T}_0 \cup \mathcal{T}_1$ wrt to $\mathcal{T}_0 \cup \mathcal{T}_1$. If the hypothesis is incoherent, then it will have \perp as a consequence. The occurrence of a contradiction between the context and the agents' theories will thus enable detection of incoherent hypotheses. This relies on the fact that the global theory itself is assumed to be consistent, so any inconsistency can only arise from the hypotheses. Indeed, if \mathcal{T} is consistent and $\mathcal{T} \cup H$ is inconsistent, then $Newcarc(\mathcal{T}, U, \mathcal{P}_{\mathcal{L}}) = \{\perp\}$. The process is as follows:

1. First, remember that during the hypothesis selection step, learner agent a_0 retrieves context Ctx_0 of its hypothesis. If no context has been memorized from previous iteration, then a new one is computed as $Newcarc(\mathcal{T}_0, H_0, \mathcal{P}_{\mathcal{L}})$ (note that we should thus have $H_0 \in Ctx_0$).

2. When receiving H_0 and Ctx_0 (state 2^C), a_1 first checks if it already has some context Ctx' for this given hypothesis. If it is the case, then it replaces Ctx_0 by $Ctx'_0 = Ctx_0 \cup Ctx'$. Context Ctx_1 is then computed as $Newcarc(\mathcal{T}_1, Ctx'_0, \mathcal{P}_{\mathcal{L}})$, and sent back to a_0 with message *CheckCtx*(Ctx_1) (unless a contradiction is found).

3. The process continues. At each step Ctx_i is computed by agent a_α as the new consequences $Newcarc(\mathcal{T}_\alpha, Ctx_{i-1}, \mathcal{P}_{\mathcal{L}})$, where $\alpha = 0$ if i is odd (state 3^L), and $\alpha = 1$ otherwise (state 2^C), and sent with a *checkCtx* message.

4. This computation stops when either a contradiction is found or Ctx_i is included in either Ctx_{i-1} or Ctx_{i-2} , in which case all consequences have been computed.

- If an inconsistency is discovered, the part p_0 of the hypothesis responsible for it is sent to the other agents with message *incons*(p_0), leading eventually to state 5^L . Both agents rule out p_0 (and any hypothesis containing it) by adding its negation to their theory. The learner agent then moves on to its next hypothesis and proposes it, triggering a new critic phase (states 12^C and 1^L).
- Otherwise, the end of the computation is acknowledged by sending *okCtx*. Both agents memorize the final context $Ctx_f = Ctx_i \cap Ctx_{i-1}$ (where i is the final

step) of this hypothesis. Any element in respectively $Ctx_i \setminus Ctx_{i-1}$ and $Ctx_{i-2} \setminus Ctx_{i-1}$ are added to $\mathcal{T}_{1-\alpha}$ and \mathcal{T}_α where again $\alpha = 0$ if i is odd and 1 otherwise. Indeed an element will only be removed from the context if it is a direct consequence of one of the agent's theory. The critic phase moves to the next step (state 7^C).

3) *Accountability check*: In this step, the critic agent checks if all its observations are explained by $H_0 \cup \mathcal{T}_1$. If an unexplained observation o is found, the message *uncovered*(o) is sent to the learner agent, now in state 8^L . We then have two possibilities.

- If o is not explained by $H_0 \cup \mathcal{T}_0$, it is a true counterexample. The learner agent then computes a new hypothesis that will also cover o , and proposes it, triggering a new critic phase (states 12^C and 1^L).
- If o is already explained by $H_0 \cup \mathcal{T}_0$, then the learner agent will notify the critic of this fact with *argue*(p_0), where p_0 is the part of the hypothesis that is used in explaining o with \mathcal{T}_0 . The critic agent will add the clause $\{o \vee \neg p_0\}$ in its theory². This new information will ensure that the critic agent can find the hypothesis on its own in further steps, or build up upon it. It will then proceed to the next unexplained observation.

If there is no unexplained observation, the critic proceeds to the next step (state 9^C).

4) *Acceptability check*: Any hypothesis that reaches this step is consistent and accounts for the observations, but it might include some non-abducible literals, or unnecessary parts. This step ensures that alternative hypotheses are explored if needed.

1. If the critic has a hypothesis H_c that is preferred to H_0 (according to \geq_p), it will reverse roles (*hasBetterHyp*) and submit it. This will finally either result in the acceptance of a better hypothesis, or cause the former critic agent to learn why its hypothesis cannot be used.

2. Otherwise, if the hypothesis contains non-abducibles (partial hypothesis), the critic agent will temporarily block it, and ask the other agent to do the same (*deny*). I will then also switch roles (state $13^{C \rightarrow L}$). This ensures that all partial hypotheses that could provoke information exchange leading to building an abducible hypothesis are explored if needed.

3. If the hypothesis is acceptable, or if a partial hypothesis has been reproposed (meaning the exploration is complete), then the critic sends an *accept* message. The final outcome of the interaction is thus chosen. Hypotheses that were temporarily blocked are unblocked, and the best hypothesis is chosen as the final hypothesis. It is adopted and memorized by both agents, ending the interaction.

C. Group of agents

Each interaction allows the participants to refine their hypotheses and augment their knowledge concerning their consequences. The protocol described before is enough to allow two agents to form a hypothesis that is group-consistent and

²Note that since p is a conjunction of literals, $\neg p_0$ is indeed a clause.

ensures group-accountability for the pair of agents. When more agents are involved, it is possible to chain such interactions to converge towards a consistent state of the system. To take into account possibly variable communication constraints in the system, we propose a rumor-like approach, ensuring the local behaviour and interactions of the agents make the system converges to a state in which all agents have a mas-consistent hypothesis ensuring mas-accountability.

An agent is motivated by the will to ensure it has an explanation with respect to its neighbours. As such, it will attempt to have local interactions with them whenever needed to ensure that, memorizing the result of their last interaction with each of their neighbours. In practice, an agent a_i will engage in a local interaction with a neighbour a_j whenever its hypothesis and context (h_i, Ctx^i) differ from those obtained during its last interaction with a_j .

In [1], this process was proved to be sound, and to guarantee that a solution is found if there is one.

D. Improving efficiency

The main computational cost of our mechanism lies in the multiple calls to a consequence-finding tools, which is used in the various steps to conduct the logical reasoning, especially for computing possible hypotheses through inverse entailment, computing the context of these hypotheses, and checking their accountability. To improve efficiency, it is thus crucial to reduce as much as possible the computational cost of each of these calls, as well as to reduce their number.

The tools we are using in our implementation, SOLAR, is based on tableaux methods. We assess the computational cost of a call by counting the number of inferences performed during it. Without entering in the details of the procedure, we will discuss here the factors that influence the cost of the computation of $Newcarc(F, \mathcal{T}, \mathcal{P}_{\mathcal{L}})$. The number of inferences is directly related to the number of clauses used in the procedure. Used clauses are clauses which can be resolved with one of the top clauses (elements of F), or with a consequence of them. Thus, reducing the number of clauses in \mathcal{T} and more importantly in F can both help to reduce the computations steps. Note that $Carc(\mathcal{T}, \mathcal{P}_{\mathcal{L}})$ is in practice computed as $Newcarc(\mathcal{T}, \emptyset, \mathcal{P}_{\mathcal{L}})$, so computing new consequences rather than all consequences is already a good step to ensure better efficiency. Reducing the number of literals of the top clauses also helps as it limits the number of clauses they can be resolved with. Then, another factor that can affect the computations is the size of the production field. A small production field limits the number of options to be explored and as a results, the number of inferences to be done.

With respects to our mechanism, these considerations means that we should keep each agent's individual theory as small as possible, which is ensured by adding single clauses with just the necessary parts of the hypothesis to memorize inconsistencies (when sending or receiving $incons.(p_0)$ or accountability arguments (when sending $argue(o \vee \neg p_0)$ in state 8^L). Moreover, during consistency check, we should minimize the computations for the context. We shall see in next subsection

how to reduce size of top clauses during this step by doing incremental computations. Then, we should also find ways to minimize the number of consequences computed during this consistency step by focusing on consequences that could lead to a contradiction, and even more importantly, to limit the number of partial hypotheses computed by focusing on those partial hypotheses that could trigger information exchanges leading to the formation of an acceptable hypothesis (as it would also reduce the number of applications of SOLAR).

E. Incremental consistency check

During consistency check, context is progressively computed until it does not evolve anymore, but sending the whole context at each step of the computation and using it as top clause for computing the next step. To avoid redundant communications and computations, we propose to communicate only the new consequences of the context, pruning consequence discovered in previous step. It does not change the computation and sending of Ctx_0 and Ctx_1 , but after computing Ctx_1 , the learner agent will only send back $ctxStep_1 = Ctx_1 \setminus Ctx_0$ (note that we use the original Ctx_0 here, and not Ctx'_0).

Then, when receiving $checkContext(ctxStep_i)$, agent a_α first computes $NC_{i+1} = NewCarc(ctxStep_i, \mathcal{T}_\alpha \cup Ctx_{i-1})$. It can then use it to compute the current context $Ctx_{i+1} = Ctx_{i-1} \cup NC_{i+1}$, and send the update $ctxStep_{i+1} = NC_{i+1} \setminus ctxStep_i$. If there is a clause c than is in $ctxStep_i$ that is not subsumed by any clause of NC_{i+1} , it means that it is a consequence of \mathcal{T}_α . It should then be sent to the other agent (with message $inform(c)$) to ensure that both agents will have the same final context. This replaces the theory adjustment with $Ctx_i \setminus Ctx_{i-1}$ and $Ctx_{i-2} \setminus Ctx_{i-1}$ that were made before. Note that the termination condition becomes much simpler, as the context can be confirmed as soon as $ctxStep_i$ is empty.

F. Language focus

1) *Languages*: Given a clausal theory T , we denote by $L(T)$ the set of non-logical symbols that occur in T , and by $\mathcal{L}(T)$ the language formed upon them. Each agent has its own theory \mathcal{T}_i , from which we can define its *individual language* $\mathcal{L}(\mathcal{T}_i)$. We can then compute for each pair of agent a_i, a_j ($i \neq j$), in the manner of [7], the *communications language* $\mathcal{L}_{i,j} = \mathcal{L}(\mathcal{T}_i) \cap \mathcal{L}(\mathcal{T}_j)$. It can be used to direct the focus of bilateral communications. If it is empty, a_i and a_j do not need to communicate together. However, it may be the case that a_i and a_j are never connected while $\mathcal{L}_{i,j}$ is not empty. For the sake of simplicity we will assume that the communicational links are static³. We shall then adapt the communications languages by choosing a minimal path for all such pair of unconnected agents (a_i, a_j) , and add the $\mathcal{L}_{i,j}$ to the communication language of each pair of agent in this path. In the following, when referring to the communication

³otherwise, since the system is assumed to be temporally connected, it is possible to find a connected subgraph that is included in Γ_{t+} for all t and use it as a guaranteed basis.

language $\mathcal{L}_{i,j}$, we will assume that this modification has been done. Then the *restricted individual language* of an agent a_i is defined as $\mathcal{L}_i = \bigcup_{j \in N_i} \mathcal{L}_{i,j}$, where N_i is the set of the indexes of the neighbours of a_i . At last, the *common language* is the language $\mathcal{C} = \bigcup_{i < n} \mathcal{L}_i$, that is the union of all restricted individual languages (which is also the union of all the communication languages).

2) *Context narrowing*: When computing context, we want to ensure that any new consequence of the hypothesis that can be derived from the union of the agent's theories is indeed found. It means a_0 need to send any consequence of H wrt \mathcal{T}_0 that could be resolved with a clause of \mathcal{T}_1 . In practice, a_0 compute its context using its *restricted individual language* as a production field, and then retain in Ctx_0 only the one that contains at least a literal of the concerned communication language (here $\mathcal{L}_{0,1}$). Upon receiving it, a_1 will then temporarily add $\mathcal{L}(Ctx_0)$ to $\mathcal{L}_{0,1}$, and if it already has a context Ctx' , a_1 will use the same pruning before adding it to get Ctx'_0 and compute Ctx_1 (with his restricted individual language as a production field). The pruning (with updated language) is applied to Ctx_1 (or $ctxStep_1$) before sending it, and the process continues. Each time, contexts are pruned to exclude any clauses that do not have literals in the current communication language before being sent to the other agent. Note that since we compute only new consequences, we cannot directly use the communication language as a production field, as shown by the following example:

Example 1: Let's take $\mathcal{T}_0 = \{-h \vee a \vee b, \neg h \vee o\}$, $\mathcal{T}_1 = \{-a\}$ and $\mathcal{T}_2 = \{-b\}$, all agents being connected. We have $\mathcal{L}_{0,1} = \mathcal{L}(\{a\})$, and $\mathcal{L}_{0,2} = \mathcal{L}(\{b\})$, so $\mathcal{L}_0 = \mathcal{L}(\{a, b\})$. We assume a_0 has observation o and wants to check hypothesis h with a_1 . If Ctx_0 was computed with $\mathcal{L}_{0,1}$, it would be empty, and no contradiction would be found when a_0 checks later with a_2 . However, using \mathcal{L}_0 as a production field, we get consequence $a \vee b$ that contains literal $a \in \mathcal{L}_{0,1}$. It is thus sent to a_1 that will give in return b . When proposing this context to a_2 later on, a_0 will thus be able to derive a contradiction.

3) *Choice of partial hypotheses*: For computing partial hypotheses, and deciding whether to propose a given one to a neighbour or not, the same principles can be used. When no admissible hypothesis can be found, inverse entailment is performed again with an extended set of abducibles. To ensure that at least one solution can be found, the manifestations are added to the abducibles, enabling trivial explanations for some part of the hypothesis. Then, the idea is to use literals that can act as links between the theories. In practice, it means that we should include in the extended abducibles the literals in the restricted individual language of the agent. This should also be augmented with literals obtained through the arguments of other agents (when receiving $argue(o \vee p_0)$ in state 7^C). This allows us to compute all potentially useful partial hypotheses. For a given exchange, however, it is sufficient to propose those partial hypotheses that contain at least one literal of the communication language of the interacting agents.

IV. EXPERIMENTS

We describe here preliminary experimental results on a small set of problems⁴, testing our two improvements of the MARS protocol (namely, incremental context computation and restriction of the languages). Though it might be useful to assert the validity of our conclusions on a broader number of problems, we believe that the small problems used for evaluation highlight the main difficulties that can be encountered in a distributed abduction system.

The first problem, `pb-1`, is taken from [1], where it was used as a running example. It contains 10 clauses, distributed among 2 or 3 agents. With 3 agents, we tested two communicational constraint topologies: a line ($a_0 \leftrightarrow a_1 \leftrightarrow a_2$) and a completely connected system. This problem was designed to illustrate the MARS protocol, and thus make it go through all the possible states during its unfolding. The second problem, `pb-fvar`, is a toy problem with two observations that contains some clauses with unlinked variables. We pruned out hypotheses that contain variables in the resolution to respect our language bias. It is distributed among 3 agents, and here again, we tested it with line and completely connected graph topologies. The third problem, `chain_n` is a propositional problem designed to show a kind of worst case for distribution. It consists of three chains of implications linking respectively h_1 to o_1 (through k_{n-2}, \dots, k_0), h_1 to o_2 (through m_{n-2}, \dots, m_0) and h_2 to o_1 (through l_{n-2}, \dots, l_0). Moreover, one agent (agent $a_{n/2}$) has a constraint $\neg o_1 \vee \neg o_2$, which makes h_1 inconsistent. The aim is then to explain o_1 with abducibles $\{h_1, h_2\}$. Each agent knows 3 rules, one from each chain, and agent a_0 initially has observation o_1 . This chain was tested with $n = 8$, with either a line topology (from a_0 to a_7) or a circuit topology (as the line, with an additional link between a_0 and a_7). To check the influence of the number of agents, we also used a version of this problem with 4 agents, `chain_8.4`, in which a_0, a_1, a_2, a_3 are merged with respectively a_4, a_5, a_6 and a_7 , and a version with 2 agents, `chain_8.2`, in which a_0 are merged with respect to even and odd indexed agents. At last, we used a more practical problem, `schedule_var`, which is an adaptation from a scheduling problem presented in [5], with 8 agents. `schedule_dir` is a direct translation of the same problem from its original formalization as an abductive logic program (negation by default is dealt with by using additional abducibles).

Table I gives the results for the four variants of our mechanism. Computational cost is given by the total number of operations performed by the consequence finding tool over the course of the protocol, whereas communicational cost is expressed as the total number of bits exchanged by the agents during the process. From these results, it is obvious that using individual communication languages does indeed greatly reduce both costs. It is especially true for the most complex problems, and the gain ratio is more important when there

⁴Complete description of all these problems can be found at <http://rjcia09.fr/MARS>.

TABLE I
EXPERIMENTAL RESULTS

Language focus		no	no	yes	yes
Incremental ctx comp.		no	yes	no	yes
Computational cost					
Pb-1	2 ag.	711	666	711	666
Pb-1 (line)	3 ag.	1 602	1 454	1 548	1 390
Pb-1 (clique)	3 ag.	2 216	2 003	1 713	1 673
Pb-fvar (line)	3 ag.	11 890	11 818	8 019	7 959
Pb-fvar (clique)	3 ag.	13 680	14 126	6 457	6 415
Chain_8.2	2 ag.	31 171	21 330	12 438	8 675
Chain_8.4 (line)	4 ag.	57 406	45 803	29 844	24 285
Chain_8.4 (circ.)	4 ag.	81 998	70 168	23 999	21 075
Chain_8 (line)	8 ag.	133 696	103 219	22 450	18 600
Chain_8 (circ.)	8 ag.	92 986	75 146	9 015	8 639
Schedule _{var}	8 ag.	53 607	50 571	39 391	39 725
Schedule _{dir}	8 ag.	381 992	372 998	95 909	94 963
Communicational cost					
Pb-1	2 ag.	617	552	617	552
Pb-1 (line)	3 ag.	1 832	1 700	1 624	1 511
Pb-1 (clique)	3 ag.	2 540	2 373	2 115	2 064
Pb-fvar (line)	3 ag.	3 177	3 080	2 659	2 622
Pb-fvar (clique)	3 ag.	3 568	3 494	2 139	2 106
Chain_8.2	2 ag.	12 230	8 924	5 746	4 238
Chain_8.4 (line)	4 ag.	25 606	14 312	22 278	12 620
Chain_8.4 (circ.)	4 ag.	35 072	12 576	35 531	11 883
Chain_8 (line)	8 ag.	65 582	57 305	14 625	13 383
Chain_8 (circ.)	8 ag.	50 749	47 317	6 509	6 627
Schedule _{var}	8 ag.	28 774	27 171	20 448	20 752
Schedule _{dir}	8 ag.	219 335	209 398	77 638	76 238

are a greater number of communicational links. Incremental computation of context is however less convincing, as it only helps when there are several context computations step, which is not such a common occurrence, unless theories are really mixed (it is the case for `pb-1` and all `chain_8` problems, which do benefit from this improvement). In the end, this improvement is useful, but only marginally so in most situations. While more experiment would be required to say anything more definite, the present results give us some hint about the influence of topology and “encoding”. Having a topology with cycle can lead to redundant computations, but can also provide easier exchange of information by avoiding the extra cost of bringing back a crucial fact or rule (as demonstrated by the addition of the link a_0-a_7 in `chain_8`). Overall, using individual communication language allows us to reduce the cost of redundant computations, so that we can take more benefit from situations where additional links are helpful. Moreover, reducing the number of agents can be either detrimental (in `chain_8`) or beneficial (in `pb-1`): the size of the communication languages seem to be a more relevant factor. At last, the huge difference between `schedulevar` and `scheduledir` seems to indicate that our protocol is much more efficient for finding whether an abducible hypothesis is consistent than it is for finding an abducible hypothesis by exploring all partial hypotheses. When formalizing a given problem, it is thus more efficient to ensure one agent can easily generate candidate hypotheses, and express rules that constrain it.

V. CONCLUSION

We presented in this paper a formalization of a multi-agent abduction problem, and proposed a sound mechanism for computing an abductive explanation that is guaranteed to

find a solution whenever one exists. We then discussed way to improve the average efficiency of this protocol, called Multi agent Abductive Reasoning System (MARS). Two improvements were proposed. The first one reduce the costs of building a complex context by doing the computation incrementally. It only helps when several steps are needed and was therefore shown to have only a limited impact on efficiency by experimental results. The main improvement consist of using informations about the individual language of each agent to focus the exchanges on what can really advance the search for a hypothesis (or the inconsistency of a candidate hypothesis). Contrarily to [7], we do not need the communication graph to be made cycle-free. While our approach use a similar idea of using *communication language*, we are only interested in the new consequences of some formulas, and thus want to avoid computing all consequences of a theory. As a result, we showed that we needed to allow the exchanges of clauses that belongs only partially to the communication language. Nonetheless, it is still an important efficiency improvement compared to the more naive approach of using only the common language for all exchanges. Experimental results showed that it substantially reduces the number of computations as well as the size of the communications. More improvement should however be brought to the search of hypotheses that can only be produced by using the theories of several agents. The learner-critic assumption that hypothesis are produced locally might be unadapted in such situations. It might thus be better to design a collaborative hypothesis formation, though another lead could be to refine the current information exchange to ensure a better treatment of “sub-goals”.

REFERENCES

- [1] G. Bourgne, K. Inoue, and N. Maudet, “Abduction of distributed theories through local interactions,” in *Proc. of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, August 2010.
- [2] H. Nabeshima, K. Iwanuma, and K. Inoue, “Solar: A consequence finding system for advanced reasoning,” in *Autom. Reas. with Analytic Tableaux and Rel. Meth.* Springer, 2003, pp. 257–263.
- [3] A. Ciampolini, E. Lamma, P. Mello, F. Toni, and P. Torroni, “Cooperation and competition in ALIAS: a logic framework for agents that negotiate,” *Annals of Math. and AI*, vol. 37, no. 1–2, pp. 65–91, 2003.
- [4] A. C. Kakas and P. Mancarella, “Database updates through abduction,” in *Proc. of VLDB ’90*. Morgan Kaufmann Pub., 1990, pp. 650–661.
- [5] J. Ma, A. Russo, K. Broda, and K. Clark, “DARE: a system for distributed abductive reasoning,” *JAAMAS*, vol. 16-3, pp. 271–297, 2008.
- [6] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, “Distributed reasoning in a peer-to-peer setting: Application to the semantic web,” *J. Artif. Intell. Res. (JAIR)*, vol. 25, pp. 269–314, 2006.
- [7] E. Amir and S. A. McIlraith, “Partition-based logical reasoning for first-order and propositional theories,” *AI*, vol. 162, no. 1-2, pp. 49–88, 2005.
- [8] K. Inoue, “Linear resolution for consequence finding,” *Artif. Intell.*, vol. 56, no. 2-3, pp. 301–353, 1992.
- [9] S. Muggleton, “Inverse entailment and prolog,” *New Generation Comput.*, vol. 13, no. 3&4, pp. 245–286, 1995.
- [10] R. Letz, K. Mayr, and C. Goller, “Controlled integration of the cut rule into connection tableau calculi,” *JAR*, vol. 13, pp. 297–338, 1994.
- [11] K. Inoue, T. Sato, M. Ishihata, Y. Kameya, and H. Nabeshima, “Evaluating abductive hypotheses using an em algorithm on bdds,” in *Proc. of IJCAI’09*, 2009, pp. 810–815.

Validation of Agile Workflows using Simulation

Kai Jander Lars Braubach Alexander Pokahr Winfried Lamersdorf
Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
{jander | braubach | pokahr | lamersd}@informatik.uni-hamburg.de

Abstract—Increasing automation of business processes and industrial demand for complex workflow features have led to the development of more flexible and agile workflow concepts. One of those concepts is the use of goal-oriented workflows, which rely on ideas derived from agent technology like describing the workflows based on a goal hierarchy. While this reduces the gap between business view and IT view and allows for easy implementation of contingencies, the concepts have greater conceptual abstraction obscuring the control flow and reducing the ability of workflow engineers to identify specification flaws in the workflow. This paper shows an approach to address this problem by presenting a system for testing and validating workflows within a specified parameter space. The system allows the definition of test cases (scenarios), each of which contains parameter states applied during workflow execution. The workflow engineer can define a set of scenarios for a workflow testing specific situation that are likely to occur during operation or are otherwise interesting corner cases, allowing automated tests and correction of faults before deployment of the workflow in production environments.

I. INTRODUCTION

Business process management (BPM) is considered a very promising strategy that helps in aligning companies towards effective and efficient business operation [1]. This is mainly achieved by completely thinking in terms of processes, which means that even the structure of organizations has to follow the processes and cannot be kept in a functional orientation. The vision of BPM assumes that processes can at least partially be automated, monitored according to key performance indicators (KPI) and continually improved or even renewed according to the measurement and defined KPI targets. It becomes clear that this vision heavily depends on adequate IT means supporting these tasks, which e.g. manifests in the development of workflow notations and management solutions.

One fundamental problem with existing solutions that has been experienced in practice is that modelling notations like event process chains (EPCs) or the business process modelling notation (BPMN) are activity oriented and thus focus heavily on the ordering and conditions of activity execution. This leads to a particular neglect of the underlying process motivations, which are only implicitly existing during workflow elicitation. Without this so called workflow context perspective [2] it becomes e.g. difficult to optimize the processes because it cannot be easily afforded why specific activities in the workflow exist and if they e.g. could be completely cut out. These observations led to the development of more abstract, flexible and agile workflow concepts. While these concepts contribute to closing the gap between business view and IT view, the greater conceptual abstraction also partially hides the complexities of all possibilities of the exact runtime control

flow and reduces the ability of workflow engineers to easily identify specification flaws in the workflow.

In order to equip a workflow modeller with a toolset to better understand the possible runtime behaviour of more abstract workflows, simulation and validation gain importance. In this paper a new concept and implementation for a simulation based validation approach of workflows is presented. It allows to specify execution scenarios and automatically evaluate them with respect to expected process outcomes. The approach does not depend on the concrete workflow notation in which processes are described but only assumes that specific workflow management facilities are available. Thus, the approach is viable for validating traditional e.g. BPMN based workflows as well.

The rest of the paper is structured as follows. In Section II related work with respect to verification and validation approaches of workflows is discussed. Thereafter, in Section III the concept of the new simulation based validation approach is presented. Its implementation and usage is explained in Section IV and the usefulness of the approach is further illustrated by an example application taken from our industry cooperation partner Daimler AG. The paper concludes with a summary and a short outlook on possible future enhancements.

II. RELATED WORK

Literature and practical application contains a large variety of validation and verification approaches for workflows. They can roughly be divided into two categories. The first category consists of formal static analysis of the workflow model to identify conceptual or implementation flaws. The second category comprises of validation using simulation-based execution of the workflows.

Formal verification approaches of workflows apply a number of techniques, such as propositional logic [3], model verification [4] and graph reduction [5]. The approaches are usually either based on a formally well-defined representation like petri nets [6] or attempt to translate workflows implemented in different representation like EPCs or BPMN into a more approachable form from a formal perspective [7]. While formal verification has the distinct advantage of guaranteeing correctness within the given constraints of the verification approach, translation of workflow models weakens this guarantee unless the translation itself is formally proven correct. Verification also requires a well-defined language with well-known properties, however, some languages used in practices lack these criteria. For example, many parts of BPMN in its current form are left ambiguous and underspecified. The

reason for this usually is not negligence but an attempt to bridge the gap between the technical side and the business side of workflows. Parts of the specification are deliberately left fuzzy with ambiguous semantics which enables the use of the specification in informal and non-technical settings. However, some of the issues of ambiguity are currently addressed in the upcoming version 2.0 of the BPMN specification.

Furthermore, verification approaches are often quite limited on what they can guarantee. For example, while there are excellent approaches to guarantee correctness of the workflow diagram like ensuring that branches in the workflow are terminated with a correct join element, they are generally unable to verify non-trivial semantics like task instructions written in a programming language or complex runtime behavior.

This problem can be approached by the validation techniques in the second category. Instead of statically analyzing the workflow models, the workflow is executed in a simulated environment which attempts to imitate the environment in which the workflow is planned to be deployed. Examples of this approach include a variety of tools like LSIM [8], iGrafx Process [9], the Corporate Modeler Suite [10] and the ARIS Toolset [11]. The tools generally target specific workflow notations such as BPMN to combine static analysis described above with simulation. Furthermore, the focus of the tools tend to be performance measurement and optimization. For example, the ARIS Toolset offers a large number of features used to measure operating and performance figures of the workflow during simulation.

In contrast, the focus of our approach has primarily been the validation of the workflow using predefined test cases centering around expected real world scenarios. The disadvantage of this approach compared to verification is that the correctness can only be ensured within the given conditions of the test. Since the number of configurations in any non-trivial workflow is very large, exhaustive search becomes infeasible. This means the approach can only validate the workflow to a certain degree and may not detect all errors present. The advantage of the approach is that the complexity of the workflow language is irrelevant to the test as long as there is a workflow engine capable of executing the workflow. Furthermore, the current approach uses few assumptions about the workflow itself allowing the system to be useful for different kinds of workflow notations. While the focus has been on validation of workflows, the approach can be extended to include simulation of the environment in which the workflow is running but which is, by itself, not part of the workflow. This includes logistic operations, production machinery, behavior of workflow participants and market influences. This allows additional applications beyond workflow validation like workflow optimization similar to the tools mentioned above.

In the following we will present this simulation-based testing approach which uses simulation of workflow participants to validate the correctness of workflows. The focus will be on goal-oriented and agile workflows, however, as mentioned, the approach itself can at least partially be applied to any workflow which relies on interaction with participants.

III. VALIDATION APPROACH

Due to its application in industry and business automation and intense research interest, a great variety of workflow approaches and notations are available. While our validation approach is designed to be generic, two kinds of notations in particular were the focus of the project. The first is the well-known *Business Process Modeling Notation* (BPMN, see [12]), which has been extended with task and edge annotations, allowing it to be directly executed by an interpreter. In addition, an additional notation called *Goal-oriented Process Modeling Notation* (GPMN, see [13]) has been developed, which allows the description of goal-oriented workflows using goal hierarchies. Workflows implemented using GPMN are converted into BDI agents at runtime. BDI agents are based on the Belief-Desire-Intention model where beliefs represents the agent's current knowledge about the world, goals represent its abstract desires of what should be accomplished and plans represent concrete intents of the agent with explicit actions the agent follows (see [14]). The goals used in GPMN workflows are directly converted to goals of the resulting agents while the plans that are on the leaf nodes of the goal hierarchy are represented by small workflow fragments implementing the concrete steps in BPMN. During execution, the GPMN-derived BDI agent employs a BPMN interpreter to execute the fragments as plans of the agent.

In addition, GPMN workflows contain a context which represents the current workflow state. This context is used as the belief base of the converted BDI agent at runtime. This context may be changed during execution of the workflow, either directly by the workflow itself or by effects outside the workflow. Changes in the context can directly affect the workflow and thus the agent behavior by influencing adoption, pursuit and rejection of goals.

A. Requirements for Automated Testing

Both GPMN and BPMN offer language features which allow the workflow engineer to implement different execution paths or branches. In the case of BPMN workflows, this is accomplished using the gateway element, which can split the control flow into either multiple paths executed concurrently or diverts the flow towards one of multiple possible control flow edges. The implementation of execution paths in GPMN workflows is more subtle and indirect. Depending on the state of the workflow context, different goals may become active resulting in the execution of different BPMN plans. This control flow subtlety of implicit control flow paths in GPMN workflows increases the difficulties of a workflow engineer to accurately predict possible runtime execution paths and is the primary motivation for our validation approach.

The core idea of our approach towards validating such workflows is the use of automated tests. The goal of the approach is to provide the workflow engineer with tools to specify realistic business cases which are likely to occur in the real world which are then applied to a workflow instance. As a result, the test coverage of the approach is limited to those cases, however, it is aimed at the most likely situations, providing assurance of validity in the cases most likely to

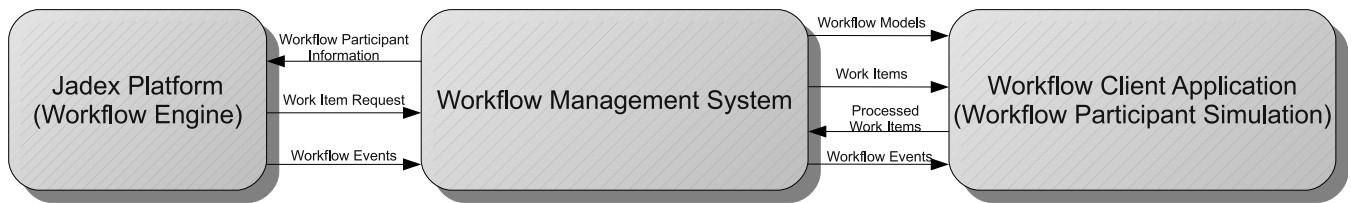


Figure 1. Structure of the proposed test system

occur after deployment. Since only the most trivial workflows can be automatically executed merely using a workflow engine and since most workflows require interaction with workflow participants or automated systems while running, additional system components are necessary beyond the workflow engine itself. Moreover, while most workflows specify a range of possible responses by workflow participants, they generally do not specify which responses will influence workflow behavior, thus necessitating the specification of additional information by the workflow engineer before the test.

Consequently from a simulation perspective, a workflow engine executing a workflow is not a viable simulation model for validation workflows since it lacks sufficient detail even for simple automated execution, much less being a realistic representation of a production environment using workflows. Therefore it is necessary to add additional components to the system which are equivalent or at least sufficiently similar to their production counterparts to represent an adequate model of a workflow in production use (see Figure 1).

B. Workflow Management System

One component which is routinely part of workflow systems in businesses is a *workflow management system* (WfMS). The task of a WfMS among other things is to facilitate interaction between workflows and workflow participants. This is generally done by providing work items, which are packages generated by the WfMS on behalf of the workflow containing all the information needed by the workflow participant to accomplish their part of the workflow. The WfMS then distributes the work items among the workflow participants using a variety of approaches such as roles. As a result, the simulation model of a realistic test system needs to include a WfMS which accurately represents a WfMS used in production.

Work items generated by the WfMS not only include information for the workflow participant but often ask the participant to gather and provide external information like customer data or processed documents for the workflow. This is defined in the workflow with the specification of typed parameters in tasks which generate work items. This information often influence further behavior of the workflow at critical junctions like BPMN gateways or goal deliberation. Since real workflow participants are not available during test runs, it is necessary to simulate their actions, including the supply of external information.

Work items are generally retrieved and processed by the workflow participant using a workflow application client interacting with the WfMS. The work items are retrieved, processed by the workflow participant and finally committed back to the

workflow management system, thus allowing the workflow to continue executing. In order to simulate this behavior, the workflow client application used by the workflow participant needs to be replaced with an automated workflow client application which simulates its behavior and the behavior of the workflow participant.

C. Client Application

The simulated workflow application client is required to provide the information which is normally provided by the workflow participant. As a first step, the client identifies the workflow tasks which generate work items and require the workflow participant to provide information in the form of work item parameters by examining the workflow model and the models of possible subprocess, such as BPMN workflow fragments in case of GPMN workflows. Parameters are typed and thus already have a limited parameter space. However, this parameter space in cases such as string types is extremely large, precluding an exhaustive test of the full parameter space. Since a complete verification of the process using this approach would also require to test the cartesian product of the parameter space of all parameters provided by the workflow participant, the complexity of such a verification exceeds the limits for a practical test and cannot be considered a useful approach.

As a result, it is necessary to restrict the scope of the test to only include the part of the parameter space which includes the most promising cases, such as corner cases of branching decisions and validating the workflow only for those cases. Since the test cases cannot be identified automatically, the workflow engineer has to define the parameter space that needs to be tested. This is accomplished by the system by allowing the workflow engineer to define test *scenarios*. Scenarios represent a subset of the full parameter space of the workflow participant interaction with the workflow. For each parameter in the process the workflow engineer can define a set of parameter values which are used to process work items while the workflow is executing. If the workflow engineer defines multiple values for each parameter within the same scenario, the cartesian product of those values is tested at runtime.

The workflow engineer can define multiple scenarios for each workflow. When the test is started, the first scenario is selected and the workflow is started repeatedly, once for every element of the cartesian product of the parameter values in that scenario. Once the scenario finishes, the next scenario is selected until all scenarios have been tested. An event log is kept during each execution, recording notable events occurring at runtime for later analysis. The workflow engineer

can use this log to identify errors in the workflow and correct them before the workflow is used in a production system. In addition, a test report is generated and can be reviewed.

Errors in the workflow can be unrecoverable states of the workflow like a raised exception during execution or unintended behavior of the workflow such as performing a faulty execution order of tasks. In additions, it is also considered to be an error if the workflow returns the wrong results or reaches the wrong internal state. Since the workflow engineer has to be informed about such errors occurring, monitoring of the workflow is required. On raised exceptions, executed steps of the workflow and state changes the workflow engine can generate a workflow event which is passed through the WfMS to the client application for review by the workflow engineer. It would also be desirable to allow the workflow engineer to define a validation function which receives the information provided to the workflow and the final state and result of the workflow.

The following section will elaborate on the individual parts of the testing system. It will include an overview of the workflow management system and provide details about the simulated workflow client application and how the workflow engineer can define the parameter space subset for each scenario.

IV. SIMULATION SYSTEM COMPONENTS

As mentioned in the previous section, the testing system requires a minimum of three components. In order to execute the workflows themselves, a workflow engine is needed, a workflow management system is needed for work item management and user interaction and there needs to be a special workflow client which simulates user behavior.

Since GPMN workflows are translated into BDI agents with BPMN plans and thus require a workflow engine which can execute both, the Jadex Active Components Platform [15] has been chosen as the execution environment for the workflows. The platform is not only capable of executing GPMN-derived BDI agents but also includes a BPMN interpreter which allows the execution of BPMN processes alongside agents. While the platform is able to execute standalone BPMN processes as active components, BPMN workflow fragments which represent GPMN plans are executed with a special BPMN plan interpreter, which allows the BPMN workflow fragments to access the GPMN context in the form of the agent belief base.

A. Workflow Management System Architecture

The second required component is the workflow management system. This system component is implement largely based on the reference model of the Workflow Management Coalition [16]. It uses the Jadex platform with Jadex platform services implementing the services required for the system which will be explained in further detail. In addition, the system includes three interface agents which realize a message-based interface with workflow application clients.

The services of the WfMS are divided into internal and external services, the former implementing the actual functionality of the WfMS while the latter, represented by the agents,

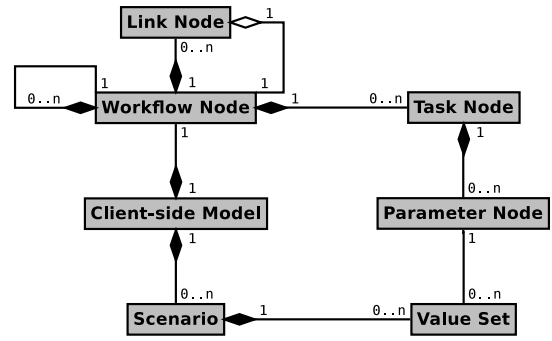


Figure 2. The workflow tree model used on the client side of the testing system

act as an interface which can be used by workflow client applications to connect with the WfMS. The Jadex platform itself represents the workflow engine and enactment service of the WfMS by providing the necessary support for instantiation of workflow models.

Workflow models are managed by the WfMS using a process model repository service. This service supports the addition and removal of process models by employing the Jadex library service which allows Jadex to dynamically load new models, resources and executable code by linking directories or jar-archives. In addition, it offers access to workflow models for workflow client software, which is a necessity for the testing system since the simulated workflow client application requires the workflow model in order to identify tasks in the workflow which require interaction with a workflow participant.

The Authentication, Access Control and Accounting (AAA) Service of the WfMS provides additional workflow services like access control and role management. Each task generating work items can assign a role to the work item, restricting this work item to participants who represent that role. Work items without a role become available to any participant connected to the system.

The external services of the WfMS consist of three parts. The first service is the *workflow client interface*, which manages the work item queue and distributes work items to connected clients. The second service is the *process definition interface*, which provides access to the model repository to allow the user to add and remove workflow models. Finally, the *administration and monitoring service* offers access to administrative and monitoring functions. The monitoring functions are especially critical for the testing system since they provide feedback regarding events happening during workflow execution.

This system provides a similar functionality to a workflow system in production use. In addition to this basic system, a workflow application client which simulates the behavior of workflow participants is needed to create an automated testing system which can execute tests of workflows without user intervention. This client is implemented as a BDI agent which connects to the three external service agents to the WfMS. The agent provides a user interface to the workflow engineer, which allows them to open the desired workflow model which

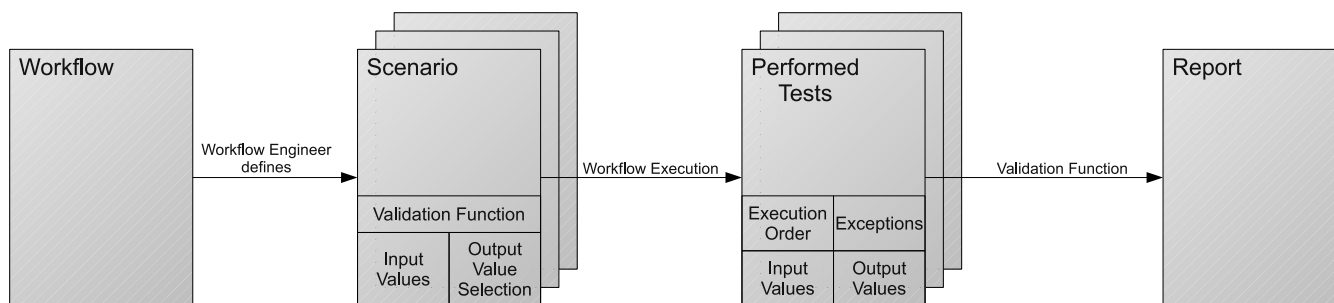


Figure 3. Procedure for testing workflows

is retrieved from the WfMS.

B. Client-side Workflow Model

Since it is desirable for the workflow engineer to gain an overview of the parts of the workflow which are relevant to interaction with workflow participants, the client agent uses the workflow model to generate a tree representation of the workflow model which can be seen in Figure 2. The tree consists of a *workflow node* as the root node, which represents the workflow originally opened by the workflow engineer. If a workflow contains sub-workflows like BPMN plans, the children of its workflow node can include further workflow nodes representing those sub-workflows. The sub-workflow graph of a workflow can contain cycles, for example, when a sub-workflow uses its parent as a sub-workflow. This would suggest a graph representation to be the natural form for representing the workflow structure. However, cycles in the workflow graph are rare in practice and a workflow engineer would expect a tree form rather than a graph. Therefore the tree representation is more desirable, nevertheless the special case of a cyclic workflow structure should be supported. This problem is solved with the use of *link nodes*. If a particular sub-workflow is found again after having been found before, it is represented as a link node in the tree. This link node is a simple reference to the first occurrence of the sub-workflow in the structure and no further expansion of the tree is done beyond this reference to avoid endless expansion.

If a workflow node is a BPMN workflow or workflow fragment its child nodes can, in addition to sub-workflow nodes, contain *task nodes* which represent tasks containing interaction with workflow participants. Lastly, the children of task nodes are *parameter nodes*, which represent typed parameters which would ordinarily be provided by a workflow participant.

C. Scenarios

The tree structure of a modelled workflow is presented to the workflow engineer in graphical form in the user interface. This allows them to define *scenarios*. Scenarios consist of sets of input values for each parameter of the workflow, definition of data collected from the workflow during the simulation and a validation function which is used to evaluate the success of the test and generate a report for the workflow engineer. Figure 3 demonstrates the use of scenarios in the full test procedure.

For each of the input parameters, the workflow engineer can add multiple values depending on the type of the parameter. The cartesian product of the input values in the scenario is used to create tests for the workflow which means that a minimum of a single value for each parameter is required for the scenario to generate at least a single viable test.

The workflow engineer can also select what is monitored during execution. This can include output values of the workflow, its final internal state, exceptions and the BPMN task elements executed. These values are passed to the *validation function* after execution to determine whether the test was a success and to generate a useful report. The validation function is also defined by the workflow engineer and included in the scenario. The validation function is defined in advance and is specified either by a Java class implementing the validation functionality or alternatively through the use of a number of configurable basic tests like comparison of result values with expected results.

Multiple scenarios can be defined for each workflow which can be automatically executed in succession. The total number of required test runs of the workflow are reported to the workflow engineer while assembling the scenarios. Since the cartesian product of multiple parameter values in a scenario quickly increases the complexity of the whole test, the workflow engineer has to carefully choose the tests and carefully balance between adding additional scenarios for the test run or adding additional parameter values to a single scenario.

The results of a test run can be reviewed in the report generated by the validation functions of the scenarios and is displayed to the workflow engineer in the client application. In addition, several tools provided by the Jadex platform can be used during the simulation as well. This includes and introspector tool for investigated the state of the workflow and a message center tool for monitoring messages between workflows and their support systems like the WfMS and the client application.

The next section will present an example use case for an industrial workflow used for change management as envisioned by Daimler AG and will demonstrate how the test system can be used to find implementation errors in advance of deployment.

V. EXAMPLE USE CASE

This section will present how the system can be used to validate a workflow. The workflow was developed by Daimler

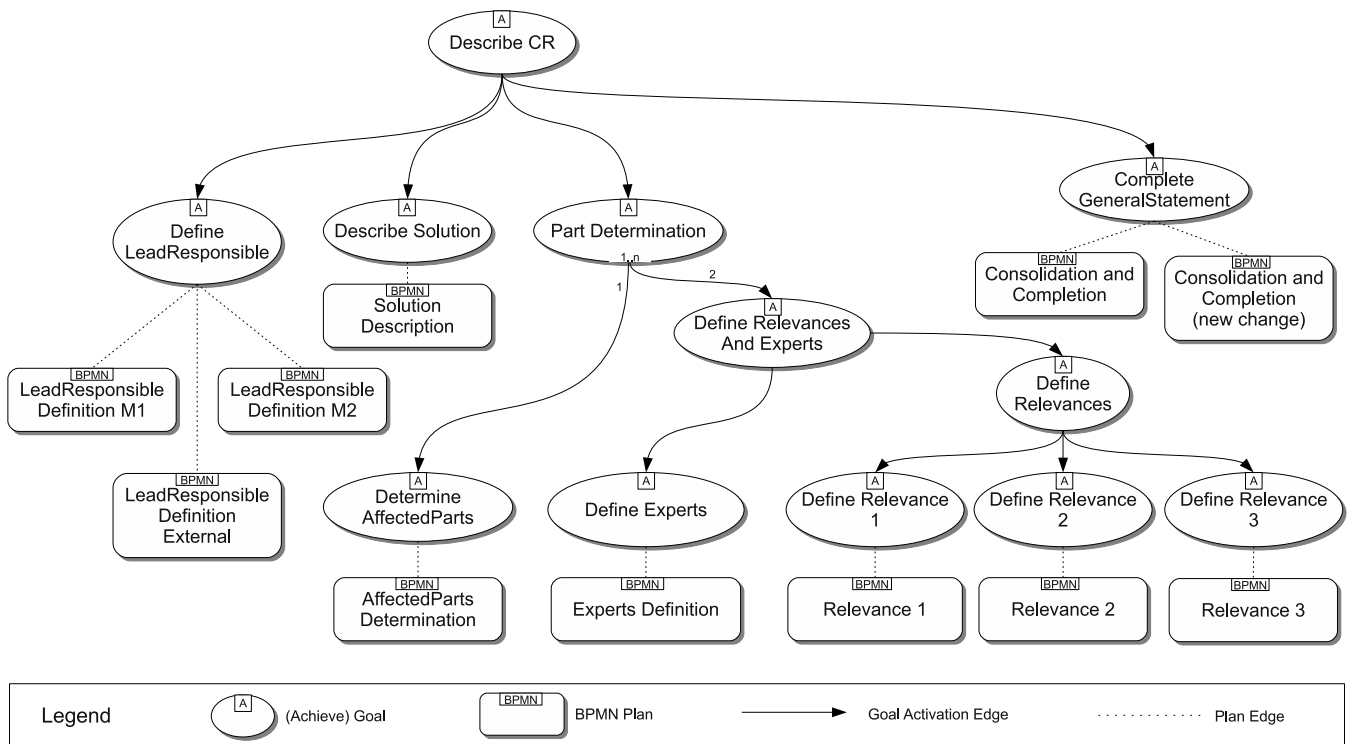


Figure 4. The goal hierarchy of the Active Change Management workflow

Group Research and represents an industrial workflow used for change management (see [17]). Change management workflow coordinate the process of developing and implementing changes for an existing product and ensures that production line changes and adaptations of the physical geometry of the product are performed in order to allow a smooth introduction of the changed product.

Since the workflow is very large and complex, this section will focus on a smaller subset of this workflow (cf. Figure 4). This subset involves the gathering of information about the planned change to the product, designating key personnel and assigning required resources. The final result of this part of the process is a description of the change request and requirements which will be used in the later part of the workflow to perform the change.

The process fragment contains a single goal for defining the change request. This goal is decomposed into subgoals, some of which contain context conditions which suspend their execution until the context has the required state for the goal to be adopted. Goals without further subgoals are associated with plans which are implemented by BPMN workflow fragments.

After implementation, one of the BPMN workflow fragments contained an error. The BPMN fragment containing the error was the fragment determining the parts affected by the change in the product. During the execution of the fragment, the leading developer responsible for the change request is required to enter the parts of the product which are affected by the change. The developer has the choice to do this using three different ways of providing this list of parts. The first way is to provide a list of serial numbers of the affected parts. The second way is to provide a drawing which is processed for

part information and finally, the developer can give a structured description of the components affected by the change.

The first task lets the developer choose between those three ways of providing the part list (cf. Figure 5). The first task in this fragment generates a work item containing a list of three strings which represent the choices of the developer. The developer can select one of the strings and commits the work item. The string is then passed to the gateway, and compared to strings provided by the edges behind the gateway branch. If the string matches, the process continues executing using that path and provides the developer with a new work item which contains the information for the chosen method of entry.

The implementation error in this part of the process was that one of the strings provided by the edges did not match with the corresponding string generated by the entry type selection task. Since none of the edges on the gateway branch are marked as default edge which would be taken if no other edge matches, the workflow will terminate with an exception if the developer selects the faulty choice.

This error was found using the test system. A scenario had been created test each of the branches leaving the gateway in this workflow fragment. For every parameter in the workflow except the entry method choice of the developer a single value was added to the scenario. All three possible strings were then added to the parameter concerning the entry choice resulting in a scenario which specifically target this branch in the workflow (cf. Figure 6). In addition, example entries for the part specification had been added in order to verify the correct function of the workflow in identifying the parts affected by the change, test corner case entry such as empty strings for parts and test another branch in the workflow where

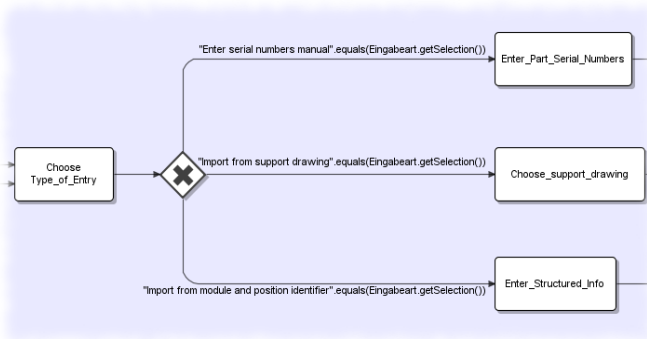


Figure 5. The leading developer of the change has three ways of providing a list of affected parts

the developer has to confirm the list of parts or otherwise cause a restart of the workflow fragment. The test complexity of the scenario required 972 test runs which were executed on an Intel i5 CPU clocked at 2.67GHz in less than a minute.

Some test runs resulted in an exception and the termination of the workflow. This event was logged during the simulation, including the parameter configuration used which caused the error. This result allowed the workflow engineer to find the fault in the workflow and correct the problem.

VI. SUMMARY AND FUTURE ENHANCEMENTS

This paper has presented a simulation-based validation approach for workflows. The approach allows specifying execution scenarios in form of test cases, which include ranges of input values to be tested and defined output states to be reached for a successful execution. The approach is especially well-suited for agile process descriptions with abstract specification means, such that possible process execution paths cannot easily be predicted. Nevertheless, the approach also works well with traditional process specification languages like BPMN. The implementation of the validation approach is based on the process execution facilities of the Jadex active component platform and provides additional tools for the specification, execution and validation of scenarios. The applicability of the approach has been exemplified by a case study from our project partner Daimler AG. It has been shown how the validation approach allows testing a modeled process to find and resolve existing problems in the process description.

Two interesting areas for future work are envisaged. First, the approach can be extended towards being used not only for process validation, but also for process analysis and optimization. Extending the simulation engine with elaborated analysis tools would allow measuring the quality of processes and benchmarking alternative processes against each other. Second, instead of using pre-specified test cases as scenarios, complex simulation models could be used to dynamically produce realistic test data. E.g. for logistics management processes, a simulation model of a supply chain could be connected to the process engine and provide input data for the workflow application to be tested.

Acknowledgement: We would like to thank the DFG for supporting the technology transfer project Go4Flex.

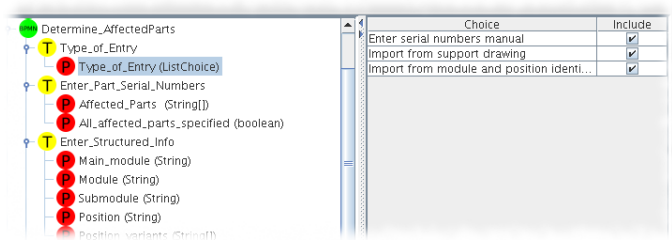


Figure 6. The scenario for testing the entry choice branch contains all three possible parameter values used

REFERENCES

- [1] W. S. H. J. Schmelzer, *Geschäftsprozessmanagement in der Praxis*. Hanser Fachbuchverlag, 2008.
- [2] B. List and B. Korherr, "An evaluation of conceptual business process modelling languages," in *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2006, pp. 1532–1539.
- [3] H. H. Bi and J. L. Zhao, "Applying propositional logic to workflow verification," *Information & Software Technology*, vol. 5, pp. 293–318, July 2004.
- [4] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based verification of web service compositions," in *18th IEEE international conference on automated software engineering, Montreal, Canada, 2003*, 2003.
- [5] W. Sadiq and M. E. Orłowska, "Analyzing process models using graph reduction techniques," *Information Systems*, vol. 25, no. 2, pp. 117 – 134, 2000, the 11th International Conference on Advanced Information System Engineering.
- [6] W. M. P. v. d. Aalst, "Workflow Verification: Finding control-flow errors using petri-net-based techniques," in *Business Process Management, Models, Techniques, and Empirical Studies*. London, UK: Springer-Verlag, 2000, pp. 161–183.
- [7] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," *Information & Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008.
- [8] L. J. Enstone and M. F. Clark, "BPMN and Simulation," Lanner Group Limited, 2006. [Online]. Available: http://www.dynamic.co.kr/Witness_Training_Center/Articles/Bpnm%20-%20simulation.pdf
- [9] "iGrafx Process," Corel Inc., 2009. [Online]. Available: <http://www.igrafx.de/products/process/index.html>
- [10] "Corporate Modeler Suite," Casewise Ltd, 2009. [Online]. Available: <http://www.casewise.com/Products/CorporateModelerSuite/>
- [11] A.-W. Scheer and M. Nüttgens, "ARIS architecture and reference models for business process management," in *Business Process Management, W. van der Aalst, J. Desel, and A. Oberweis, Eds.* Springer, 2000, pp. 376–389.
- [12] *Business Process Modeling Notation (BPMN) Specification*, Version 1.1 ed., Object Management Group (OMG), Feb. 2008. [Online]. Available: http://www.bpmn.org/Documents/BPMN_1-1_Specification.pdf
- [13] L. Braubach, A. Pokahr, K. Jander, and W. Lamersdorf, "Go4flex: Goal-oriented process modelling," in *In Proceedings of the 4th International Symposium on Intelligent Distributed Computing (IDC 2009)*. Springer, 2010.
- [14] M. Bratman, *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.
- [15] A. Pokahr, L. Braubach, and K. Jander, "Unifying agent and component concepts - Jadex Active Components," in *Proceedings of the 8th German conference on Multi-Agent System Technologies (MATES-2005)*, J. Dix and C. Witteveen, Eds. Springer, 2010.
- [16] *Workflow Reference Model*, Workflow Management Coalition (WfMC), Jan. 1995. [Online]. Available: <http://www.wfmc.org/reference-model.html>
- [17] B. Burmeister, M. Arnold, F. Copaciu, and G. Rimassa, "BDI-agents for agile goal-oriented business processes," in *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multi-agent systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 37–44.

JaCa-Android: An Agent-based Platform for Building Smart Mobile Applications

Andrea Santi
University of Bologna
Cesena, Italy
Email: a.santi@unibo.it

Marco Guidi
University of Bologna
Cesena, Italy
Email: marco.guidi7@studio.unibo.it

Alessandro Ricci
DEIS, University of Bologna
Cesena, Italy
Email: a.ricci@unibo.it

Abstract—Agent-Oriented Programming (AOP) provides an effective level of abstraction for tackling the programming of mainstream software applications, in particular those that involve complexities related to concurrency, asynchronous events management and context-sensitive behaviour. In this paper we support this claim in practice by discussing the application of AOP technologies – *Jason* and *CARtAgO* in particular – for the development of smart mobile applications based on the Google Android platform.

I. INTRODUCTION

The value of Agent-Oriented Programming (AOP) [20] – including Multi-Agent Programming (MAP) – is often remarked and evaluated in the context of Artificial Intelligence (AI) and Distributed AI problems. This is evident, for instance, by considering existing agent programming languages (see [4], [6] for comprehensive surveys) – whose features are typically demonstrated by considering toy problems such as block worlds and alike.

Besides this view, we argue that the level of abstraction introduced by AOP is effective for organizing and programming software applications in general, starting from those programs that involve aspects related to reactivity, asynchronous interactions, concurrency, up to those involving different degrees of autonomy and intelligence. Following this view, one of our current research lines investigates the adoption and the evaluation of existing agent-based programming languages and technologies for the development of applications in some of the most modern and relevant application domains. In this context, a relevant one is represented by next generation mobile applications. Applications of this kind are getting a strong momentum given the diffusion of mobile devices which are more and more powerful, in terms of computing power, memory, connectivity, sensors and so on. Main examples are smart-phones such as the iPhone and Android-based devices. On the one side, smart mobile applications share more and more features with desktop applications, and eventually extending such features with capabilities related to context-awareness, reactivity, usability, and so on, all aspects that are important in the context of Internet of Things and Ubiquitous Computing scenarios. All this increases – on the other side – the complexity required for their design and programming, introducing aspects that – we argue – are not suitably tackled

by mainstream programming languages such as the object-oriented ones.

So, in this paper we discuss the application of an agent-oriented programming platform called *JaCa* for the development of smart mobile applications. Actually *JaCa* is not a new platform, but simply the integration of two existing agent programming technologies: *Jason* [5] agent programming language and platform, and *CARtAgO* [17] framework, for programming and running the environments where agents work. *JaCa* is meant to be a general-purpose programming platform, so useful for developing software applications in general. In order to apply *JaCa* to mobile computing, we developed a porting of the platform on top of Google Android, which we refer as *JaCa-Android*. Google Android is an open-source software stack for mobile devices provided by Google that includes an operating system (Linux-based), middleware, SDK and key applications.

Other works in literature discuss the use of agent-based technology on mobile devices—examples include AgentFactory [13], 3APL [10], JADE [3]. Differently from these works, here we do not consider the issue of porting agent technologies on limited capability devices, but we focus on the advantages brought by the adoption of agent-oriented programming level of abstraction for the development of complex mobile applications.

The remainder of the paper is organised as follows: in Section II we provide a brief overview of the *JaCa* platform – which we consider part of the background of this paper; then, in Section III we introduce and discuss the application of *JaCa* for the development of smart mobile applications on top of Android, and in Section IV we describe two practical application samples useful to evaluate the approach. Finally, in Section V we briefly discuss some open issues related to *JaCa-Android* and, more generally, to the use of current agent-oriented programming technologies for developing applications and related future works.

II. AGENT-ORIENTED PROGRAMMING FOR MAINSTREAM APPLICATION DEVELOPMENT – THE JACA APPROACH

An application in *JaCa* is designed and programmed as a set of agents which work and cooperate inside a common environment. Programming the application means then programming the agents on the one side, encapsulating the

logic of control of the tasks that must be executed, and the environment on the other side, as a first-class abstraction providing the actions and functionalities exploited by the agents to do their tasks. It is worth remarking that this is an *endogenous* notion of environment, i.e. the environment here is part of the software system to be developed [18].

More specifically, in JaCa *Jason* [5] is adopted as programming language to implement and execute the agents and CArtaGo [17] as the framework to program and execute the environments.

Being a concrete implementation of an extended version of AgentSpeak(L) [15], *Jason* adopts a BDI (Belief-Desire-Intention)-based computational model and architecture to define the structure and behaviour of individual agents. In that, agents are implemented as reactive planning systems: they run continuously, reacting to events (e.g., perceived changes in the environment) by executing plans given by the programmer. Plans are courses of actions that agents commit to execute so as to achieve their goals. The pro-active behaviour of agents is possible through the notion of goals (desired states of the world) that are also part of the language in which plans are written. Besides interacting with the environment, *Jason* agents can communicate by means of speech acts.

On the environment side, CArtaGo – following the A&A meta-model [14], [19] – adopts the notion of *artifact* as first-class abstraction to define the structure and behaviour of environments and the notion of *workspace* as a logical container of agents and artifacts. Artifacts explicitly represent the environment resources and tools that agents may dynamically instantiate, share and use, encapsulating functionalities designed by the environment programmer. In order to be used by the agents, each artifact provides a usage interface composed by a set of *operations* and *observable properties*. Operations correspond to the actions that the artifact makes it available to agents to interact with such a piece of the environment; observable properties define the observable state of the artifact, which is represented by a set of information items whose value (and value change) can be perceived by agents as percepts. Besides observable properties, the execution of operations can generate signals perceivable by agents as percepts, too. As a principle of composability, artifacts can be assembled together by a link mechanism, which allows for an artifact to execute operations over another artifact. CArtaGo provides a Java-based API to program the types of artifacts that can be instantiated and used by agents at runtime, and then an object-oriented data-model for defining the data structures used in actions, observable properties and events.

The notion of workspace is used to define the topology of complex environments, that can be organised as multiple sub-environments, possibly distributed over the network. By default, each workspace contains a predefined set of artifacts created at boot time, providing basic actions to manage the overall set of artifacts (for instance, to create, lookup, link together artifacts), to join multiple workspaces, to print messages on the console, and so on.

JaCa integrates *Jason* and CArtaGo so as to make the

use of artifact-based environments by *Jason* agents seamless. To this purpose, first, the overall set of external actions that a *Jason* agent can perform is determined by the overall set of artifacts that are actually available in the workspaces where the agent is working. So, the action repertoire is dynamic and can be changed by agents themselves by creating, disposing artifacts. Then, the overall set of percepts that a *Jason* agent can observe is given by the observable properties and observable events of the artifacts available in the workspace at runtime. Actually an agent can explicitly select which artifacts to observe, by means of a specific action called *focus*. By observing an artifact, artifacts' observable properties are directly mapped into beliefs in the belief-base, updated automatically each time the observable properties change their value. So a *Jason* agent can specify plans reacting to changes to beliefs that concern observable properties or can select plans according to the value of beliefs which refer to observable properties. Artifacts' signals instead are not mapped into the belief-base, but processed as non persistent percepts possibly triggering plans—like in the case of message receipt events. Finally, the *Jason* data-model – essentially based on Prolog terms – is extended to manage also (Java) objects, so as to work with data exchanged by performing actions and processing percepts.

A full description of *Jason* language/platform and CArtaGo framework – and their integration – is out of the scope of this paper: the interested reader can find details in literature [17], [16] and on *Jason* and CArtaGo open-source web sites¹².

III. PROGRAMMING SMART MOBILE APPLICATIONS WITH JACA

In this section we describe how JaCa's features can be effectively exploited to program smart mobile applications, providing benefits over existing non-agent approaches. First, we briefly sketch some of the complexities related to the design and programming of such a kind of applications; then, we describe how these are addressed in JaCa-Android—which is the porting of JaCa on Android, extended with a predefined set of artifacts specifically designed for exploiting Android functionalities.

A. Programming Mobile Applications: Complexities

Mobile systems and mobile applications have gained a lot of importance and magnitude both in research and industry over the last years. This is mainly due to the introduction of mobile devices such as the iPhone³ and the most modern Android⁴-based devices that changed radically the concept of smart-phone thanks to: (i) hardware specifications that allow to compare these devices to miniaturised computers, situated – thanks to the use of practically every kind of known connectivity (GPS, WiFi, bluetooth, etc.) – in a computational

¹<http://jason.sourceforge.net>

²<http://cartago.sourceforge.net>

³<http://www.apple.com/it/iphone/>

⁴<http://www.android.com/>

network which is becoming more and more similar to the vision presented by both the Internet of Things and ubiquitous computing, and (ii) the evolution of both the smart-phone O.S. (Apple iOS, Android, Meego⁵) and their related SDK.

These innovations produce a twofold effect: on the one side, they open new perspectives, opportunities and application scenarios for these new mobile devices; on the other side, they introduce new challenges related to the development of the mobile applications, that are continuously increasing their complexity [1], [11]. These applications – due to the introduction of new usage scenarios – must be able to address issues such as concurrency, asynchronous interactions with different kinds of services (Web sites/Services, social-networks, messaging/mail clients, etc.) and must also expose a user-centric behaviour governed by specific context information (geographical position of the device, presence/absence of different kinds of connectivity, events notification such as the reception of an e-mail, etc.).

To cope with these new requirements, Google has developed the Android SDK⁶, which is an object-oriented Java-based framework meant to provide a set of useful abstractions for engineering mobile applications on top of Android-enable mobile devices. Among the main coarse-grain components introduced by the framework to ease the application development we mention here:

- **Activities:** an activity provides a GUI for one focused endeavor the user can undertake. For example, an activity might present a list of menu items users can choose, list of contacts to send messages to, etc.
- **Services:** a service does not have a GUI and runs in the background for an indefinite period of time. For example, a service might play background music as the user attends to other matters.
- **Broadcast Receiver:** a broadcast receiver is a component that does nothing but receive and react to broadcast announcements. Many broadcasts originate in system code - for example, announcements that the timezone has changed, that the battery is low, etc.
- **Content providers:** a content provider makes a specific set of the application's data available to other applications. The data can be stored in the file system, in an SQLite database, etc.

In Android, interactions among components are managed using a messaging facility based on the concepts of Intent and IntentFilter. An application can request the execution of a particular operation – that could be offered by another application or component – simply providing to the O.S. an Intent properly characterised with the information related to that operation. So, for example, if an application needs to display a particular Web page, it expresses this need creating a proper Intent instance, and then sending this instance to the Android operating system. The O.S. will handle this request locating a proper component – e.g. a browser – able to manage

that particular Intent. The set of Intents manageable by a component are defined by specifying, for that component, a proper set of IntentFilters.

Generally speaking, these components and the Intent-based interaction model are useful – indeed – to organise and structure applications; however – being the framework fully based on an object-oriented virtual machine and language such as Java – they do not shield programmers from using callbacks, threads, and low-level synchronisation mechanisms as soon as applications with complex reactive behaviour are considered. For instance, in classic Android applications asynchronous interactions with external resources are still managed using polling loops or some sort of mailbox; context-dependent behaviour must be realised staining the source code with a multitude of if statements; concurrency issues must be addressed using Java low-level synchronisation mechanisms. So, more generally, we run into the problems that typically arise in mainstream programming languages when integrating one or multiple threads of control with the management of asynchronous events, typically done by callbacks.

In the next section we discuss how agent-oriented programming and, in particular, the JaCa programming model, are effective to tackle these issues at a higher-level of abstraction, making it possible to create more clear, terse and extensible programs.

B. An Agent-oriented Approach based on JaCa

By adopting the JaCa programming model, a mobile Android application can be realised as a workspace in which *Jason* agents are used to encapsulate the logic and the control of tasks involved by the mobile application, and artifacts are used as tools for agents to seamlessly exploit available Android device/platform components and services.

From a conceptual viewpoint, this approach makes it possible to keep the same level of abstraction – based on agent-oriented concepts – both when designing the application and when concretely implementing it using *Jason* and *CARTAGO*. In this way we are able to provide developers a uniform guideline – without conceptual gaps between the abstractions used in the analysis and implementation phases – that drives the whole engineering process of a mobile application.

From a programming point of view, the agent paradigm makes it possible to tackle quite straightforwardly some of the main challenges mentioned in previous sections, in particular: (i) task and activity oriented behaviours can be directly mapped onto agents, possibly choosing different kinds of concurrent architectures according to the needs—either using multiple agents to concurrently execute tasks, or using a single agent to manage the interleaved execution of multiple tasks; (ii) agents' capability of adapting the behaviour on the basis of the current context information can be effectively exploited to realise context-sensitive and context-dependent applications; (iii) asynchronous interactions can be managed by properly specifying the agents' reactive behaviour in relation to the reception of particular percepts (e.g. the reception of a new e-mail).

⁵<http://meego.com>

⁶<http://developer.android.com/sdk/index.html>

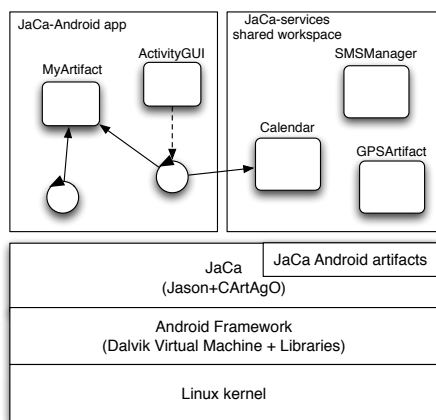


Fig. 1. Abstract representation of the JaCa-Android platform – with in evidence the different agent technologies on which the platform is based – and of generic applications running on top of it.

To see this in practice, we developed a porting of JaCa on top of Android – referred as JaCa-Android – available as open-source project⁷. Fig. 1 shows an abstract representation of the levels characterising the JaCa-Android platform and of a generic applications running on top of it.

The platform includes a set of predefined types of artifacts (BroadcastReceiverArtifact, ActivityArtifact, ContentProviderArtifact, ServiceArtifact) specifically designed to build compliant Android components. So, standard Android components become fully-fledged artifacts that agents and agent developers can exploit without worrying and knowing about infrastructural issues related to the Android SDK. This makes it possible for developers to conceive and realise mobile applications that are seamlessly integrated with the Android SDK, possibly interacting/re-using every component and application developed using the standard SDK. This integration is fundamental in order to guarantee to developers the re-use of existing legacy – i.e. the standard Android components and applications – and for avoiding the development of the entire set of functionalities required by an application from scratch.

Besides, the platform also provides a set of artifacts that encapsulate some of the most common functionalities used in the context of smart mobile applications. In detail these artifacts are:

- SMSManager/MailManager, managing sms/mail-related functionalities (send and receive sms/mail, retrieve stored sms/mail, etc.).
- GPSManager, managing gps-related functionalities (e.g. geolocalisation of the device).
- CallManager, providing functionalities for handling – answer/reject – phone calls.
- ConnectivityManager, managing the access to the different kinds of connectivity supported by the mobile device.

- CalendarArtifact, providing functionalities for managing the built-in Google calendar.

These artifacts, being general purpose, are situated in a workspace called `jaca-services` (see Fig. 1) which is shared by all the JaCa-Android applications—being stored and executed into a proper Android service installed with the JaCa-Android platform. More generally, any JaCa-Android workspace can be shared among different applications—promoting, then, the modularisation and the reuse of the functionalities provided by JaCa-Android applications.

In next section we discuss more in detail the benefits of the JaCa programming model for implementing smart mobile applications by considering two application samples that have been developed on top of JaCa-Android.

IV. EVALUATION THROUGH PRACTICAL EXAMPLES

The first example aims at showing how the approach allows for easily realising context-sensitive mobile applications. For this purpose, we consider a JaCa-Android application inspired to Locale⁸, one of the most famous Android applications and also one of the winners of the first Android Developer Contest⁹. This application (JaCa-Locale) can be considered as a sort of intelligent smart-phone manager realised using a simple *Jason* agent. The agent during its execution uses some of the built-in JaCa-Android artifacts described in Section III and two application-specific artifacts: a PhoneSettingsManager artifact used for managing the device ringtone/vibration and the ContactManager used for managing the list of contacts stored into the smart-phone (this list is an observable property of the artifact, so directly mapped into agents beliefs). The agent manages the smart-phone behaviour discriminating the execution of its plans on the basis of a comparison among its actual context information and a set of user preferences that are specified into the agent’s plans contexts. TABLE I reports a snippet of the *Jason* agent used in JaCa-Locale, in particular the plans shown in TABLE I are the ones responsible of the context-dependent management of the incoming phone calls.

The behaviour of the agent, once completed the initialisation phase (lines 00-07), is governed by a set of reactive plans. The first two plans (lines 9-15) are used for regulating the ringtone level and the vibration for the incoming calls on the basis of the notifications provided by the CalendarArtifact about the start or the end of an event stored into the user calendar. Instead, the behaviour related to the handling of the incoming calls is managed by the two reactive plans `incoming_call` (lines 17-28). The first one (lines 17-19) is applicable when a new incoming call arrives and the phone owner is not busy, or when the incoming call is considered critical. In this case the agent normally handles the incoming call – the ringtone/vibration settings have already been regulated by the plans at lines 9-15 – using the `handle_call` operation provided by the CallManager artifact. The second plan instead

⁷<http://jaca-android.sourceforge.net>

⁸<http://www.twofortyfouram.com/>

⁹<http://code.google.com/intl/it-IT/android/ad/>

```

00 !init.
01
02 +!init
03   <- focus("SMSManager"); focus("MailManager");
04   focus("CallManager"); focus("ContactManager");
05   focus("CalendarArtifact");
06   focus("PhoneSettingsManager");
07   focus("ConnectivityManager").
08
09 +cal_event_start(EventInfo) : true
10   <- set_ringtone_volume(0);
11   set_vibration(on).
12
13 +cal_event_end(EventInfo) : true
14   <- set_ringtone_volume(100);
15   set_vibration(off).
16
17 +incoming_call(Contact, TimeStamp)
18   : not_busy(TimeStamp) | is_call_critical(Contact)
19   <- handle_call.
20
21 +incoming_call(Contact, TimeStamp)
22   : busy(TimeStamp) & not is_call_critical(Contact)
23   <- get_event_description(TimeStamp,EventDescription);
24   drop_call;
25   .concat("Sorry, I'm busy due
26     to", EventDescription, "I will call you back
27     as soon as possible.", OutStr);
28   !handle_auto_reply(OutStr).
29
30 +!handle_auto_reply(Reason) : wifi_status(on)
31   <- send_mail("Auto-reply", Reason).
32
33 +!handle_auto_reply(Reason) : wifi_status(off)
34   <- send_sms(Reason).

```

TABLE I
SOURCE CODE SNIPPET OF THE JACA-LOCALE *Jason* AGENT

(lines 21-28) is applicable when the user is busy and the call does not come from a relevant contact. In this case the phone call is automatically rejected using the `drop_call` operation of the `CallManager` artifact (line 24), and an auto-reply message containing the motivation of the user unavailability is sent back to the contact that performed the call. This notification is sent – using one of the `handle_auto_reply` plans (lines 30-34) – via sms or via mail (using respectively the `SMSManager` or the `MailManager`) depending on the current availability of the WiFi connection on the mobile device (availability checked using the `wifi_status` observable property of the `ConnectivityManager`). It is worth remarking that `busy` and `is_call_critical` refer to rules – not reported in the source code – used for checking respectively: (i) if the phone owner is busy – by checking the belief related to one of the `CalendarArtifact` observable properties (`current_app`) – or (ii) if the call is critical – by checking if the call comes from one of the contact in the `ContactManager` list considered critical: e.g. the user boss/wife.

Generalising the example, context-sensitive applications can be designed and programmed in terms of one or more agents with proper plans that are executed only when the specific context conditions hold.

The example is useful also for highlighting the benefits introduced by artifact-based endogenous environments: (i) it makes it possible to represent and exploit platform/device functionalities at an agent level of abstractions – so in terms of actions and perceptions, modularised into artifacts; (ii) it provides a strong *separation of concerns*, in that developers

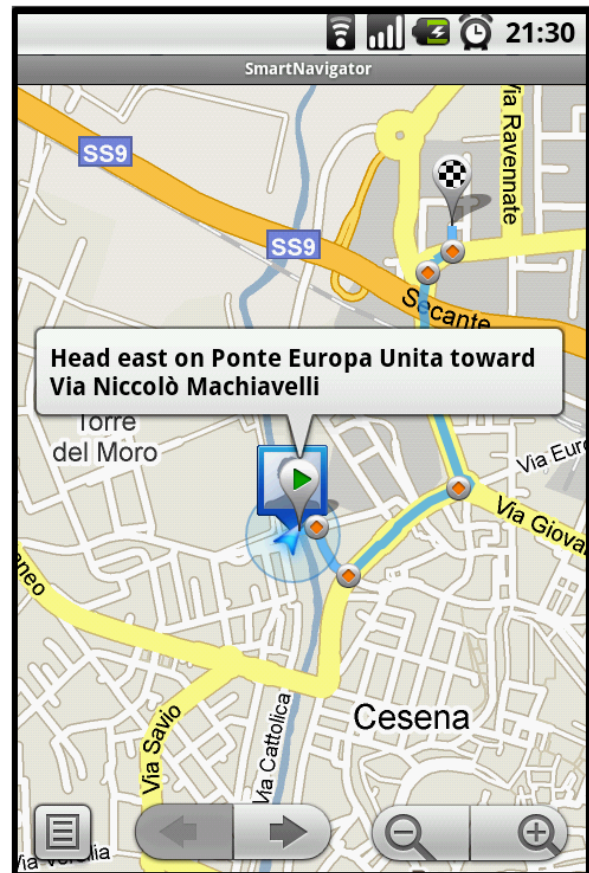


Fig. 2. Screenshot of the SmartNavigator application that integrate in its GUI some of the Google Maps components for showing: (i) the user current position, (ii) the road directions, and (iii) the route to the designed destination.

can fully separate the code that defines the control logic of the application (into agents) from the reusable functionalities (embedded into artifacts) that are needed by the application, making agents' source code more dry.

The second application sample – called `SmartNavigator` (see Fig. 2 for a screenshot) – aims at showing the effectiveness of the approach in managing asynchronous interactions with external resources, such as – for example – Web Services. This application is a sort of smart navigator able to assist the user during his trips in an intelligent way, taking into account the current traffic conditions.

The application is realised using a single *Jason* agent and four different artifacts: (i) the `GPSManager` used for the smart-phone geolocalisation, (ii) the `GoogleMapsArtifact`, an artifact specifically developed for this application, used for encapsulating the functionalities provided by Google Maps (e.g. calculate a route, show points of interest on a map, etc.), (iii) the `SmartNavigatorGUI`, an artifact developed on the basis of the `ActivityArtifact` and some other Google Maps components, used for realizing the GUI of the application and (iv) an artifact, `TrafficConditionsNotifier`, used for managing the interactions with a Web site¹⁰ that

¹⁰<http://www.stradeanas.it/traffico/>

provides real-time traffic information.

TABLE II shows a snippet of the agent source code. The agent main goal `assist_user_trips` is managed by a set of reactive plans that are structured in a hierarchy of sub-goals – handled by a set of proper sub-plans. The agent has a set of initial beliefs (lines 00-01) and an initial plan (lines 5-9) that manages the initialisation of the artifacts that will be used by the agent during its execution. The first plan, reported at lines 11-12, is executed after the reception of an event related to the modification of the `SmartNavigatorGUI` route observable property – a property that contains both the starting and arriving locations provided in input by the user. The handling of this event is managed by the `handle_navigation` plan that: (i) retrieves (line 15) and updates the appropriate agent beliefs (line 16 and 19), (ii) computes the route using an operation provided by the `GoogleMapsArtifact` (`calculate_route` lines 17-18), (iii) makes the subscription – for the route of interest – to the Web site that provides the traffic information using the `TrafficConditionsNotifier` (lines 20-21), and finally (iv) updates the map showed by the application (using the `SmartNavigatorGUI` operations `set_current_position` and `update_map`, lines 22-23) with both the current position of the mobile device (provided by the observable property `current_position` of the `GPSManager`) and the new route.

In the case that no meaningful changes occur in the traffic conditions and the user strictly follows the indications provided by the `SmartNavigator`, the map displayed in the application GUI will be updated, until arriving to the designed destination, simply moving the current position of the mobile device using the plan reported at lines 34-38. This plan, activated by a change of the observable property `current_position`, simply considers (using the sub-plan `check_position_consistency` instantiated at line 36, not reported for simplicity) if the new device position is consistent with the current route (retrieved from the agent beliefs at line 35) before updating the map with the new geolocation information (line 37-38). In the case in which the new position is not consistent – i.e. the user chose the wrong direction – the sub-plan `check_position_consistency` fails. This fail is handled by a proper *Jason* failure handling plan (lines 40-42) that simply re-instantiate the `handle_navigation` plan for computing a new route able to bring the user to the desired destination from his current position (that was not considered in the previous route).

Finally, the `new_traffic_info` plan (lines 25-32) is worth of particular attention. This is the reactive plan that manages the reception of the updates related to the traffic conditions. If the new information are considered relevant with respect to the user preferences (sub-plan `check_info_relevance` instantiated at line 28 and not shown) then, on the basis of this information, the current route (sub-plan `update_route` instantiated at lines 29-30), the Web site subscription (sub-plan `update_subscription` instantiated at line 31), and finally the map displayed on the GUI (line 32) are updated.

So, this example shows how it is possible to integrate the

```

00 preferences(...).
01 relevance_range(10).
02
03 !assist_user_trips.
04
05 +!assist_user_trips
06   <- focus("GPSManager");
07     focus("GoogleMapsArtifact");
08     focus("SmartNavigatorGUI");
09     focus("TrafficConditionsNotifier").
10
11 +route(StartLocation, EndLocation)
12   <- !handle_navigation(StartLocation, EndLocation).
13
14 +!handle_navigation(StartLocation, EndLocation)
15   <- ?relevance_range(Range); ?current_position(Pos);
16     +-leaving(StartLocation); +-arriving(EndLocation);
17     calculate_route(StartLocation,
18                   EndLocation, OutputRoute);
19     +-route(OutputRoute);
20     subscribe_for_traffic_condition(OutputRoute,
21                                   Range);
22     set_current_position(Pos);
23     update_map.
24
25 +new_traffic_info(TrafficInfo)
26   <- ?preferences(Preferences);
27     ?leaving(StartLocation); ?arriving(EndLocation);
28     !check_info_relevance(TrafficInfo, Preferences);
29     !update_route(StartLocation, EndLocation,
30                  TrafficInfo, NewRoute);
31     !update_subscription(NewRoute);
32     update_map.
33
34 +current_position(Pos)
35   <- ?route(Route);
36     !check_position_consistency(Pos, Route);
37     set_current_position(Pos);
38     update_map.
39
40 -!check_position_consistency(Pos, Route)
41   : arriving(EndLocation)
42   <- !handle_navigation(Pos, EndLocation).

```

TABLE II
SOURCE CODE SNIPPET OF THE SMARTNAVIGATOR JASON AGENT

reactive behaviour of a JaCa-Android application – in this example the asynchronous reception of information from a certain source – with its pro-active behaviour — assisting the user during his trips. This integration allows to easily modify and adapt the pro-active behaviour of an application after the reception of new events that can be handled by proper reactive plans: in this example, the reception of the traffic updates can lead the `SmartNavigator` to consider a new route for the trip on the basis of the new information.

V. OPEN ISSUES AND FUTURE WORK

Besides the advantages described in previous section, the application of current agent programming technologies to the development of concrete software systems such as mobile applications have been useful to focus some main weaknesses that these technologies currently have to this end. Here we have identified three general issues that will be subject of future work:

(i) *Devising of a notion of type for agents and artifacts* — current agent programming languages and technologies lack of a notion of type as the one found in mainstream programming languages and this makes the development of large system hard and error-prone. This would make it possible to detect many errors at compile time, allowing for strongly reducing the development time and enhancing the safety of

the developed system. In JaCa we have a notion of type just for artifacts: however it is based on the lower OO layer and so not expressive enough to characterise at a proper level of abstraction the features of environment programming.

(ii) *Improving modularity in agent definition* — this is a main issue already recognised in the literature [7], [8], [9], where constructs such as *capabilities* have been proposed to this end. *Jason* still lacks of a construct to properly modularise and structure the set of plans defining an agent's behaviour — a recent proposal is described here [12].

(iii) *Improving the integration with the OO layer* — To represent data structures, *Jason* — as well as the majority of agent programming languages — adopts Prolog terms, which are very effective to support mechanisms such as unification, but quite weak — from an abstraction and expressiveness point of view — to deal with complex data structures. Main agent frameworks (not languages) in Agent-Oriented Software Engineering contexts — such as Jade [2] or JACK¹¹ — adopt object-oriented data models, typically exploiting the one of existing OO languages (such as Java). By integrating *Jason* with CArtAgO, we introduced a first support to work with an object-oriented data model, in particular to access and create objects that are exchanged as parameters in actions/percepts. However, it is just a first integration level and some important aspects — such as the use of unification with object-oriented data structures — are still not tackled.

Finally, concerning the specific mobile application context, JaCa-Android is just a prototype and indeed needs further development for stressing more in depth the benefits provided by agent-oriented programming compared to mainstream non-agent platforms. Therefore, in future works we aim at improving JaCa-Android in order to tackle some other important features of modern mobile applications such as the smart use of the battery and the efficient management of the computational workload of the device.

VI. CONCLUSION

To conclude, we believe that agent-oriented programming — including multi-agent programming — would provide a suitable level of abstraction for tackling the development of complex software applications, extending traditional programming paradigms such as the Object-Oriented to deal with aspects such as concurrency, reactivity, asynchronous interaction managements, dynamism and so on. In this paper, in particular, we showed the advantages of applying such an approach to the development of smart mobile applications on the Google Android platform, exploiting the JaCa integrated platform. However, we argue that in order to stress and investigate the full value of the agent-oriented approach to this end, further work is needed to extend current agent languages and technologies — or to devise new ones — tackling main aspects that have not been considered so far, being not related to AI but to the principles of software development. This is the core of our current and future work.

¹¹<http://www.agent-software.com>

REFERENCES

- [1] A. Battestini, C. Del Rosso, A. Flanagan, and M. Miettinen. Creating next generation applications and services for mobile devices: Challenges and opportunities. In *EEE 18th Int. Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2007, pages 1–4, 3-7 2007.
- [2] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.
- [3] M. Berger, S. Rusitschka, D. Toropov, M. Watzke, and M. Schlichte. Porting distributed agent-middlewares to small mobile devices. In *AAMAS Workshop on Ubiquitous Agents on Embedded, Wearable and Mobile Devices*.
- [4] R. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications (vol. 1)*. Springer, 2005.
- [5] R. Bordini, J. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, Ltd, 2007.
- [6] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications (vol. 2)*. Springer Berlin / Heidelberg, 2009.
- [7] L. Braubach, A. Pokahr, and W. Lamersdorf. Extending the capability concept for flexible BDI agent modularization. In *Programming Multi-Agent Systems*, volume 3862 of *LNAI*, pages 139–155. Springer, 2005.
- [8] M. Dastani, C. Mol, and B. Steunebrink. Modularity in agent programming languages: An illustration in extended 2APL. In *Proceedings of the 11th Pacific Rim Int. Conference on Multi-Agent Systems (PRIMA 2008)*, volume 5357 of *LNCSS*, pages 139–152. Springer, 2008.
- [9] K. Hindriks. Modules as policy-based intentions: Modular agent programming in GOAL. In *Programming Multi-Agent Systems*, volume 5357 of *LNCSS*, pages 156–171. Springer, 2008.
- [10] F. Koch, J.-J. C. Meyer, F. Dignum, and I. Rahwan. Programming deliberative agents for mobile services: The 3apl-m platform. In *PROMAS*, pages 222–235, 2005.
- [11] B. König-Ries. Challenges in mobile application development. *it - Information Technology*, 51(2):69–71, 2009.
- [12] N. Madden and B. Logan. Modularity and compositionality in Jason. In *Proceedings of Int. Workshop Programming Multi-Agent Systems (ProMAS 2009)*, 2009.
- [13] C. Muldoon, G. M. P. O'Hare, R. W. Collier, and M. J. O'Grady. Agent factory micro edition: A framework for ambient applications. In *Int. Conference on Computational Science (3)*, pages 727–734, 2006.
- [14] A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17 (3), Dec. 2008.
- [15] A. S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In *MAAMAW '96: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away*, pages 42–55, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.
- [16] A. Ricci, M. Piunti, L. D. Acay, R. Bordini, J. Hübner, and M. Dastani. Integrating artifact-based environments with heterogeneous agent-programming platforms. In *Proceedings of 7th International Conference on Agents and Multi Agents Systems (AAMAS08)*, 2008.
- [17] A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment programming in CArtAgO. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications, Vol. 2*, pages 259–288. Springer, 2009.
- [18] A. Ricci, A. Santi, and M. Piunti. Action and perception in multi-agent programming languages: From exogenous to endogenous environments. In *Proceedings of the Int. Workshop on Programming Multi-Agent Systems (ProMAS'10)*, Toronto, Canada, 2010.
- [19] A. Ricci, M. Viroli, and A. Omicini. The A&A programming model & technology for developing agent environments in MAS. In M. Dastani, A. El Fallah Seghrouchni, A. Ricci, and M. Winikoff, editors, *Programming Multi-Agent Systems*, volume 4908 of *LNAI*, pages 91–109. Springer Berlin / Heidelberg, 2007.
- [20] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

Exploiting Agent-Oriented Programming for Building Advanced Web 2.0 Applications

Mattia Minotti
University of Bologna
Cesena, Italy
Email: mattia.minotti@studio.unibo.it

Andrea Santi
DEIS, University of Bologna
Cesena, Italy
Email: a.santi@unibo.it

Alessandro Ricci
DEIS, University of Bologna
Cesena, Italy
Email: a.ricci@unibo.it

Abstract—We believe that agent-oriented programming languages and multi-agent programming technologies provide an effective level of abstraction for tackling the design and programming of mainstream software applications, besides being techniques effective for dealing with (Distributed) Artificial Intelligence problems. In this paper we support this claim in practice by discussing the use of a platform integrating two main agent programming technologies – *Jason* agent programming language and CArtAgO environment programming framework – to the development of Web 2.0 applications. Following the cloud computing perspective, these kinds of applications will more and more replace desktop applications, exploiting the Web infrastructure as a common distributed operating system, raising however challenges that are not effectively tackled – we argue – by mainstream programming paradigms, such as the object-oriented one.

I. INTRODUCTION

The value of Agent-Oriented Programming (AOP) [21] – including Multi-Agent Programming (MAP) – is often remarked and evaluated in the context of Artificial Intelligence (AI) and Distributed AI problems. This is evident, for instance, by considering existing agent programming languages (see [6], [7] for comprehensive surveys) – whose features are typically demonstrated by considering AI toy problems such as block worlds and alike. Besides this view, we argue that the level of abstraction introduced by AOP is effective for organizing and programming software applications in general, starting from those programs that involve aspects related to reactivity, asynchronous interactions, concurrency, up to those involving different degrees of autonomy and intelligence. In that context, an important example is given by Web 2.0 applications, which share more and more features with desktop applications, combining their better user experience with all the benefits provided by the Web, such as distribution, openness and accessibility. Applications of this kind are at the core of the cloud computing vision.

In this paper we show this idea in practice by describing a platform for developing Web 2.0 applications using agent programming technologies, in particular *Jason* for programming agents and CArtAgO for programming the environments where agents work. We refer to the integrated use of *Jason* and CArtAgO as JaCa and its application for building Web 2.0 application as JaCa-Web. Besides describing the platform, our aim here is to discuss the key points that make

JaCa and – more generally – agent-oriented programming a suitable paradigm for tackling main complexities of software applications, advanced Web applications in this case, that – we argue – are not properly addressed by mainstream programming languages, such as object-oriented ones. In that, this work extends a previous one [12] where we adopted a Java-based framework called simpA [20] to this end, replaced in this paper *Jason* so as to exploit the features provided by the Belief-Desire-Intention (BDI) architecture.

The remainder of the paper is organised as follows. First, we provide a brief overview of JaCa (Section II) programming model and platform. Then, we discuss the use of JaCa for developing Web 2.0 applications (Section III), remarking the advantages compared to existing state-of-the art approaches. To evaluate the approach, we describe the design and implementation of a case study (Section IV), and we conclude the paper by discussing related and future work (Section V).

II. AGENT-ORIENTED PROGRAMMING FOR MAINSTREAM APPLICATION DEVELOPMENT – THE JaCa APPROACH

An application in JaCa is designed and programmed as a set of agents which work and cooperate inside a common environment. Programming the application means then programming the agents on the one side, encapsulating the logic of control of the tasks that must be executed, and the environment on the other side, as first-class abstraction providing the actions and functionalities exploited by the agents to do their tasks. More specifically, in JaCa *Jason* [5] is adopted as programming language to implement and execute the agents and CArtAgO [18] as the framework to program and execute the computational environments where agents are situated.

Being a concrete implementation of an extended version of AgentSpeak(L) [16], *Jason* adopts a BDI (Belief-Desire-Intention)-based computational model and architecture to define the structure and behaviour of individual agents. In that, agents are implemented as reactive planning systems: they run continuously, reacting to events (e.g., perceived changes in the environment) by executing plans given by the programmer. Plans are courses of actions that agents commit to execute so as to achieve their goals. The pro-active behaviour of agents is possible through the notion of goals (desired states of the world) that are also part of the language in which plans

are written. Besides interacting with the environment, *Jason* agents can communicate by means of speech acts.

On the environment side, *CARTAgO* – following the A&A meta-model [13], [19] – adopts the notion of *artifact* as first-class abstraction to define the structure and behaviour of such computational environments and the notion of *workspace* as a logical container of agents and artifacts. Artifacts explicitly represent the resources and tools that agents may dynamically instantiate, share and use, encapsulating functionalities designed by the environment programmer. In order to be used by the agents, each artifact provides of a usage interface composed by a set of *operations* and *observable properties*. Operations correspond to the actions that the artifact makes it available to agents to interact with such a piece of the environment; observable properties define the observable state of the artifact, which is represented by a set of information items whose value (and value change) can be perceived by agents as percepts. Besides observable properties, the execution of operations can generate signals perceivable by agents as percepts, too. As a principle of composability, artifacts can be assembled together by a link mechanism, which allows for an artifact to execute operations over another artifact. *CARTAgO* provides a Java-based API to program the types of artifacts that can be instantiated and used by agents at runtime, and then an object-oriented data-model for defining the data structures in actions, observable properties and events.

Finally, the notion of workspace is used to define the topology of complex environments, that can be organised as multiple sub-environments, possibly distributed over the network. By default, each workspace contains a predefined set of artifact created at boot time, providing basic actions to manage the set of artifacts in the workspace – for instance, to create, lookup, link together artifacts – to join multiple workspaces, to print message on the console, and so on.

JaCa integrates *Jason* and *CARTAgO* so as to make the use of artifact-based environments by *Jason* agents seamless. To this purpose, first, the overall set of external actions that a *Jason* agent can perform is determined by the overall set of artifacts that are actually available in the workspaces where the agent is working. So, the action repertoire is dynamic and can be changed by agents themselves by creating, disposing artifacts. Then, the overall set of percepts that a *Jason* agent can observe is given by the observable properties and observable events of the artifacts available in the workspace at runtime. Actually an agent can explicitly select which artifacts to observe, by means of a specific action called *focus*. By observing an artifact, artifacts' observable properties are directly mapped into beliefs in the belief-base, updated automatically each time the observable property changes its value. So a *Jason* agent can specify plans reacting to changes to beliefs that concern observable properties or can select plan according to the value of beliefs which refer to observable properties. Artifacts' signals instead are not mapped into the belief base, but processed as non persistent percepts possibly triggering plans—like in the case of message receipt events. Finally, the *Jason* data-model – essentially based on Prolog terms – is

extended to manage also (Java) objects, so as to work with data exchanged by performing actions and processing percepts.

A full description of *Jason* language/platform and *CARTAgO* framework – and their integration – is out of the scope of this paper: the interested reader can find details in literature [18], [17] and on *Jason* and *CARTAgO* open-source web sites¹².

III. PROGRAMMING WEB 2.0 APPLICATIONS WITH *JaCa*

In this section, we describe how the features of *JaCa* can be exploited to program complex Web 2.0 applications, providing benefits over existing approaches. First, we sketch the main complexities related to the design and programming of modern and future web applications; then we describe how these are addressed by *JaCa-Web*, which is a framework on top of *JaCa* to develop such a kind of applications.

A. Programming Future Web Applications: Complexities

Due to network speed problems overcoming and machine computational power increasing, the client-side of so-called *rich web applications* is constantly evolving in terms of complexity. Web 2.0 applications share more and more features with desktop applications in order to combine their better user experience with all Web benefits, such as distribution, openness and accessibility. One of the most important features of Web 2.0 is a new interaction model between the client user interface of a Web browser and the server-side of the application. Such rich Web applications allow the client to send multiple concurrent requests in an asynchronous way, avoiding complete page reload and keeping the user interface live and responding. Periodic activities within the client-side of the applications can be performed in the same fashion, with clear advantages in terms of perceived performance, efficiency and interactivity.

So the more complex web apps are considered, the more the application logic put on the client side becomes richer, eventually including asynchronous interactions – with the user, with remote services – and possibly also concurrency – due to the concurrent interaction with multiple remote services. This situation is exemplified by cloud computing applications, such as Google doc³.

The direction of decentralizing responsibilities to the client is evident also by considering the new HTML standard 5.0, which enriches the set of API and features that can be used by the web application on the client side⁴. Among the others, some can have a strong impact on the way an application is designed: it is the case of the Web Worker mechanism⁵, which makes it possible to spawn background workers running scripts in parallel to their main page, allowing for thread-like operation with message-passing as coordination mechanism.

¹<http://jason.sourceforge.net>

²<http://cartago.sourceforge.net>

³<http://docs.google.com>

⁴<http://dev.w3.org/html5/spec/>

⁵<http://www.whatwg.org/specs/web-workers/current-work/>

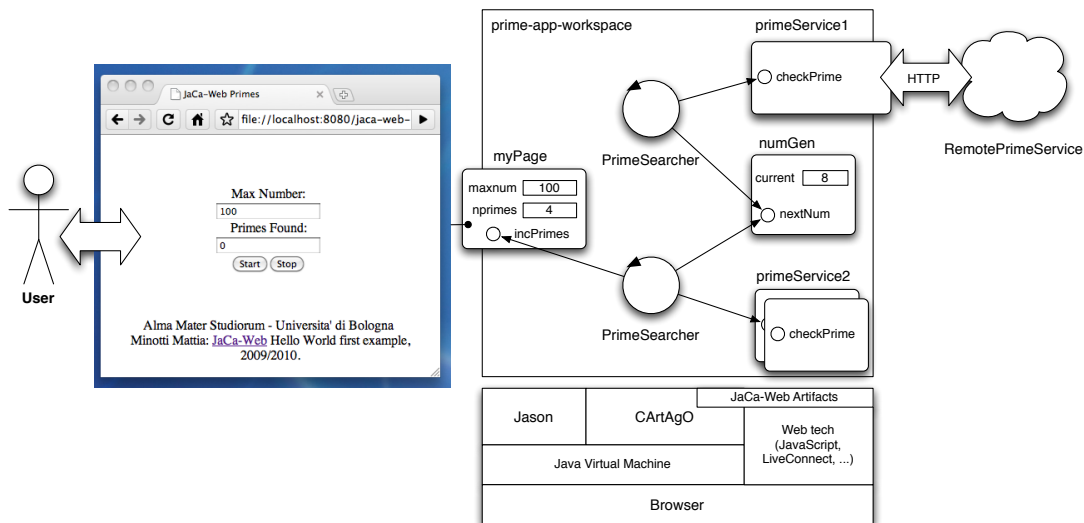


Fig. 1. An abstract overview of a JaCa-Web application, referring in particular to the toy example described in the paper. In evidence: (Top) the workspace with the agents (circles) and artifacts (rounded square); among the artifacts, *myPage* and *primeService1* enable and rule the interaction with the external environment sources, namely the human user and the remote HTTP service; (Bottom) the layers composing the JaCa-Web platform, which includes – on top of the Java Virtual Machine and browser/web infrastructure – *Jason* and *CArtAgO* sub-system and then a pre-defined library of artifacts (JaCa-Web artifacts) specifically designed for the Web context.

Another one is cross-document messaging⁶, which defines mechanisms for communicating between browsing contexts in HTML documents.

Besides devising enabling mechanisms, a main issue is then how to design and program applications of this kind [11].

A basic and standard way to realise the client side of web app is to embed in the page scripts written in some scripting language – such as JavaScript. Originally such scripts were meant just to perform check on the inputs and to create visual effects. The problem is that scripting languages – like JavaScript – have not been designed for programming in the large, so using them to organize, design, implements complex programs is hard and error-prone.

To address the problems related to scripting languages, higher-level approaches have been proposed, based on frameworks that exploit mainstream object-oriented programming languages. A main example is Google Web Toolkit (GWT)⁷, which allows for developing client-side apps with Java. This choice makes it possible to reuse and exploit all the strength of mainstream programming-in-the-large languages that are typically not provided by scripting languages—an example is strong typing. However it does not provide significant improvement for those aspects that are still an issue for OO programming languages, such as concurrency, asynchronous events and interactions, and so on.

We argue then that these aspects can be effectively captured by adopting an agent-oriented level of abstraction and programmed by exploiting agent-oriented technologies such as JaCa: in next section we discuss this point in detail.

B. An Agent-Oriented Programming Approach based on JaCa

By exploiting JaCa, we directly program the Web 2.0 application as a normal JaCa agent program, composed by a workspace with one or multiple agents working within an artifact-based environment including a set of pre-defined type of artifacts specifically designed for the Web context domain (see Fig. 1). Generally speaking, agents are used to encapsulate the logic of control and execution of the tasks that characterise the Web 2.0 app, while artifacts are used to implement the environment needed for executing the tasks, including those coordination artifacts that can ease the coordination of the agents' work. As soon as the page is downloaded by the browser, the application is launched – creating the workspace, the initial set of agents and artifacts.

Among the pre-defined types of artifact available in the workspace, two main ones are the *Page* artifact and the *HTTPService* artifact. *Page* provides a twofold functionality to agents: (i) to access and change the web page, internally exploiting specific low-level technology to work with the DOM (Document Object Model) object, allowing for dynamically updating its content, structure, and visualisation style; (ii) to make events related to user's actions on the page observable to agents as percepts. An application may either exploit directly *Page* or define its own extension with specific operations and observable properties linked to the specific content of the page. *HTTPService* provides basic functionalities to interact with a remote HTTP service, exploiting and hiding the use of sockets and low-level mechanisms. Analogously to *Page*, this kind of artifact can be used as it is – providing actions to do HTTP requests – or can be extended providing an higher-level application specific usage interface hiding the HTTP level.

To exemplify the description of these elements and of JaCa-

⁶<http://dev.w3.org/html5/postmsg/>

⁷<http://code.google.com/webtoolkit/>

Web programming in the overall, in the following we consider a toy example of Web 2.0 app, in which two agents are used to search for prime numbers up to a maximum value which can be specified and dynamically changed by the user through the web page. As soon as an agent finds a new prime number, a field on the web page reporting the total number of values is updated.

The environment (shown in Fig. 1) includes – besides the artifact representing the page, called here `myPage` – an artifact called `numGen`, functioning as a number generator, shared and used by agents to get the numbers to verify, and two artifacts, `primeService1` and `primeService2`, providing the (same) functionality that is verifying if a number is prime.

`myPage` is an instance of `MyPage` extending the basic *Page* artifact so as to be application specific, by: (i) defining an observable property `maxnum` whose value is directly linked to the related input field on the web page; (ii) generating `start` and `stop` signals as soon as the page button controls `start` and `stop` are pressed; (iii) defining an operation `incPrimes` that updates the output field of the page reporting the actual number of prime numbers found.

`numGen` is an instance of the `NumGen` artifact (see Fig. 3), which provides an action `getNextNum` to generate a new number – retrieved as output (i.e. action feedback) parameter.

The two prime number service artifacts provide the same usage interface, composed by a `checkPrime(num: integer)` action, which generates an observable event `is_prime(num: integer)` if the number is found to be prime. One artifact does the computation locally (`LocalPrimeService`); the other one, instead – which is an instance of `RemotePrimeService`, extending the pre-defined *HTTPService* artifact – provides the functionality by interacting with a remote HTTP service.

Fig. 2 shows the source code of one of the two agents. After having set up the tools needed to work, the agent waits to perceive a `start` event generated by the page. Then, it starts working, repeatedly getting a new number to check – by executing a `getNextNum` – until the maximum number is achieved. The agent knows such a maximum value by means of the `maxnum` page observable property—which is mapped onto the related agent belief. The agent checks the number by performing the action `checkPrime` and then reacting to `is_prime(Num: integer)` event, updating the page by performing `incPrimes`. If a `stop` event is perceived – which means that the user pressed the stop button on the Web page – the agent promptly reacts and stops working, dropping its main intention.

A final note about implementation: Java applet technology is used to run the full application stack (including *Jason* and *CARTAgO*) in the browser, using signed applets so to avoid limitations imposed by the sandbox model. LiveConnect technology⁸ is exploited to enable a bi-direction interaction between the applet and the web page resources (DOM, scripts).

⁸<https://jdk6.dev.java.net/plugin2/liveconnect/>

```

00 !setup.
01
02 +!setup
03   <- focusByName("MyPage");
04     makeArtifact("primeService1",
05               "RemotePrimeService");
06     makeArtifact("numGen", "NumGen").
07
08 +start
09   <- focusByName("primeService1");
10     focusByName("numGen");
11     !!checkPrimes.
12
13 +!checkPrimes
14   <- nextNum(Num);
15     !checkNum(Num).
16
17 +!checkNum(Num): maxnum(Max) & Num <= Max
18   <- checkPrime(Num);
19     !checkPrimes.
20
21 +!checkNum(Num) <- maxnum(Max) & Num > Max.
22
23 +!is_prime(Num) <- incPrimes.
24
25 +stop <- .drop_intention(checkPrimes).

```

Fig. 2. *Jason* source code of a prime searcher agent.

```

public class MyPage extends PageArtifact {
    protected void setup() {
        defineObsProperty("maxnum", getMaxValue());
        //Operation for event propagation
        linkEventToOp("start", "click", "startClicked");
        linkEventToOp("stop", "click", "stopClicked");
        linkEventToOp("maxnum", "change", "maxnumChange");
    }
    @OPERATION void incPrimes(){
        Elem el = getElementById("primes_found");
        el.setValue(el.intValue()+1);
    }
    @INTERNAL_OPERATION private void startClicked(){
        signal("start");
    }
    @INTERNAL_OPERATION private void stopClicked(){
        signal("stop");
    }
    @INTERNAL_OPERATION private void maxnumChange(){
        updateObsProperty("maxnum", getMaxValue());
    }
    private int getMaxValue(){
        return getElementById("maxnum").intValue();
    }
}

public class RemotePrimeService extends HTTPService {
    @OPERATION void checkPrime(double n){
        HTTPResponse res =
            doHTTPRequest(serverAddr, "isPrime", n);
        if (res.getElem("is_prime").equals("true")){
            signal("is_prime", n);
        }
    }
}

public class NumGen extends Artifact {
    void init(){ defineObsProperty("current", 0); }

    @OPERATION void nextNum(OpFeedbackParam<Integer> res){
        int v = getObsProperty("current").intValue();
        updateObsProperty("current", ++v);
        res.set(v);
    }
}

```

Fig. 3. Artifacts' definition in *CARTAgO*: `MyPage` and `RemotePrimeService` extending respectively `PageArtifact` and `HTTPService` artifact types which are available by default in *JaCa-Web* workspaces, and `NumGen` to coordinate number generation and sharing.

C. Key points

We have identified three key points that, in our opinion, represent main benefits of adopting agent-oriented programming and, in particular, the JaCa-Web programming model, for developing applications of this kind.

First, agents are first-class abstractions for mapping possibly concurrent tasks identified at the design level, so reducing the gap from design to implementation. The approach allows for choosing the more appropriate concurrent architecture, allocating more tasks to the same kind of agent or defining multiple kind of agents working concurrently. This allows for easily programming Web 2.0 concurrent applications, that are able to exploit parallel hardware on the client side (such as multi-core architectures). In the example, two agents are used to fairly divide the overall job and work concurrently, exploiting the number generator artifact as a coordination tool to share the sequence of numbers. Actually, changing the solution by using a single agent or more than two agents would not require any substantial change in the code.

A second key point provided by the agent control architecture is the capability of defining task-oriented computational behaviours that straightforwardly integrate the management of asynchronous events generated by the environment – such as the input of the user or the responses retrieved from remote services – and the management of workflows of possibly articulated activities, which can be organized and structured in plans and sub-plans. This makes it possible to avoid the typical problems produced by the use of callbacks – that can be referred as *asynchronous spaghetti code* – to manage events within programs that need – at the same time – to have one or multiple threads of control.

In the prime searcher agent shown in the example, for instance, on the one hand we use a plan handling the `checkPrimes` goal to pro-actively search for prime numbers. The plan is structured then into a subgoal `checkNum` to process the number retrieved by interacting with the number generator. Then, the plan executed to handle this subgoal depends on the dynamic condition of the system: if the number to process is greater than the current value of the `maxnum` page observable property (i.e. of its related agent belief), then no checks are done and the goal is achieved; otherwise, the number is checked by exploiting a prime service available in the environment and the a new `checkPrimes` goal is issued to go on exploring the rest of the numbers. The user can dynamically change the value of the maximum number to explore, and this is promptly perceived by the agents which can change then their course of actions accordingly. On the other hand, reactive plans are used to process asynchronous events from the environment, in particular to process incoming results from prime services (line 22) and user input to stop the research (line 24).

Finally, the third aspect concerns the strong separation of concerns which is obtained by exploiting the environment as first class abstraction. *Jason* agents, on the one side, encapsulates solely the logic and control of tasks execution; on the

other side, basic low-level mechanisms to interact and exploit the Web infrastructure are wrapped inside artifacts, whose functionalities are seamlessly exploited by agents in terms of actions (operations) and percepts (observable properties and events). Also, application specific artifacts – such as `NumGen` – can be designed to both encapsulate shared data structures useful for agents’ work and regulate their access by agents, functioning as a coordination mechanism.

IV. A CASE STUDY

To stress the features of agent-oriented programming and test-drive the capabilities of the JaCa-Web framework, we developed a real-world Web application – with features that go beyond the ones that are typically found in current Web 2.0 app. The application is about searching products and comparing prices from multiple services, a “classic” problem on the Web.

We imagine the existence of N services that offer product lists with features and prices, codified in some standard machine-readable format. The client-side in the Web application needs to search all services for a product that satisfies a set of user-defined parameters and has a price inferior to a certain user-defined threshold. The client also needs to periodically monitor services so as to search for new offerings of the same product. A new offering satisfying the constraints should be visualised only when its price is more convenient than the currently best price. The client may finish its search and monitoring activities when some user-defined conditions are met—a certain amount of time is elapsed, a product with a price less than a specified threshold is found, or the user interrupts the search with a click on a proper button in the page displayed by the browser. Finally, if such an interruption took place, by pressing another button it must be possible to let the search continue from the point where it was blocked.

Typically applications of this kind are realised by implementing all the features on the server side, without – however – any support for long-term searching and monitoring capabilities. In the following, we describe a solution based on JaCa-Web, in which responsibilities related to the long-term search and comparison are decentralised into the client side of the application, improving then the scalability and quality of service for the users.

A. Application Design

The solution includes two kinds of agents (see Fig. 4): a *UserAssistant* agent – which is responsible of setting up the application environment and manage interaction with the user – and multiple *ProductFinder* agents, which are responsible to periodically interact with remote product services to find the products satisfying the user-defined parameters. To aggregate data retrieved from services and coordinate the activities of the *UserAssistant* and *ProductFinder* we introduce a *ProductDirectory* artifact, while a *MyPage* page artifact and multiple instances of *ProductService* artifacts are used respectively by the *UserAssistant* and *ProductFinder* to interact with the user and with remote product services.

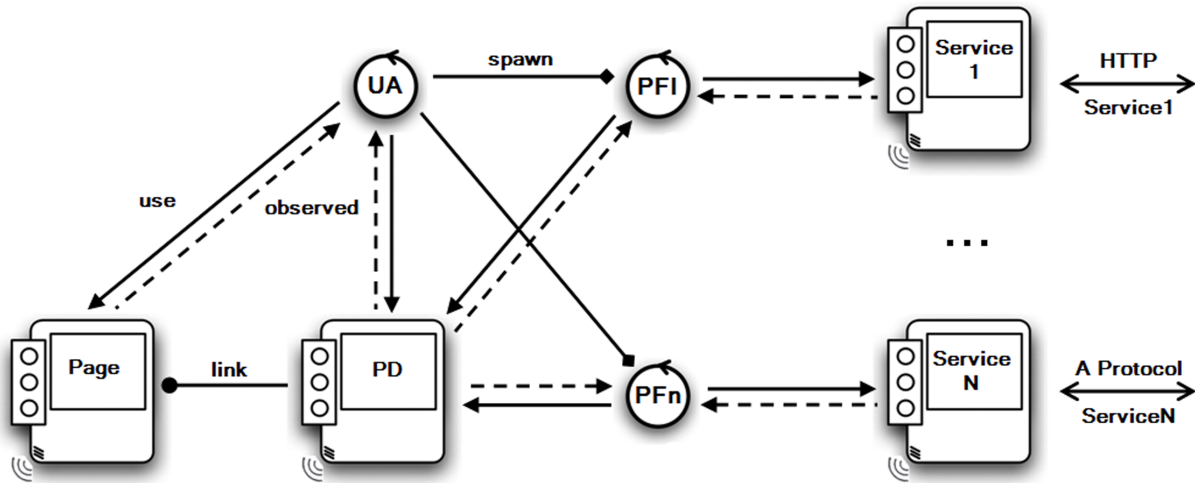


Fig. 4. The architecture of the client-side Web application sample in terms of agent, artifacts, and their interactions. UA is the *UserAgent*, PFs are the *ProductFinder* agents, PD is the *ProductDirectory* artifact and finally Services are the *ProductService* artifacts

More in detail, the *UserAssistant* agent is the first agent booted on the client side, and it setups the application environment by creating the *ProductDirectory* artifact and spawning a number of *ProductFinder* agents, one for each service to monitor. Then, by observing the *MyPage* artifact, the agent monitors user's actions and inputs. In particular, the web page provides controls to start, stop the searching process and to specify and change dynamically the keywords related to the product to search, along with the conditions to possibly terminate the process. Page events are mapped onto start and stop observable events generated by *MyPage*, while specific observable properties – keywords and termination conditions – are used to make it observable the input information specified by the user.

The *UserAssistant* reacts to these observable events and to changes to observable properties, and interacts with *ProductFinder* agents to coordinate the searching process. The interaction is mediated by the *ProductDirectory* artifact, which is used and observed by both the *UserAssistant* and *ProductFinders*. In particular, this artifact provides a usage interface with operations to: (i) dynamically update the state and modality of the searching process – in particular `startSearch` and `stopSearch` to change the value of a `searchState` observable property – useful to coordinate agents' work – and `changeBasePrice`, `changeKeywords` to change the value of the base price and the keywords describing the product, which are stored in a keyword observable property; (ii) aggregate product information found by *ProductFinders* – in particular `addProducts`, `removeProducts`, `clearAllProducts` to respectively add and remove a product, and remove all products found so far. Besides `searchState` and `keywords`, the artifact has further observable properties, `bestProduct`, to store and make it observable the best product found so far.

Finally, each *ProductFinders* periodically interact with a

```
// ProductFinder agent
...
+searchState("start")
  <- focus("service1");
  !!search.

+!search: keywords(Keywords)
  <- requestProducts(Keywords,ProductList);
  !processProducts(ProductList,
    ProductsToAdd,
    ProductsToRemove);
  addProducts(ProductsToAdd);
  removeProducts(ProductsToRemove);
  .wait({+keywords(_)},5000,_);
  !search.

+searchState("stop")
  <- .drop_intention(search).
```

Fig. 5. A snippet of *ProductFinder* agent's plans.

remote product service by means of a private *ProductService* artifact, which extends a *HTTPService* artifact providing an operation (`requestProducts`) to directly perform high-level product-oriented requests, hiding the HTTP level.

B. Implementation

The source code of the application can be consulted on the *JaCa-Web* web site⁹, where the interested reader can find also the address of a running instance that can be used for tests. Here we just report a snippet of the *ProductFinder* agents' source code (Fig. 5), with in evidence (i) the plans used by the agent to react to changes to the search state property perceived from the *ProductDirectory* artifact - adding and removing a new search goal, and (ii) the plan used to achieve that goal, first getting the product list by means of the `requestProducts` operation and then updating the

⁹<http://jacaweb.sourceforge.net>

ProductDirectory accordingly by adding new products and removing products no more available. It is worth noting the use of the keywords belief – related to the keywords observable property of the *ProductDirectory* artifact – in the context condition of the plan to automatically retrieve and exploit updated information about the product to search.

V. RELATED WORKS AND CONCLUSION

To conclude, we believe that agent-oriented programming – including multi-agent programming – would provide a suitable level of abstraction for tackling the development of complex software applications, extending traditional programming paradigms such as the Object-Oriented to deal with aspects such as concurrency, reactivity, asynchronous interaction managements, dynamism and so on. In this paper, in particular, we discussed the advantages of applying such an approach to the development of Web 2.0 advanced applications, exploiting the JaCa integrated platform.

Concerning the specific application domain, several frameworks and bridges have been developed to exploit agent technologies for the development of Web applications. Main examples are the Jadex Webbridge [14], JACK WebBot [2] and the JADE Gateway Agent [1]. The Webbridge Framework enables a seamless integration of the Jadex BDI agent framework [15] with JSP technology, combining the strength of agent-based computing with Web interactions. In particular, the framework extends the the Model 2 architecture – which brings the Model-View-Controller (MVC) pattern in the context of Web application development – to include also agents, replacing the controller with a bridge to an agent application, where agents react to user requests. JACK WebBot is a framework on top of the JACK BDI agent platform which supports the mapping of HTTP requests to JACK event handlers, and the generation of responses in the form of HTML pages. Using WebBot, you can implement a web application which makes use of JACK agents to dynamically generate web pages in response to user input. Finally, the JADE Gateway Agent is a simple interface to connect any Java non-agent application – including Web Applications based on Servlets and JSP – to an agent application running on the JADE platform [3].

All these approaches explore the use of agent technologies on the *server* side of Web Applications, while in our work we focus on the *client* side, which is what characterises Web 2.0 applications. So – roughly speaking – our agents are running not on a Web server, but inside the Web browser, so in a fully decentralized fashion. Indeed, these two views can be combined together so as to frame an agent-based way to conceive next generation Web applications, with agents running on both the client and server side.

Finally, the use of agents to represent concurrent and interoperable computational entities already sets the stage for a possible evolution of Web 2.0 applications into *Semantic Web* applications [4]. From the very beginning [9], research activity on the Semantic Web has always dealt with *intelligent agents* capable of reasoning on machine-readable descriptions of Web resources, adapting their plans to the open Internet

environment in order to reach a user-defined goal, and negotiating, collaborating, and interacting with each other during their activities. So, a main future work accounts for extending the JaCa-Web platform with Semantic Web technologies: to this purpose, existing works such as JASDL [10] and the NUI project [8] will be main references.

REFERENCES

- [1] JADE gateway agent (JADE 4.0 api) – <http://jade.tilab.com/doc/api/jade/wrapper/gateway/jadegateway.html>.
- [2] Agent Oriented Software Pty. JACK intelligent agents webbot manual – http://www.aosgrp.com/documentation/jack/webbot_manual_web/index.html#thejackwebbotarchitecture.
- [3] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.
- [4] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001.
- [5] R. Bordini, J. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, Ltd, 2007.
- [6] R. e. a. Bordini, editor. *Multi-Agent Programming: Languages, Platforms and Applications (vol. 1)*. Springer, 2005.
- [7] R. e. a. Bordini, editor. *Multi-Agent Programming: Languages, Platforms and Applications (vol. 2)*. Springer Berlin / Heidelberg, 2009.
- [8] I. Dickinson. *BDI Agents and the Semantic Web: Developing User-Facing Autonomous Applications*. PhD thesis, University of Liverpool, September 2006.
- [9] J. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30–37, 2001.
- [10] T. Klapiscak and R. H. Bordini. JASDL: A practical programming approach combining agent and semantic web technologies. In *Declarative Agent Languages and Technologies VI*, volume 5397/2009 of *LNCSE*, pages 91–110, Berlin, Heidelberg, 2009. Springer-Verlag.
- [11] T. Mikkonen and A. Taivalsaari. Web applications: spaghetti code for the 21st century. Technical report, Mountain View, CA, USA, 2007.
- [12] M. Minotti, G. Piancastelli, and A. Ricci. An agent-based programming model for developing client-side concurrent web 2.0 applications. In J. Filipe and J. Cordeiro, editors, *Web Information Systems and Technologies*, volume 45 of *Lecture Notes in Business Information Processing*. Springer Berlin Heidelberg, 2010.
- [13] A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17 (3), Dec. 2008.
- [14] A. Pokahr and L. Braubach. The webbridge framework for building web-based agent applications. pages 173–190, 2008.
- [15] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In R. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, editors, *Multi-Agent Programming*. Kluwer, 2005.
- [16] A. S. Rao. Agentspeak(1): BDI agents speak out in a logical computable language. In *MAAMAW '96: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world*, pages 42–55. Springer-Verlag New York, Inc., 1996.
- [17] A. Ricci, M. Piunti, L. D. Acay, R. Bordini, J. Hübner, and M. Dastani. Integrating artifact-based environments with heterogeneous agent-programming platforms. In *Proceedings of 7th International Conference on Agents and Multi Agents Systems (AAMAS08)*, 2008.
- [18] A. Ricci, M. Piunti, M. Viroli, and A. Omicini. Environment programming in CArtAgO. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications, Vol. 2*, pages 259–288. Springer, 2009.
- [19] A. Ricci, M. Viroli, and A. Omicini. The A&A programming model & technology for developing agent environments in MAS. In M. Dastani, A. El Fallah Seghrouchni, A. Ricci, and M. Winikoff, editors, *Programming Multi-Agent Systems*, volume 4908 of *LNAI*, pages 91–109. Springer Berlin / Heidelberg, 2007.
- [20] A. Ricci, M. Viroli, and G. Piancastelli. simpA: A simple agent-oriented Java extension for developing concurrent applications. In M. Dastani, A. E. F. Seghrouchni, J. Leite, and P. Torroni, editors, *Languages, Methodologies and Development Tools for Multi-Agent Systems*, volume 5118 of *LNAI*, pages 176–191, Durham, UK, 2007. Springer-Verlag.
- [21] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

Using HDS for Realizing Multi-Agent Applications

Federico Bergenti, Enrico Franchi, Agostino Poggi

Abstract— Although some of the most important works on multi-agent systems focused on interoperating multi-agent systems with legacy applications, the main results consisted in the definition of two agent communication languages, i.e., KQML and FIPA ACL, and a set of specifications, i.e., the FIPA specifications, for the realization of interoperable multi-agent systems. Nowadays, web services are the primary mean to provide interoperability with legacy applications and the large part of multi-agent applications have been realized without any strong requirement for the interoperability with other multi-agent applications. This paper presents the HDS software framework, which provides a software infrastructure to realize multi-agent applications that either take advantage of the specifications for agent-to-agent interoperability or are implemented for optimizing their performance, reducing their development cost and/or simplifying their interaction with some specific legacy applications. Typed messages and message filters are the elements that mainly characterize such a software framework. Besides describing the main features of such a software framework, this paper introduces two different application scenarios designed to exploit HDS features: i) a prototype framework for distributed constraint satisfaction algorithms and ii) a distributed social network system.

Index Terms—software framework, multi-agent systems, typed messages, composition filters, Java.

I. INTRODUCTION

SOME of the most important works on multi-agent systems considered them the solution to provide and maintain the interoperability among legacy applications [1][2][3]. This expectation motivated researchers to work on the problem of proving interoperability both between agents and legacy applications and among agents that are realized by different people and with different software tools. The main results of such works were not related to the interoperability between agents and legacy applications, but consisted in the definition of two agent communication languages, i.e., KQML and FIPA ACL, [4][5][6] and a set of specifications, i.e., the FIPA

specifications [7], for the realization of interoperable multi-agent systems. Nowadays, the solution for providing the interoperability among legacy applications has been identified in the Web services technologies and the large part of multi-agent applications have been realized without any strong requirement for the interoperability with other multi-agent applications.

In this paper, we present a software framework, called HDS whose goal is to simplify the realization of multi-agent system by taking advantage of typed messages and message filters and avoiding to be constrained by the use of a specific ACL and by the rules of any specification for the realization of multi-agent systems.

We are not concerned with non-agent based software frameworks; among the agent based software frameworks (Jade, AgentFactory [8]) the dominant approach is using FIPA ACL, whose focus is on interoperability. However, in FIPA based frameworks the communication time is almost always dominated by the parsing and construction times of FIPA messages [9][10]. On the other hand, HDS approach permits to shift the focus on performance, without hindering interoperability by design.

Section II gives a short introduction to HDS framework architecture; Section III presents the three models that concur to the definition of the architecture of a HDS application, while Section IV presents some details on the implementation of HDS. Section V and VI discuss two experimentations where HDS has been used to study distributed constraint solving algorithms and to design distributed social network systems. Eventually, section VII concludes the paper sketching some future research directions.

II. SOFTWARE FRAMEWORK OVERVIEW

HDS (Heterogeneous Distributed System) is a software framework that has the goal of simplifying the realization of distributed applications by merging the client-server and the peer-to-peer paradigms and by implementing all the interactions among all the processes of a system through the exchange of messages.

This software framework allows the realization of systems based on two types of processes: actors and servers. Actors have their own thread of execution and perform tasks by interacting, if necessary, with other processes through synchronous and asynchronous messages. Servers perform tasks on request of other processes by composing, if necessary, the services offered by other processes through

Manuscript received June 7, 2010

Federico Bergenti is with the Dipartimento di Matematica, Università degli Studi di Parma, Viale G.P. Usberti, 53/A, 43124 Parma, Italy (e-mail: federico.bergenti@unipr.it).

Enrico Franchi is with the Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy (e-mail: efranchi@ce.unipr.it).

Agostino Poggi is with the Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy (e-mail: poggi@ce.unipr.it).

synchronous messages. Moreover, while both servers and actors may directly take advantage of the services provided by other kinds of application, only the servers can provide services to external applications by simply providing one or more public interfaces.

Actors and servers can be distributed on a (heterogeneous) network of computational nodes (thereafter called runtime nodes) for the realization of different kinds of application. In particular, actors and servers are grouped into some runtime nodes that realize a platform. An application can be obtained by combining some preexistent applications by realizing a federation.

III. APPLICATION ARCHITECTURE MODEL

The software architecture of a HDS application can be described through the three different models:

- the concurrency model, which describes how the processes of a runtime node can interact and share resources.
- the runtime model, which describes the services available for managing the processes of an application.
- the distribution model, which describes how the processes of different runtime nodes can communicate.

A. The concurrency model

The concurrency model is based on seven main elements: process, description, description selector, mailer, message, content and message filter.

A process is a computational unit able to perform one or more tasks taking, if necessary, advantage of the tasks provided by other processes. To facilitate the cooperation among processes, a process can advertize itself making available to the other processes its description. The process identifier and the process type represent the default information contained in a description; however, a process may introduce some additional information in its description.

A process can be either an actor or a server. An actor is an active process that can have an active behavior and so can start the execution of some tasks without the request of other processes. A server is a passive process that is only able to perform tasks in response of the request of other processes.

A process can interact with the other processes through the exchange of messages based on one of the following three types of communication:

- synchronous communication, the process sends a message to another process and waits for its answer;
- asynchronous communication, the process sends a message to another process, performs some actions and then waits for its answer;
- one-way communication, the process sends a message to another process, but it does not wait for an answer.

In particular, while an actor can start all the three previous types of communication with all the other processes, a server can only respond to the requests of the other processes it serves them, composing the services provided by other processes through synchronous communications. Moreover, a server can respond to a request through more than one answer

(e.g., when it acts as a broker in a publisher subscriber system) and can forward a request to another server for its execution.

A process has also the ability of discovering the other processes of the application. In fact, it can both get the identifiers of the other mailers of the systems and check if an identifier is bound to another mailer of the system taking advantage of the registry service provided by HDS middleware. Moreover, a process can take advantage of some special objects, called description selectors, for requiring the listing of specific subsets of mailer identifiers. In fact, a description selector allows the definition of some constraints on the information maintained by the process descriptions (e.g., the process must be of a specific type, the process identifier must have a specific prefix and the process must be located in a specific runtime node) and the registry service is able to apply their constraints on the information of the registered descriptions for building the required subsets of identifiers.

A process does not exchange directly messages with the other processes, but delegates this duty to a mailer. In fact, a mailer provides a complete management of the messages of a process: it receives messages from the mailers of the other processes, maintains them up to the process requests their processing and, finally, sends messages to the mailers of the other processes.

In a way similar to a process, a mailer can be either an actor mailer or a server mailer. Of course, it depends on the fact that, as described above, an actor and a server can assume a different set of roles in message exchanging.

A message contains the typical information used for exchanging data on the net, i.e., some fields representing the header information, and a special object, called content, that contains the data to be exchanged. In particular, the content object is used for defining the semantics of messages (e.g., if the content is an instance of the Ping class, then the message represents a ping request and if the content is an instance of the Result class, then the message contains the result of a previous request).

Normally, a mailer can communicate with all the other mailers and the sending of messages does not involve any operation that is not related to deliver messages to the destination; however, the presence of message filters can modify the normal delivery of messages.

A message filter is a composition filter [11] whose primary scope is to define the constraints on the reception/sending of messages; however, it can also be used for manipulating messages (e.g., their encryption and decryption) and for the implementation of replication and logging services.

Each mailer has two lists of message filters: the ones in the first list (input message filters) are applied to the input messages and the others (output message filters) are applied to the output messages (Fig 1 shows the flow of the messages from the input message filters to the output message filters). When a new message arrives or is to be sent, the message filters of the appropriate list are applied in sequence until a message filter fails; therefore, such a message is stored in the input queue or is sent only if all the message filters have success.

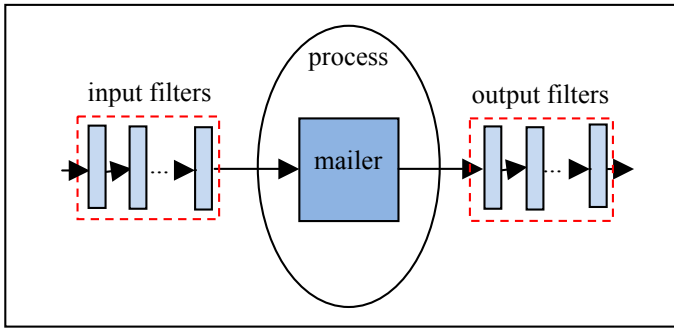


Figure 1. Flow of the messages from the input to the output message filters.

Message filters are not only used for customizing the reception and sending of messages, but are also used by the processes for asking their mailer for the input messages they need for completing their current task. In fact, as described above, a message filter allows to define the constraints that are necessary to identify a specific message and a mailer is able to use it for selecting the first message in the input queue that satisfies its constraints (e.g., the reply to a message sent by the process, a message sent by a specific process and a message with a specific kind of content).

B. The runtime model

The runtime model defines the basic services provided by the middleware to the processes of an application. This model is based on four main elements: registry, processor, filterer and porter.

A registry is a runtime service that allows the discovery of the processes of the application. In fact, a registry provides the binding and unbinding of the processes with their identifiers, the listing of the identifiers of the processes and the retrieval of a special object, called reference, on the basis of the process identifier.

A reference is a proxy of the process that makes transparent the communication respect to the location of the process. Therefore, when a process wants to send a message to another process, it must obtain the reference to the other process and then use it for sending the message.

A processor is a runtime service that has the duty of creating new processes in the local runtime node. Of course, an important side effect of the creation of a process is the creation of the related mailer. The creation is performed on the basis of the qualified name of the class implementing the process, a list initialization parameters.

The processes cannot directly modify the lists of message filters, but they can take advantage of a filterer to do it. A filterer is a runtime service that allows the creation and modification of the lists of message filters associated with the processes of the local runtime node. Therefore, a process can use such a service for managing the lists of its message filters, but also for modifying the lists of message filters associated with the other processes of the local runtime node.

Finally, a porter is a runtime service that has the duty of creating some special objects, called ports, that allows an external application to use the services implemented by a

server of the local runtime node. In particular, a port is a wrapper that encapsulates a server for limiting the access to the functionalities of the process by masquerading the use of some its services and by adding some constraints on the use of some other its services.

C. The distribution model

The distribution model has the goal of defining the software infrastructure that allows the communication of a runtime node with the other nodes of an application possibly through different types of communication supports, guaranteeing a transparent communication among their processes. This model is based on three kinds of element: distributor, connector and connection.

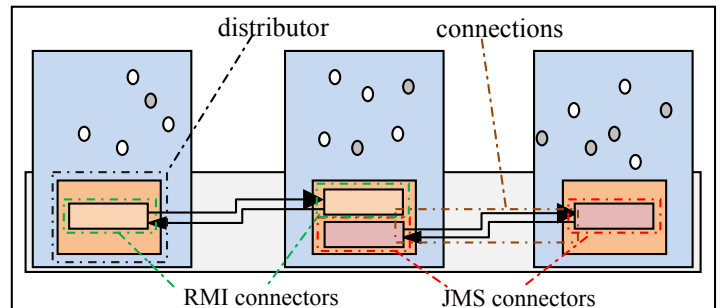


Figure 2. An HDS application based on three runtime nodes connected through RMI and JMS technologies.

A distributor has the duty of managing the connections with the other runtime nodes of the application. This distributor manages connections that can be realized with different kinds of communication technology through the use of different connectors (see Figure 2). Moreover, a pair of runtime nodes can be connected through different connections.

A connector is a connections handler that manages the connections of a runtime node with a specific communication technology allowing the exchange of messages between the processes of the accessible runtime nodes that support such a communication technology.

A connection is a mono-directional communication channel that provides the communication between the processes of two runtime nodes through the use of remote references. In particular, a connection provides a remote lookup service offering the listing of the remote processes and the access to their remote references.

IV. SOFTWARE FRAMEWORK IMPLEMENTATION

The HDS software framework has been realized taking advantage of the Java programming language. The application architecture model has been defined through the use of Java interfaces and its implementation has been divided in two modules.

The first module contains the software components that define the software infrastructure and that are not directly used by the developer, that is, all the software components necessary for managing the lifecycle of processes, the local and remote delivery of messages and their filtering. In particular, the remote delivery of messages has been provided

through both Java RMI [12] and JMS [13] communication technologies.

The second module contains both the software components that application developers extend, implement or, at least, use in their code, and the software components that help them in the deployment and execution of the realized applications. The identification of such software components can be easily done by analyzing what application developers need to realize: i) the actor and server classes used for the implementation of the processes involved in the application, ii) the description of selector classes used for the discovery of the processes involved in common tasks, iii) the message filter classes used for customizing the communication among the processes, iv) the typed messages used in the interaction among the processes, and v) the artifacts (i.e., Java classes and/or configuration files) for the deployment of the runtime nodes and of the communication channels among runtime nodes, and for the startup of the initial sets of processes and message filters.

The above items imply that such a module needs to contain; i) some software components for simplifying the realization of actors, servers, description selectors and message filters (realized through four abstract classes called *AbstractActor*, *AbstractServer*, *AbstractSelector* and *AbstractFilter*), ii) a set of abstract and concrete typed messages useful for realizing the typical communication protocols used in distributed applications, and iii) a software tool that allows the deployment of a HDS software application through the use of a set of configuration files (realized through a concrete class called *Launcher*).

In regard to type messages and the related communication protocols, the software framework provides the basis interfaces and classes for realizing application dependent client-server protocols and the basic interfaces and classes for supporting the interaction among processes through the use of communication language derived by the agent communication language (ACL) defined in the FIPA specifications [7].

In particular, besides realizing an implementation of the FIPA ACL, that completely satisfies the FIPA specifications and uses the SL language for the content, we used some of the FIPA ACL performatives as top layer interfaces for the definition of typed message classes that combines the semantics of the performative with the semantics of the ACL content in a particular ontology (e.g., the typed message, *Sell*, is sent to another process for requiring it to sell something to the requester; of course, typed messages are usually specialized for an application domain and it can be easily done grouping the typed messages related to a domain ontology in a Java package. In a similar way, we provided an abstract implementation of some interaction protocols (i.e. the English and Dutch auction protocols, the Contracted Net and the iterated Contract Net protocols and the brokering and the recruiting protocols) that derive from the interaction protocols defined in the FIPA specifications [7]. This implementation replaces the ACL messages with typed messages and delegates to the application developer only the duty of writing the code for processing the content of the messages, selecting

the messages to be sent and building their content.

For example, the abstract implementation of the iterated Contract Net protocol is based on two abstract classes, that describe the two roles involved in the protocol, i.e., the initiator and the participant, and an interface, called *Contract*, used in the content of the exchanged messages for maintaining the information about both the task to be executed and the bids of the participants.

The abstract class that represents the initiator role defines three main methods; the first method sends an “offer” message to the list of processes acting as participants. The second method is an abstract method whose implementation must select the participant to which send either an “accept” or another “offer” message. Finally, the third method is an abstract method whose implementation must process the message containing the results of the execution of the required task.

The abstract class, that represents the participant role, defines two main methods. The first method is an abstract method whose implementation must decide to propose a bid for the task described by the “offer” message or to refuse it. The second message must decide to execute the task and then must send the information about the results of its execution.

Therefore, using the iterated Contract Net protocol inside an application requires: i) the definition of concrete class implementing the *Contract* interface, ii) the definition of a concrete class that extend the initiator abstract class implementing the methods for accepting, refusing or sending an updated contract and for processing the result received by the participant(s) to which the contract(s) have been assigned, and iii) the definition of at least a concrete class that extend the participant abstract class implementing the methods for accepting or refusing an offer and for performing the task associated with the contract.

V. HDS FOR DISTRIBUTED CONSTRAINT SATISFACTION

Recently we used HDS to develop a Java framework for prototyping and evaluating distributed constraint satisfaction algorithms. A brief introduction to distributed constraint satisfaction is needed to better explain the role of HDS; see [14] for an in-depth introduction to the subject and for a discussion of possible application scenarios.

Distributed Constraint Satisfaction Problems (DCSPs) are a very general class of problems that extend *Constraint Satisfaction Problems (CSPs)* to the realm of distributed computing; the literature defines DCSPs as a distributed and decentralized generalization of CSPs.

A CSP consists of (i) n variables $\langle x_1, x_2, \dots, x_n \rangle$, whose values are taken from finite, discrete domains $\langle D_1, D_2, \dots, D_n \rangle$, respectively, and (ii) a set of constraints on such variables. In very general terms, a constraint is defined by a relation on a subset of the Cartesian product $D_1 \times D_2 \times \dots \times D_n$ that holds for certain assignments of values to variables. Solving a CSP is equivalent to finding an assignment of values to all variables such that all constraints are satisfied. Since constraint satisfaction is NP-complete in general, a trial-and-error exploration of alternatives is inevitable.

A DCSP is a CSP in which variables and constraints are distributed among agents; each agent has some owned variables and it tries to determine their values. Agents independently try to find assignments to their variables and the problem is solved when all variables are assigned consistent values.

More precisely, when dealing with DCSP, we take the following assumptions:

1. No central orchestration is allowed and the problem is solved by peer agents in cooperative/competitive ways.
2. Agents communicate by means of directed messages.
3. Each agent has a unique identifier and an agent can send messages to other agents if and only if it knows the unique identifiers of the receiving agents.
4. The delay in delivering a message is finite, though unknown and possibly random.
5. For the transmission between any pair of agents, messages are received in the order in which they were sent.
6. Each agent has exactly one variable and it knows all constraint predicates relevant to its variable.
7. All constraints are binary.

It is worth noting that although algorithms for solving DCSPs are similar to parallel/distributed processing methods for solving CSPs (see, e.g., [15] [16]), the applicability of both approaches is fundamentally different. The primary concern in parallel/distributed processing is efficiency, and we can choose any type of parallel/distributed computer architecture for solving a given problem efficiently. In contrast, in a DCSP, there already exists a situation where knowledge about the problem is distributed among agents and no central orchestration is available. This is the case, e.g., of sensor networks where nodes interact independently and strive to coordinate with no central master. If all knowledge about the problem could be gathered into a single master agent, such an agent could solve the problem more effectively alone by using every day, centralized constraint satisfaction algorithms.

The Asynchronous Backtracking Algorithm (ABT) is one of the algorithms, that we developed using HDS. This algorithm is a distributed, asynchronous version of a backtracking algorithm. The main message types communicated among agents are *ok?*, to communicate the current assigned value, and *nogood* to communicate a new constraint.

In the ABT algorithm, the priority order of agents is predetermined, and each agent communicates its tentative value assignment to neighboring agents via *ok?* messages. An agent changes its assignment if its current value assignment is not consistent with the assignments of higher priority agents. If there exists no value that is consistent with the higher priority agents, the agent generates a new constraint, called a *nogood*, and it communicates the *nogood* to a higher priority agent; thus the higher priority agent changes its value.

A *nogood* is a subset of an *agent view*, i.e., the current value assignment of other agents from its viewpoint, where the agent is not able to find any consistent value with the subset. Ideally, generated *nogood* should be minimal, i.e., no subset of them

should be a *nogood*. However, since finding minimal *nogoods* requires certain computation costs, an agent can do with non-minimal *nogoods* and, in the simplest case, it could use its entire agent view as a valid *nogood*.

It must be noted that since each agent acts asynchronously and concurrently and agents communicate by sending messages, the agent view may contain obsolete information. Therefore, if x_i does not have a consistent value with the higher priority agents according to its agent view, we cannot use a simple control method such as x_i orders a higher priority agent to change its value, since the agent view may be obsolete. Each agent needs to generate and communicate a new *nogood*, and the receiver of the new *nogood* must check whether the *nogood* is actually violated based on its own agent view.

The potential growth of the size of *nogoods* is a severe issue that went often unnoticed and that we identified during our initial experiments on solving Sudoku puzzles; this was the main reason why we switched our initial implementation from JADE to HDS. Moreover, we found HDS ideal for the implementation of this kind of algorithms because:

1. Performances are important as all such algorithms are typically demanding in terms of communication throughput;
2. Most of such algorithms are expressed in terms of reactions to typed messages; and
3. Composition filters allow instrumenting code with no modifications to developed algorithms, thus enabling performance measurement, debugging and fine tuning.

Finally, it is worth noting that HDS gave us a new dimension for experimentations, i.e., the impact of the underlying transport mechanism on the performances of algorithms. Actually, distributed constraint satisfaction algorithms use messages with very diverse sizes, ranging from few bytes in initial stages of the process to megabytes when agents send entire agent views across the network. We noted that different transport protocol exhibit different performances with message sizes with strange and unforeseen behaviors.

VI. USING HDS IN SOCIAL NETWORK SYSTEMS

Moreover, we are using HDS for the realization of an agent based support layer for the interaction among users in a social network (SN). In particular, we associate an agent with each user and such an agent can also proactively act on her/his behalf by taking advantage the information contained in the profile of the user.

The agent has two main roles: i) it mediates access to the profile information, allowing or refusing queries from other agents; ii) it uses information in the profile in order to discover new friendships and acquaintances on his owner's behalf. While the first role does not need a full-fledged software agent, since a simple rule-based strategy suffices, the second role exhibits a typical proactive behavior, as agents actively pursue their owner's goal, without direct human intervention.

Currently available SNs are implemented with centralized systems where information is stored on a logical central server and users simply connect to that server. The system as a whole

has proactive behavior proposing the users new acquaintances, but its monolithic structure places the system outside the multi-agent paradigm. Moreover, the system has access to every piece of information users provided: this both raises security and privacy concerns and simplifies the proposal of new friendships.

We have designed a system where independent multiple proactive agents exchange minimal sets of data in order to discover relationship suggested by the user profiles. For example, if two users work in the same company, it is likely they know each other, thus they are to be connected in the SN.

Data are distributed among the agents and are protected by the agents themselves, since every access to a datum is mediated through an agent. Thus, privacy is not an issue.

The system supports “typed” connections, where the parts involved are aware they are connected, for example, because both attended to the same University or worked in the same company. In order to store data in the profile, we use FOAF [17] and DOAC [18] and the “type” of the connections is derived from those RDF descriptions. However, we do not detail the semantics of connections in order to focus the system presentation from a multi-agent modeling point of view.

The HDS framework is used as the foundation of our system because of its high efficiency and built-in support for typed messages, which are of paramount importance in expressing our connection negotiation algorithm.

Since the HDS framework distinguishes between active processes (actors) and servers, and since our agents feature both proactive and passive behavior, we decided to model the abstract agents with more than one concrete HDS process. Essentially, the agent discovery algorithm can be decomposed in three main tasks: i) search new connections and friendships according to the data available; ii) broker connections between possibly mutual friends iii) accept/refuse connections proposed by some other agent performing function i and ii. Tasks i) and ii) are clearly proactive, since the agent has to actively contact other agents, thus both tasks are implemented through HDS actors. Although task iii) is not proactive, and can be modeled with a server process. A passive server process mediates access to the profile and this can be seen as a fourth task.

Since agents are implemented through multiple processes, they are essentially only logical entities in the system, the only indication of their existence being a unique id in the system (such as, e.g., their owner's username) and rules granting full access among processes implementing the same agent.

We use capital letters to refer to the agents ids (e.g., A), and the same capital letter with a subscript (A₁, A₂, A₃, A₄) to refer to the HDS ids of the processes implementing the agent, e.g., A₁ implements the first task and so on.

In the following paragraphs we describe the connection discovery algorithm (Fig. 3), which is the component in our system that more heavily exploits HDS typed messages.

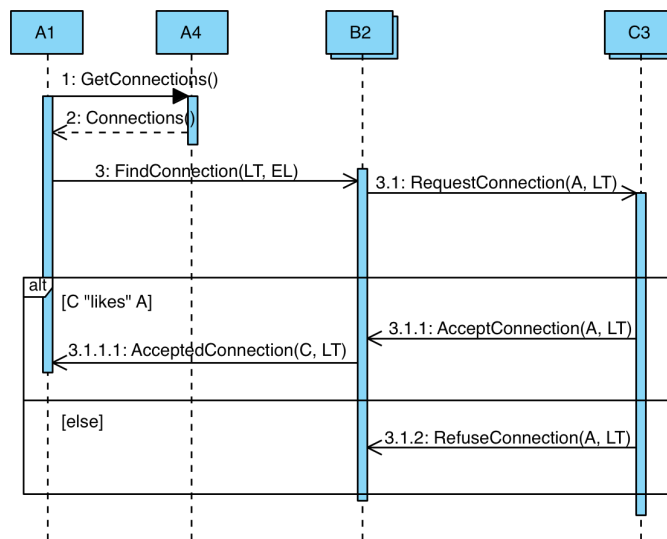


Figure 3. A sequence diagram presenting the connection discovery algorithm.

In order to describe the algorithm, we assume agent A wants to find new friends. As the first step, actor A₁ sends a GetConnections message to A₄ to obtain the list of connections. Each entry in the lists consists of an agent id and an RDF payload specifying the type of connection. Of course, the same pair of agents may be connected through multiple connections.

Let B an identifier in the list: A sends some FindConnection(LT, EL) messages to B₂, where each message has a different link type LT (derived from the types of links connecting A and B). B₂ is then entitled to share pieces of information derived from LT with every agent C that is connected with B through a LT connection and not present in the exclude list EL. EL contains both the ids of agents A is already connected with and the ids of agents A does not want to connect with.

Essentially B₂ acts as a broker between A and C, since mailer filters are configured not to accept connections from processes implementing unknown agents and consequently A and C cannot communicate directly. Notice that A determines the exact amount of information it wants to use in order to find new friends. B is not allowed to use information on A to find new friends for A or for himself until A allows usage of information contained in LT sending the FindConnection(LT, EL) message. For example, the link type is “attended University of Parma”. C already knows that both he and B attended the University of Parma: A allowed B to inform C that A attended that University as well.

The next step consists in B₂ sending C₃ a RequestConnection(A, LT) message. If C₃ answers with a RefuseConnection(A, LT), B will not tell A that he is connected with C (and not even C existence). If C wants to be connected with A, C₃ sends an AcceptConnection(A, LT) to B₂ and consequently B₂ sends an AcceptedConnection(C, LT) message to A₃.

A can confirm the connection to C or refuse it. In the former situation, A₄ will be notified it is allowed to share some information with C and a direct negotiation between A and C

is started in order to establish further connections (or more specific connections, e.g. attended the University of Parma between 2002 and 2005, in place of the simple “attended University of Parma”).

VII. CONCLUSION

This paper presented the HDS software framework, with the goal of simplifying the realization of distributed applications by merging the client-server and the peer-to-peer paradigms and by implementing the interactions among all the processes of a system through the exchange of typed messages.

HDS is implemented by using the Java language and its use simplify the realization of systems in heterogeneous environments where computers, mobile and sensor devices must cooperate for the execution of tasks. Moreover, since different protocols can be used to exchange messages between processes of different computational nodes, it is possible to use multiple implementations of the HDS framework for different languages in the same application as long as there are some shared protocols; this way it is possible to integrate hardware and software platforms without Java support.

HDS can be considered a software framework for the realization of any kind of distributed system. Some of its functionalities derive from the one offered by JADE [19][20], a software framework that can be considered one of the most known and used software framework for the developing of multi-agent systems. This derivation does not depend only on the fact that some of the people involved in the development of the HDS software framework were involved in the development of JADE too, but because HDS proposes a new view of multi-agent systems where the respect of the FIPA specifications are not considered mandatory and ACL messages can be expressed in a way that is more usable by software developers outside the multi-agent system community. This work may be of interest not only for enriching other theories and technologies with some aspects of multi-agent system theories and technologies, but also for providing new opportunities for the diffusion of both the knowledge and use of multi-agent system theory and technologies.

HDS is a suitable software framework for the realization of pervasive applications. Some of its features introduced above (i.e., the java implementation, the possibility of using different communication protocols and the possibility a multi-language implementation) are fit for such kinds of application. However, the combination of multi-agent and aspect-oriented techniques [21] might be one of the best solutions for providing an appropriate adaptation level in a pervasive application. In fact, this solution allows to couple the power of multi-agent based solutions with the simplicity of compositional filters solutions guaranteeing both a good adaptation to the evolution of the environment and a limited overhead to the performances of the applications.

Current and future research activities are dedicated, besides to continue the experimentation and validation of the HDS software framework in the realization of collaborative services for social network, to the improvement of the HDS software

framework. In particular, current activities are dedicated to: i) the automatic creation of the Java classes representing the typed messages from OWL ontologies taking advantage of the O3L software library [22], and iii) the extension of the software framework with a high-performance software library to support the communication between remote processes, i.e., MINA [23].

REFERENCES

- [1] Genesereth, M.R., Ketchpel, S.P. Software agents. *Communications of ACM*, 37(7): 48-53, 1994.
- [2] Genesereth, M.R. An agent-based framework for interoperability. In J. M. Bradshaw (Ed.). *Software Agents*, pp. 317-345. MIT Press, Cambridge, MA, 1997.
- [3] O'Brien, P.D., Nicol, R.C. FIPA - Towards a Standard for Software Agents. *BT Technology Journal*, 16(3):51-59. 1998.
- [4] Finin, T., Fritzson, R., McKay, D. McEntire, R. KQML as an agent communication language. In *Proc. of the 3rd Int. Conf. on information and Knowledge Management*, pp. 456-463, Gaithersburg, MD, 1994.
- [5] Labrou, Y., Finin, T., Peng, Y. Agent Communication Languages: The Current Landscape. *IEEE Intelligent Systems*, 14(2):45-52, 1999.
- [6] Singh, M.P. Agent Communication Languages: Rethinking the Principles. *IEEE Computer*, 31(12):40-47, 1998. G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
- [7] FIPA Consortium. FIPA Specifications. Available from <http://www.fipa.org>.
- [8] Collier, R.: Agent Factory: An Environment for the Engineering of Agent-Oriented Applications. Ph.D. Thesis, University College Dublin, Ireland, 2001
- [9] Mulet, L., Such, J. M., and Alberola, J. M. 2006. Performance evaluation of open-source multiagent platforms. *Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multiagent Systems*, Hakodate, Japan, , 2006.
- [10] G. Vitaglione, F. Quarta, and E. Cortese, "Scalability and performance of jade message transport system," 2002.
- [11] Bergmans, L., Aksit, M. Composing crosscutting concerns using composition filters. *Communications of ACM*, 44(10):51-57, 2001.
- [12] Pitt, E. McNiff, K. *Java.rmi: the Remote Method Invocation Guide*. Addison-Wesley, 2001.
- [13] Monson-Haefel, R. Chappell, D. *Java Message Service*. O'Reilly & Associates, 2000.
- [14] Yokoo, M., Katsutoshi, H. Algorithms for Distributed Constraint Satisfaction: A Review, *Procs. Int'l Conf. Autonomous Agents and Multiagent Systems*, Vol. 3, pp. 185-207, 2000.
- [15] Zhang, Y., Mackworth, A. Parallel and distributed algorithms for finite constraint satisfaction problems. *Procs. 3rd IEEE Symposium on Parallel and Distributed Processing*. 394-397, 1991.\
- [16] Collin, Z., Dechter, R. , Katz S. On the Feasibility of Distributed Constraint Satisfaction. *Procs. 12th Int'l Joint Conference on Artificial Intelligence*. 318-324, 1991.
- [17] D. Brickley and L. Miller, <http://www.foaf-project.org>
- [18] R. Antonio, <http://ramonantonio.net/doac>
- [19] Bellifemine, F., Poggi, A., Rimassa, G. Developing multi agent systems with a FIPA-compliant agent framework. *Software Practice & Experience*, 31:103-128, 2001.
- [20] Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.. JADE: a Software Framework for Developing Multi-Agent Applications. *Lessons Learned. Information and Software Technology Journal*, 50:10-21, 2008.
- [21] Kiczales, Gregor; John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin (1997). "Aspect-Oriented Programming". *Proceedings of the European Conference on Object-Oriented Programming*, vol.1241. pp. 220.242. The paper generally considered to be the authoritative reference for AOP.
- [22] Poggi, A. Developing Ontology Based Applications with O3L. *WSEAS Trans. on Computers*, 8(8):1286-1295, 2009
- [23] Apache Foundation. MINA software. Web site. Available from: <http://mina.apache.org>.

Author Index

Aldewereld, H., 14

Bergenti, F., 62

Bourgne, G., 34

Braubach, L., 41

Collier, R., 7

Costa, A.C.R., 21

Dignum, V., 14

Dimuro, G., 21

Franchi, E., 62

Guidi, M., 48

Hübner, A., 21

Inoue, K., 34

Jander, K., 41

Lamersdorf, W., 41

Lillis, D., 7

Lorini, E., 27

Mattos, V., 21

Maudet, N., 34

Minotti, M., 55

Perrussel, L., 27

Poggi, A., 62

Pokahr, A., 41

Ricci, A., 48, 55

Santi, A., 48, 55

Thévenin, J., 27

Third International Workshop on Logics for Resource Bounded Agents (LRBA 2010)

held as part of

**Multi-Agent Logics, Languages and Organisations – Federated Workshops
(MALLOW 2010)**

Lyon, September 1-2 2010

Workshop Notes

Preface

Formal models of knowledge and belief, as well as other attitudes such as desire or intention, have been extensively studied. However, most of the treatments of knowledge and belief make strong assumptions about reasoners. For example, traditional epistemic logic says that agents know all logical consequences of their knowledge. Similarly, logics of action and strategic interaction are usually based on game theoretic models which assume perfect rationality. Models based on such assumptions can be used to describe ideal agents without bounds on resources such as time, memory, etc, but they fail to accurately describe non-ideal agents which are computationally bounded. The *Logics for Resource Bounded Agents Workshop* (LRBA) aims to provide a forum for discussing possible solutions to the problem of formally capturing the properties of knowledge, belief, action, etc. of non-idealised resource-bounded agents.

The 2010 workshop is the third *LRBA* workshop, following previous events at *ESSLLI* 2006 in Malaga and *MALLOW* 2007 in Durham. We would like to thank the authors and the reviewers for helping to make *LRBA* 2010 a success.

Thomas Ågotnes
Natasha Alechina
Brian Logan

Contents

Computationally grounded account of belief and awareness for AI agents <i>Natasha Alechina and Brian Logan</i>	1
Logic meets cognition: empirical reasoning in games <i>Sujata Ghosh, Ben Meijering and Rineke Verbrugge</i>	15
An extension of RB-ATL <i>Nguyen Hoang Nga</i>	35
Abduction for (non-omniscient) agents <i>Fernando Soler-Toscano and Fernando R. Velázquez-Quesada</i>	51
Dynamic Epistemic Logic for Implicit and Explicit Beliefs <i>Fernando R. Velázquez-Quesada</i>	65
A Construction of Logic-Constrained Functions with Respect to Awareness <i>Susumu Yamasaki</i>	84

Computationally grounded account of belief and awareness for AI agents

Natasha Alechina and Brian Logan

Abstract

We discuss the problem of designing a computationally grounded logic for reasoning about epistemic attitudes of AI agents, mainly concentrating on beliefs. We briefly review existing work and analyse problems with semantics for epistemic logic based on accessibility relations, including interpreted systems. We then make a case for syntactic epistemic logics and describe some applications of those logics in verifying AI agents.

1 Introduction

The *Belief-Desire-Intention* (BDI) model of agency is arguably the most widely adopted approach to modelling artificial intelligence agents [18]. In the BDI approach, agents are both characterised and programmed in terms of propositional attitudes such as beliefs and goals and the relationships between them. For the BDI model to be useful in developing AI agents, we must be able to correctly ascribe beliefs and other propositional attitudes to an agent. However standard epistemic logics suffer from several problems in ascribing beliefs to computational agents. Critically, it is not clear how to connect the computational implementation of an agent to the beliefs we ascribe to it. As a result, standard epistemic logics model agents as logically omniscient. The concept of logical omniscience was introduced by Hintikka in [19], and is usually defined as the agent knowing all logical tautologies and all the consequences of its knowledge. However, logical omniscience is problematic when attempting to build realistic models of agent behaviour, as closure under logical consequence implies that deliberation takes no time. For example, if processes within the agent such as belief revision, planning and problem solving are modelled as derivations in a logical language, such derivations require no investment of computational resources by the agent.

In this paper we present an alternative approach to modelling agents which addresses these problems. We distinguish between beliefs and reasoning abilities which we ascribe to the agent ('the agent's logic') and the logic we use to reason *about* the agent. In this we follow, e.g., [21, 20, 17]. In the spirit of [33], our logic to reason about the agent's beliefs is grounded in a concrete computational model. However, unlike [33, 29] we choose not to interpret the agent's beliefs as propositions corresponding to sets of possible states or runs of the agent's program, but syntactically, as formulas 'translating' some particular configuration of variables in the agent's internal

state. One of the consequences of this choice is that we avoid modelling the agent as logically omniscient. This has some similarities with the bounded-resources approach of [15] and more recent work such as [1, 4].

This paper is essentially a high-level summary of the course on logics and agent programming languages the authors gave at the 21st European Summer School in Logic, Language and Information held in Bordeaux in 2009. Some of the ideas have appeared in our previous work, for example [5, 6], but have never been summarised in a single article.

The rest of the paper is organised as follows. In section 2 we discuss motivations for modelling intentional attitudes of AI agents in logic. In section 3 we analyse problems with the standard semantics for epistemic logic, including interpreted systems. In section 4 we discuss other approaches to modelling knowledge and belief, namely the syntactic approach, logic of awareness, and algorithmic knowledge. Then we introduce our proposal based on the syntactic approach in section 5 and briefly survey some of the applications of the syntactic approach in verification of agent programs in section 6.

2 Logic for verification

There are many reasons for modelling agents in logic. The focus of our work is on specifying and verifying AI agents using logic. The specification and verification of agent architectures and programs is a key problem in agent research and development. Formal verification provides a degree of certainty regarding system behaviour which is difficult or impossible to obtain using conventional testing methodologies, particularly when applied to autonomous systems operating in open environments. For example, the use of appropriate specification and verification techniques can allow agent researchers to check that agent architectures and programming languages conform to general principles of rational agency, or agent developers to check that a particular agent program will achieve the agent's goals in a given range of environments.

Ideally, such techniques should allow specification of key aspects of the agent's architecture and program, and should admit a fully automated verification procedure. One such procedure is model-checking [12]. Model-checking involves representing the system to be verified as a transition system M which can serve as a model of some (usually temporal) logic, specifying a property of the system as a formula ϕ in that logic, and using an automated procedure to check whether ϕ is true in M . However, while there has been considerable work on the formal verification of software systems and on logics of agency, it has proved difficult to bring this work to bear on verification of agent architectures and programs. On the one hand, it can be difficult to specify and verify relevant properties of agent programs using conventional formal verification techniques, and on the other, standard epistemic logics of agency (e.g., [16]) fail to take into account the computational limitations of agent implementations.

Since an agent program is a special kind of program, logics intended for the specification of conventional programs can be used for specifying agent programming languages. In this approach we have some set of propo-

sitional variables to encode the agent’s state, and, for example, dynamic or temporal operators for describing how the state changes as the computation evolves. However, for agents based on the Belief-Desire-Intention model of agency, such an approach fails to capture important structure in the agent’s state which can be usefully exploited in verification. For example, we could encode the fact that the agent has the belief that p as the proposition u_1 , and the fact that the agent has the goal that p as the proposition u_2 . However such an encoding obscures the key logical relationship between the two facts, making it difficult to express general properties such as ‘an agent cannot have as a goal a proposition which it currently believes’. It therefore seems natural for a logical language intended for reasoning about agent programs to include primitives for the beliefs and goals of the agent, e.g., where Bp means that the agent believes that p , and Gp means that the agent has a goal that p .

Given that a logical language intended for reasoning about agent programs should include primitives for the beliefs and goals of an agent, what should the semantics of these operators be? For example, should the belief operator satisfy the KD45 properties? In our view, it is critical that the properties of the agent’s beliefs and goals should be grounded in the computation of the agent (in the sense of [31], that is, there should be a clear relationship between the semantics of beliefs and goals and the concrete computational model of the agent). If the agent implements a full classical reasoner (perhaps in a restricted logic), then we can formalise its beliefs as closed under classical inference. However if the agent’s implementation simply matches belief literals against a database of believed propositions without any additional logical reasoning, we should not model its beliefs as closed under classical consequence. The notion of ‘computationally grounded’ logics is discussed in more detail in the next section.

3 Standard epistemic logic is not computationally grounded

Since the first BDI logics such as [13] and [27], the knowledge and beliefs of AI agents have been modelled using epistemic modal logics with possible worlds semantics. An agent i believes a formula ϕ in a possible world or state s if ϕ is true in all states s' which are belief-accessible from s . For knowledge, the accessibility relation is usually assumed to be an equivalence relation between states, intuitively meaning that the agent cannot tell whether the actual state is s or one of the other knowledge-accessible states. This relation is often referred to as the ‘indistinguishability relation’ of agent i and denoted by \sim_i . A more concrete version of the possible worlds semantics are interpreted systems introduced in [16], where each state is an n -tuple of the agents’ local states and the state of the environment (assuming the system consists of n agents and an environment) and $s \sim_i s'$ holds if the local state of agent i is the same in s and s' . The logic over interpreted systems has temporal operators in addition to the epistemic ones, and the formulas are interpreted over computational runs (sequences of states). The epistemic logics based on this semantics have attractive formal properties. However, they suffer from two main problems: the problem of correctly ascribing beliefs to

an agent, and the problem of logical omniscience.

The problem of belief ascription is concerned with the difficulty of determining what an agent's beliefs are at a given point in its execution. Many agent designs do not make use of an explicit representation of beliefs within the agent. For example, the behaviour of an agent may be controlled by a collection of decision rules or reactive behaviours which simply respond to the agent's current environment. However when modelling the agent, it can still be useful to view the agent as having beliefs. For example, when modelling an agent with a reactive architecture which does not explicitly represent beliefs, we may say that "the agent believes there is an obstacle to the left" and "if the agent believes there is an obstacle to the left, it will turn to the right". However, for this to be possible, we need some principled way of deciding what the agent believes.

However, even when agents do represent beliefs explicitly, the mapping between the agent's belief state and the logical model of the agent is not straightforward. As noted by van der Hoek and Wooldridge [32, p.149] "possible worlds semantics are generally *ungrounded*. That is, there is usually no precise relationship between the abstract accessibility relations that are used to characterise an agent's state, and any concrete computational model." This makes it difficult to use BDI logics for specifying agent systems or to use model-checking tools and algorithms to model-check a particular agent program, since one would need to somehow extract from the program the belief accessibility relations for generating a logical model for use in model-checking. "Because, as we noted earlier, there is no clear relationship between the BDI logic and the concrete computational models used to implement agents, it is not clear how such a model could be derived." [32, p. 153] This problem does surface in model-checking BDI agent programs; see, for example, [8], where the beliefs of an agent which is intended to implement the LORA architecture [34] (which uses standard semantics for beliefs) are modelled syntactically as a finite list of formulas rather than using an accessibility relation. Similar concerns about a gap between BDI logics and concrete agent programs, or the lack of groundedness, were raised by Meyer in [23].

One consequence of this lack of computational grounding is that epistemic logics based on possible worlds semantics model agents as logically omniscient reasoners: they believe/know all tautologies and they believe/know all logical consequences of their beliefs/knowledge ($B_i \top$ and $B_i \phi \wedge B_i (\phi \rightarrow \psi) \rightarrow B_i \psi$ are tautologies of any logic with a modal operator B_i defined as truth in all i -accessible worlds). In effect, agents are modelled as perfect logical reasoners with unlimited computational powers. This is problematic when attempting to build realistic models of agent behaviour, where the time required by the agent to solve a problem is often of critical importance.

Some authors (for example, [26], [25]) argue that unlike the possible worlds structures, interpreted systems can be seen as a grounded semantics for intensional logics. The following arguments are paraphrased from [25, p.36]

- since the semantics of interpreted systems refers to computational runs, a system description in terms of runs (using local states, protocols, etc.) immediately provides a logical model to evaluate formulae;

- epistemic properties are based on the equivalence of local states (which is a concrete computational notion); and
- local states could be represented as e.g. arrays of variables, thereby allowing for a ‘fine grained’ description of agents.

The last point and the examples of modelling multi-agent systems such as Dining Cryptographers given in [26, 25] suggest the following way of ascribing knowledge to agents. The local state of the agent (values of variables) determines what the agent ‘knows’. For example, a propositional variable $paid_i$ may mean that a variable v_1 in agent i ’s state has value $Paid$ (see [26]). Clearly, we want that when the agent i is in the same local state s_i^0 where $v_1 = Paid$, $K_i paid_i$ holds. And this works out with the interpreted systems definition of knowledge: since $paid_i$ holds in all global states where agent i ’s state is s_i^0 , in all those global states $K_i paid_i$ holds.

However, while at first sight this semantics may appear to be computationally grounded, we argue that it is a very roundabout way of defining an agent’s knowledge. The indistinguishability relation \sim_i between global states is used for the truth definition of K_i formulas, so to determine the truth of such formulas we need to examine all global states related by \sim_i , which adds significant complexity to, for example, the model-checking problem. This additional (and unnecessary) complexity is purely an artefact of the possible worlds truth definition of K_i . From the start we decided that what the agent actually knows depends on the properties of the agent’s local state and we should really only have to examine the agent’s state.

Another, more serious, artefact of this truth definition, is that $K_i \phi$ holds not only for the formulas ϕ which do correspond to some properties of the agent’s state (the real or explicit knowledge of the agent), but also for a host of other formulas. Among those additional formulas are tautologies and logical consequences of the real knowledge of the agent such as $K_i(\neg(paid_i \wedge \neg paid_i))$, consequences by introspection (such as $K_i K_i K_i K_i paid_i$ and $K_i K_i K_i K_i \neg K_i \neg paid_i$) and, even more paradoxically, formulas talking about the global properties of the system. For example, if there is just one global state s^0 where agent i ’s local state is s_i^0 , and s^0 has a single successor s^1 , then i ‘knows’ precisely what the next global state looks like. For example, suppose some proposition q is true in s^1 . Then $\bigcirc q$ (‘in the next state, q ’) is true in s^0 . Since s^0 is the only state \sim_i -accessible from s^0 , agent i knows that in the next state q : $K_i \bigcirc q$, and similarly for all the other formulas true in s^1 . This is by no means a grounded knowledge ascription. Even if the system is entirely deterministic, agent i does not necessarily have any knowledge of this.

4 Other approaches

Problems such as logical omniscience which arise when interpreted systems are used to model resource-bounded reasoners have been known for a long time, and some proposed solutions are described in [16]. One of the solutions is termed syntactic in [16]: instead of using a possible worlds truth definition for the knowledge modality K_i , in each state we get essentially a syntactic assignment of formulas agent i believes in that state. For consistency with

what follows we will denote this set of formulas as $\mathcal{A}_i(s)$. The truth definition for $K_i\phi$ in state s of a model M will then become $M, s \models K_i\phi$ iff $\phi \in \mathcal{A}_i(s)$. Clearly, this truth definition makes K_i entirely free of the problem of logical omniscience; however in [16] it is argued that its shortcoming is the lack of any interesting properties of K_i . They prefer a related approach, of combining K_i defined using the possible worlds definition with a new syntactic operator A_i standing for ‘awareness’. K_i is the standard (true in all \sim_i) notion of knowledge (called ‘implicit knowledge’ in [16]); $A_i\phi$ is true if $\phi \in \mathcal{A}_i(s)$ and this only means that i is aware of ϕ , not that i necessarily knows that ϕ ; and $X_i\phi =_{df} K_i\phi \wedge A_i\phi$ means that i explicitly knows ϕ . Clearly, explicit knowledge is also not closed under consequence since i may not be aware of some of the consequences.

Another related approach to ‘fixing’ the idealisation inherent in the possible worlds definition of knowledge is the concept of algorithmic knowledge [16]. Instead of some arbitrary syntactic set $\mathcal{A}_i(s)$, we assume that each state s comes equipped with an algorithm $alg_i(s)$ and agent data (used by the algorithm) $data_i(s)$. An agent algorithmically knows a formula ϕ in s if the output of $alg_i(s)$ for ϕ is ‘yes’. In [16], a number of interesting questions are raised, for example how to relate the computation of the knowledge answering algorithm to the rest of the system dynamics, but the authors decided to keep the two issues separate: the algorithm computation is assumed to be instantaneous and not included in the rest of the system transitions. Subsequent work on algorithmic knowledge, see for example [24], continues to adopt this ‘closure under the algorithm’ condition for the algorithmic knowledge, although beliefs are represented as tokens, and the algorithm as a set of rewriting rules.

5 Syntactic belief ascription

In this section we present *syntactic belief ascription* as an alternative approach to computationally grounded belief ascription. We distinguish between beliefs and reasoning abilities which we ascribe to the agent (‘the agent’s logic’) and the logic we use to reason *about* the agent. In this we follow, e.g., [21, 20, 17]. Our approach grounds the ascription of belief in the state of the agent and allows us to explicitly model the computational delay involved in updating the agent’s state. Our logic for reasoning about the agent’s beliefs is grounded in a concrete computational model in the sense of [33]. However, unlike [33, 29] we choose not to interpret the agent’s beliefs as propositions corresponding to sets of possible states or runs of the agent’s program, but syntactically, as formulas ‘translating’ some particular configuration of variables in the agent’s internal state. One of the consequences of this choice is that we avoid modelling the agent as logically omniscient. This has some similarities with the bounded-resources approach of [15] and more recent work such as [14, 4, 1]. We first consider grounded belief ascription (given an agent state, how to ascribe beliefs to it in a grounded way), and then discuss closure assumptions (what assumptions is it safe to make concerning the closure of the agent’s belief set under the agent’s inferential capabilities).

5.1 Grounded belief ascription

Similarly to interpreted systems, we consider states to be $n + 1$ -tuples of local states of n agents and the state of the environment, in other words, $s = (s_1, \dots, s_n, e)$. We describe the properties of the system in a language built from a set of propositional variables \mathcal{P} . Beliefs ascribable to an agent i are a finite set of literals (variables or their negations) over \mathcal{P} which we will denote L_i . Following, for example, Rosenschein and Kaelbling [28], we assume that each agent's state consists of finitely many 'memory locations' l_1, \dots, l_m , and that each location l_j can contain (exactly) one of finitely many values, v_{j_1}, \dots, v_{j_k} . For example, we could have a location l_t for the output of a temperature sensor which may take an integer value between -50 and 50. Based on those values, we can ascribe beliefs about the external world to the agent: for example, based on $l_t = 20$ we ascribe to the agent a belief that the outside temperature is 20 C. Each literal in L_i corresponds to the fact that a given memory location l_j (or set of memory locations) has a given set of values, but 'translates' this into a statement about the world. We assume a mapping \mathcal{A}_i assigning to each state s a set of propositional variables and their negations which form beliefs of agent i in state s . Note that this 'translation' is fixed and does not depend on the truth or falsity of the formulas in the real world. In general, there is no requirement that \mathcal{A}_i be consistent; if a propositional variable and its negation are associated with two different memory locations (e.g., in an agent which has two temperature sensors) then the agent may simultaneously believe that p and $\neg p$ ¹. \mathcal{A}_i does not have to map a single value to a single belief, for example, all values of $l_t > 20$ could be mapped to a single belief that it's "warm". Conversely, we do not assume that for every propositional variable $p \in \mathcal{P}$, either p or $\neg p$ belong to \mathcal{A}_i ; if a location l_j has no value (e.g., if a sensor fails) or has a value that does not correspond to any proposition, then the agent may have no beliefs about the outside world at all. Other intentional notions such as goals can be modelled analogously to beliefs, i.e., by introducing an explicit translation from the contents of the agent's state into the set of goals. We elaborate belief and goal ascription using the notion of a memory location rather than assuming that agents have an internal representation of beliefs or goals e.g., as a list of literals, for reasons of generality. The ascription mechanism described above is applicable to arbitrary agents, not only those with an explicit internal representation of beliefs and goals. In certain sense, we can say that this ascription $\mathcal{A}_i(s)$ corresponds to the agent's 'awareness' of the facts explicitly represented in agent i 's local state in the global state s .

Our aim is to model the transitions of the agent-environment system as a kind of Kripke structure and express properties of the agents in a modal logic. We consider transition systems similar to the interpreted systems of [16], except that the beliefs of agents are modelled as a local property of each agent's state using syntactic assignment \mathcal{A}_i corresponding to agent i 's beliefs. The state of the environment e corresponds to a classical possible world, or a complete truth assignment to propositional variables in \mathcal{P} . Agent

¹This assumes that the agent's program is using beliefs about the outside world, rather than indirect beliefs about sensor reading; in the latter case of course there would be no contradictory beliefs, since different sensors would have different propositions associated with them.

i believes that p in state s , $M, s \models B_i p$, if $p \in \mathcal{A}_i(s)$. Our proposal is technically equivalent to the syntactic model of belief in [16]. The difference between our approach and [16], is that while a syntactic assignment in [16] is an arbitrary set of formulas, we show how to ground this set in the set of values of variables in the agent’s state. Note that this truth definition for B_i does not give rise to any interesting logical properties of B_i , e.g., to KD45 axioms. This is intentional: we do not want our agents to be logically omniscient and the logical properties of agent’s beliefs should be determined by the agent’s architecture and program. However, the agent’s state changes as the agent executes its program; it could be argued that we may assume that some computation of ‘consequences’ of the agent’s beliefs takes so little time that we can safely assume that the set of beliefs is closed with respect to the agent’s ‘internal logic’ (this argument is made in favour of the closure under algorithmic knowledge in [16]). This is the topic of the next section.

5.2 Deductive closure assumptions

Clearly, any assumptions concerning deductive closure of the agent’s beliefs should be based on the agent’s program and on the requirements of modelling. One could argue that if the agent only ever ‘tests’ its beliefs to check whether it believes p or $\neg p$, and belief ascription with respect to p is correct, it is safe to ascribe to the agent a set of beliefs closed with respect to full classical logic since the agent program does not make any choices depending on the presence of all the extraneous beliefs (logical tautologies and the like). However, we would argue that beyond such relatively trivial agent programs, the deductive closure assumptions should be taken seriously since they may result in incorrect belief ascription.

We argue that for any agent which answers queries or chooses actions depending on its beliefs, the assumption of deductive closure of beliefs is only safe if

1. the closure is with respect to the agent’s real ‘internal logic’ or query answering algorithm implemented by the agent program (so the postulated consequences are actually derivable)
2. the requirements of modelling allow for a reasonably coarse granularity of time, and it is reasonable to assume that the agent’s deductive algorithm completes within a finite and reasonably short period of time.

For example, it may make sense to model beliefs of a forward-chaining rule-based agent as closed under applying the forward chaining procedure to a finite set of ground beliefs, provided that it does not take a long time to terminate and we are not concerned with the precise timing of the agent’s response to a query. Under such conditions, it is reasonable to model the agent’s beliefs using the deductive algorithmic knowledge approach [24]. Note that although the set of beliefs of such an agent is deductively closed, it is deductively closed with respect to a very weak logic (basically, a logic containing universal quantifier elimination and modus ponens, which is much weaker than the full classical logic).

Consider again a forward-chaining rule-based agent but assume that its deductive procedure is not guaranteed to terminate (for example, the agent’s

rules contain arithmetic expressions). In this case, even if modelling allows for quite coarse granularity of time, it would not be safe to model beliefs of the agent as deductively closed (since some of the logical consequences will never be derived in reality but will be ascribed to the agent by the deductive closure assumption). In this case, the deductively closed set of beliefs, even in the agent’s own ‘logic’ (its ‘implicit knowledge’) will be an idealisation, and the algorithmic knowledge approach could be made to work (for a given granularity of modelling) by amending the agent’s forward-chaining algorithm with a ‘timeout’ corresponding to the granularity of modelling: e.g. if the agent’s state is ‘sampled’ at 10 minute intervals then the agent’s beliefs can be modelled as closed with respect to applying the forward-chaining algorithm for 10 minutes.

Finally, arguably the safest way of ascribing beliefs to a resource-bounded agent is not to make any closure assumptions for the set of agent’s beliefs, and to model each inference step in the agent’s internal logic as an explicit transition of the system; this choice gives rise to dynamic syntactic logics such as [14, 1, 4].

6 Verifying agent programs

In this section we briefly outline some applications of syntactic epistemic logics in verifying agent programming languages.

6.1 Theorem proving

SimpleAPL is simplified version of the BDI-based agent programming languages 3APL and 2APL, see, e.g., [9]. SimpleAPL programs have explicit data structures for beliefs and goals, and a program is specified in terms of the agent’s beliefs, goals and planning goal rules which specify which plans the agent should adopt given its goals and beliefs. An approach to verification of SimpleAPL programs based on syntactic epistemic logic was described in [2] and extended to verification of agent programs under different deliberation strategies in [3]. Given the explicit representation of beliefs, belief ascription for SimpleAPL agents is straightforward: Bp (the agent believes that p) is true if p is present in the belief base of the agent. SimpleAPL agents do not do any inference, so the set of beliefs is not closed under any inference rules. Verification in [2] is done by theorem proving; an agent program is axiomatised in Propositional Dynamic Logic (PDL) extended with syntactic belief and goal operators, and a statement such as ‘all executions of this program starting in a state with initial beliefs p_1, \dots, p_n and goals $\kappa_1, \dots, \kappa_m$ will achieve the agent’s goals’ can be verified by checking whether the following formula is derivable from the axiomatisation of the agent program:

$$\bigwedge_{i=1}^n Bp_i \wedge \bigwedge_{j=1}^m G\kappa_j \rightarrow [prog] \bigwedge_{j=1}^m B\kappa_j$$

where $prog$ is a translation of the agent’s program into PDL with syntactic belief and goal operators.

As an example, consider the following example of a vacuum cleaner agent. Actions in SimpleAPL are specified using pre and postcondition pairs (intuitively, what the agent should believe before it can execute an action, and what its beliefs are expected to be after it executes an action). Suppose the agent has the following actions:

```
{room1} moveR {-room1, room2}
{room2} moveL {-room2, room1}
{room1, battery} suck {clean1, -battery}
{room2, battery} suck {clean2, -battery}
{-battery} charge {battery}
```

We will abbreviate goals and beliefs of the agent as: c_i for clean_i , r_i for room_i , b for battery, and actions s for suck, c for charge, r for moveR, l for moveL. Suppose the program of the agent contains the following planning goal rules:

```
c1 <- b | if r1 then s else l; s
c2 <- b | if r2 then s else r; s
      <- -b | if r2 then c else r; c
```

These rules allow the agent to select an appropriate plan to achieve a goal given its current beliefs. For example, the first rule can be read as “if the goal is to clean room 1, and the battery is charged, adopt the following plan: if in room one, suck, else move left and then suck”.

The corresponding PDL program expression *prog* is:

$$\begin{aligned} \text{prog} =_{df} & ((Gc_1 \wedge Bb)?; (Br_1?; s) \cup (\neg Br_1?; l; s)) \cup \\ & ((Gc_2 \wedge Bb)?; (Br_2?; s) \cup (\neg Br_2?; r; s)) \cup \\ & (\neg Bb?; (Br_2?; c) \cup (\neg Br_2?; r; c)) \end{aligned}$$

Some example axioms:

$Bp \rightarrow \neg Gp$ (for every variable p : the agent does not have as a goal something that it believes has been achieved)

$Br_2 \wedge Bb \wedge Gc_2 \rightarrow [s](Bc_2 \wedge \neg Bb \wedge Br_2)$ (corresponds to the pre and postconditions of the `suck` action).

We used MSPASS and `pdl.tableaux` theorem provers to prove the following properties:

- if the agent has goals to clean rooms 1 and 2, and starts in the state where its battery is charged and it is in room 1, it can reach a state where both rooms are clean: $Gc_1 \wedge Gc_2 \wedge Bb \wedge Br_1 \rightarrow \langle \text{prog}^3 \rangle (Bc_1 \wedge Bc_2)$ (where prog^3 stands for *prog* repeated three times)
- the agent is guaranteed to achieve its goal (after 3 iterations of the program) $Gc_1 \wedge Gc_2 \wedge Bb \wedge Br_1 \rightarrow [\text{prog}^3](Bc_1 \wedge Bc_2)$

The logic sketched above is grounded in the agent programming language because its models correspond to the agent’s operational semantics. It can be used to specify and automatically verify properties of SimpleAPL programs.

6.2 Model-checking

In the previous section, we sketched an approach to verifying agent programs using theorem proving. Another approach is to use a model-checker. There are two main strands of work in model-checking multi-agent systems and agent programs which are exemplified by: model-checking based on ‘standard’ BDI logics e.g., [22] and model-checking based on a syntactic interpretation of beliefs e.g., [11, 10, 30].

The only model-checker which ‘understands’ epistemic operators (knowledge) is MCMAS [22]. MCMAS allows checking of properties along both temporal and knowledge accessibility relations. Unfortunately, it is nontrivial to relate ‘possible worlds’ knowledge to the knowledge or beliefs of implemented BDI agents.

An alternative approach is to model-check an agent programming language treating belief as a syntactic modality. This is the approach implicitly taken in [11] for the verification of AgentSpeak(F) programs. Belief, desire and intention are defined in terms of the operational semantics of AgentSpeak(F):

- an agent believes ϕ if ϕ is present in its belief base
- an agent intends ϕ if ϕ is an achievement goal that appears in the agent’s set of intentions – i.e., in the agent’s currently executing or suspended plans.

AgentSpeak(F) programs are translated into the Promela modelling language of the Spin model checker. Properties to be model-checked are expressed in a simplified BDI logic translated into the LTL-based property specification language used by Spin. BDI modalities are mapped onto the AgentSpeak(F) structures implemented as a Promela model. So, even though Bordini et al. [11] do not mention the problems with the standard semantics of belief, or dwell on using the syntactic approach to beliefs rather than the LORA framework based on the standard epistemic semantics which they officially adopt, the fact is that they do use a syntactic approach. We argue that this is inevitable in verification of a real agent programs. A similar approach (using syntactic or ‘shallow’ modalities) is adopted in [10] and [30].

The work on model-checking agent programs using syntactic approaches mentioned in this section does not model agent’s deriving consequences from its beliefs explicitly. However, in other work where the main concern is with the time required for the agents to produce a response to a query, we did use model-checking over systems where transitions correspond to agents applying inference rules (usually, forward-chaining rule firing): see [4, 7] for more details.

7 Conclusion

‘Standard’ BDI logics allow properties of beliefs and other intentional attitudes of AI agents to be formalised. The resulting specifications can be model checked using model checkers such as MCMAS. However it is not clear how to implement agents based on these specifications; in particular, it is not clear what corresponds to belief and goal accessibility relations in the

agent programming language or the implemented agent. On the other hand, ‘syntactic’ BDI logics allow more accurate modelling of AI agents. We can verify properties of real agent programs at the belief and goal level (as opposed to simply verifying the agent program as just a computer program).

Acknowledgements Natasha Alechina and Brian Logan were supported by the Engineering and Physical Sciences Research Council [grant number EP/E031226].

References

- [1] Thomas Ågotnes and Natasha Alechina. The dynamics of syntactic knowledge. *Journal of Logic and Computation*, 17(1):83–116, 2007.
- [2] Natasha Alechina, Mehdi Dastani, Brian Logan, and John-Jules Ch. Meyer. A logic of agent programs. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI 2007)*, pages 795–800. AAAI Press, 2007.
- [3] Natasha Alechina, Mehdi Dastani, Brian Logan, and John-Jules Ch. Meyer. Reasoning about agent deliberation. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR’08)*, pages 16–26, Sydney, Australia, September 2008. AAAI.
- [4] Natasha Alechina, Mark Jago, and Brian Logan. Modal logics for communicating rule-based agents. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 322–326. IOS Press, 2006.
- [5] Natasha Alechina and Brian Logan. Ascribing beliefs to resource bounded agents. In Cristiano Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, pages 881–888, Bologna, Italy, July 2002. ACM Press.
- [6] Natasha Alechina and Brian Logan. A logic of situated resource-bounded agents. *Journal of Logic, Language and Information*, 18(1):79–95, 2009.
- [7] Natasha Alechina, Brian Logan, Nguyen Hoang Nga, and Abdur Rakib. Verifying time, memory and communication bounds in systems of reasoning agents. *Synthese*, 169(2):385–403, July 2009.
- [8] Rafael Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. State-space reduction techniques in agent verification. In Nicholas R. Jennings, Carles Sierra, Liz Sonenberg, and Milind Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004)*, pages 896–903, New York, NY, 2004. ACM Press.

- [9] Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seqhrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer, 2005.
- [10] Rafael H. Bordini, Louise A. Dennis, Berndt Farwer, and Michael Fisher. Automated verification of multi-agent programs. In *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), 15-19 September 2008, L'Aquila, Italy*, pages 69–78. IEEE, 2008.
- [11] Rafael H. Bordini, Michael Fisher, Carmen Pardavila, and Michael Wooldridge. Model checking AgentSpeak. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 409–416, New York, NY, USA, 2003. ACM.
- [12] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.
- [13] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
- [14] Ho Ngoc Duc. Reasoning about rational, but not logically omniscient, agents. *Journal of Logic and Computation*, 7(5):633–648, 1997.
- [15] Jennifer J. Elgot-Drapkin and Donald Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:75–98, 1990.
- [16] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.
- [17] Ronald Fagin, Joseph Y. Halpern, and Moshe Y. Vardi. A non-standard approach to the logical omniscience problem. *Artificial Intelligence*, 79(2):203–240, 1996.
- [18] Michael P. Georgeff, Barney Pell, Martha E. Pollack, Milind Tambe, and Michael Wooldridge. The belief-desire-intention model of agency. In Jörg P. Müller, Munindar P. Singh, and Anand S. Rao, editors, *Intelligent Agents V, Agent Theories, Architectures, and Languages, 5th International Workshop, (ATAL'98), Paris, France, July 4-7, 1998, Proceedings*, volume 1555 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1999.
- [19] J. Hintikka. *Knowledge and belief*. Cornell University Press, Ithaca, NY, 1962.
- [20] K. Konolige. *A Deduction Model of Belief*. Morgan Kaufmann, San Francisco, Calif., 1986.
- [21] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence, AAAI-84*, pages 198–202. AAAI, 1984.
- [22] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, 21st*

International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer, 2009.

- [23] John-Jules Ch. Meyer. Our quest for the holy grail of agent verification. In Nicola Olivetti, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, 16th International Conference, TABLEAUX 2007, Aix en Provence, France, July 3-6, 2007, Proceedings*, volume 4548 of *Lecture Notes in Computer Science*, pages 2–9. Springer, 2007.
- [24] Riccardo Pucella. Deductive algorithmic knowledge. *Journal of Logic and Computation*, 16(2):287–309, 2006.
- [25] Franco Raimondi. *Model-checking multi-agent systems*. PhD thesis, Department of Computer Science, University College London, University of London, 2006.
- [26] Franco Raimondi and Alessio Lomuscio. A tool for specification and verification of epistemic properties in interpreted systems. *Electronic Notes in Theoretical Computer Science*, 85:176–191, 2004.
- [27] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.
- [28] S. J. Rosenschein and L. P. Kaelbling. A situated view of representation and control. *Artificial Intelligence*, 73:149–173, 1995.
- [29] N. Seel. The ‘logical omniscience’ of reactive systems. In *Proceedings of the Eighth Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB'91)*, pages 62–71, Leeds, England, 1991.
- [30] Doan Thu Trang, Brian Logan, and Natasha Alechina. Verifying Dribble agents. In Matteo Baldoni, Jamal Bentahar, M. Birna van Riemsdijk, and John Lloyd, editors, *Declarative Agent Languages and Technologies VII, 7th International Workshop, DALT 2009, Budapest, Hungary, May 11, 2009. Revised Selected and Invited Papers*, volume 5948 of *Lecture Notes in Computer Science*, pages 244–261, 2010.
- [31] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):133–157, 2003.
- [32] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):135–159, 2003.
- [33] Michael Wooldridge. Computationally grounded theories of agency. In E. Durfee, editor, *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, pages 13–20. IEEE Press, 2000.
- [34] Michael Wooldridge. *Reasoning About Rational Agents*. MIT Press, 2000.

Logic meets cognition: empirical reasoning in games

Sujata Ghosh, Ben Meijering and Rineke Verbrugge

Department of Artificial Intelligence

University of Groningen

e-mail: {sujata,b.meijering,rineke}@ai.rug.nl

Abstract

This paper presents a first attempt to bridge the gap between logical and cognitive treatments of strategic reasoning in games. The focus of the paper is backward induction, a principle which is purported to follow from common knowledge of rationality by Zermelo's theorem. There have been extensive formal debates about the merits of principle of backward induction among game theorists and logicians. Experimental economists and psychologists have shown that human subjects, perhaps due to their bounded resources, do not always follow the backward induction strategy, leading to unexpected outcomes. Recently, based on an eye-tracker study, it has turned out that even human subjects who produce the outwardly correct 'backward induction answer' use a different internal reasoning strategy to achieve it. This paper presents a formal language to represent different strategies on a finer-grained level than was possible before. The language and its semantics may lead to precisely distinguishing different cognitive reasoning strategies, that can then be tested on the basis of computational cognitive models and experiments with human subjects.

1 Introduction

Strategic reasoning in games concerns the plans or strategies that information-processing agents have for achieving certain goals. *Strategy* is one of the basic ingredients of multi-agent interaction. It is basically the plan of action an agent (or a group of agents) considers for its interactions, that can be modelled as games. From the game-theoretic viewpoint, a strategy of a player can be defined as a partial function from the set of histories (sequences of events) at each stage of the game to the set of actions of the player when it is supposed to make a move [OR94]. Agents devise their strategies so as to force maximal gain in the game.

In cognitive science, the term 'strategy' is used much more broadly than in game theory. A well-known example is formed by George Polya's problem solving strategies (understanding the problem, developing a plan for a solution, carrying out the plan, and looking back to see what can be learned) [Pol45]. Nowadays, cognitive scientists construct fine-grained theories about human reasoning strategies [Lov05, JT07], based on which they construct computational cognitive models. These models can be validated by comparing the model's predicted outcomes to results from experiments with human subjects [And07].

Every finite extensive form game with perfect information [OR94] played by rational players has a sub-game perfect equilibrium and *backward induction* is a popular technique to compute such equilibria. The backward induction strategy, which employs iterated elimination of weakly dominated strategies to obtain sub-game perfect equilibria, is a common strategy followed by rational players with common knowledge (belief) of rationality. We provide below an explicit description of the backward induction algorithm in extensive form game trees [Jon80]. We are only considering strictly competitive games played between two players.

Consider a finite extensive form game with perfect information G played between two players E and A , say. In game G , each player i is associated with a utility function u_i which maps each leaf node of the tree to the set $\{0,1\}$. The backward induction procedure $BI(G, i)$ takes as input such a game G and a player i . It decides whether player i has a winning strategy in G and if so, computes the winning strategy. The procedure proceeds as follows. Initially all nodes are unlabelled.

Step 1: All leaf nodes l are labelled with $u_i(l)$.

Step 2: Repeat the following steps till the root node r is labelled: Choose a non-leaf node t which is not labelled, but all of whose successors are labelled.

- a) If it is i 's turn at t and there exists a successor t' of t which is labelled 1, then label t with 1 and mark the edge (t, t') which gives the best response at that stage.
- b) If it is the opponent's turn at t and every successor t' is labelled 1, then label t with 1.

Player i will have a winning strategy in the game G if and only if the root node r is labelled with 1 by the backward induction procedure $BI(G, i)$.

One important critique of this backward induction procedure is that it ignores information, and such ignorance is hardly consistent with a broad definition of rationality. Under backward induction, the fact that a player ends up in one particular subgame rather than another subgame is never considered information for the player. The past moves and reasoning of the players are not taken into consideration. Only what follows is reasoned about. That is, the backward induction solution ignores any forward induction reasoning [Per10].

Before proceeding further we should mention here that there have been numerous debates surrounding the backward induction strategy from various angles. The paradigm discussion concerns the epistemic conditions of backward induction. Here, Aumann [Aum95] and Stalnaker [Sta96] have taken conflicting positions regarding the question whether *common knowledge of rationality* in a game of perfect information entails the backward induction solution. Researchers such as Binmore have argued for the need for richer models of players, incorporating irrational as well as rational behavior [Bin96]. For more details on these issues see [Bic88, ACB07, Bra07, BSZ09, HP09, Art09].

From the logical point of view, various characterisations of backward induction can be found in modal and temporal logic frameworks [Bon02, HvdHMMW03, vW03, JvdH04, BSZ09]. There are also critical voices around backward induction arising from logical investigations of strategies. While discussing large (perfect information) games played by resource-bounded players, Ramanujam and Simon emphasize that strategizing follows the flow of time in a *top-down* manner rather than the *bottom-up* one advocated by the backward induction algorithm [RS08].

Critique of a different flavor stems from experimental economics [Cam03]. As sketched above, the game-theoretic perspective assumes that people are rational agents, optimizing their gain by applying strategic reasoning. However, many experiments have shown that people are not completely rational in this sense. For example, McKelvey and Palfrey [MP92] have shown that in a traditional centipede game participants do not behave according to the Nash equilibrium reached by backward induction. In this version of the game, the payoffs are distributed in such a way that the optimal strategy is to always end the game at the first move. However, in McKelvey and Palfrey's experiment, participants stayed in the game for some rounds before ending the game: in fact,

only 37 out of 662 games ended with the backward induction solution. McKelvey and Palfrey’s explanation of their results is based on reputation and incomplete information. They compare the complete information dynamic centipede game to an incomplete information game, the iterated prisoner’s dilemma as investigated by Kreps et al. [KMRW82]. McKelvey’s and Palfrey’s main idea is that players may believe that there is some possibility that their opponent has payoffs different from the ‘official ones’: for example, they might be altruists, i.e., they give weight to the opponent’s payoff. Another interpretation of this result is that the game-theoretic perspective fails to take into account the reasoning abilities of the participants. That is, perhaps, due to cognitive constraints such as working memory capacity, participants are unable to perform optimal strategic reasoning, even if in principle they are willing to do so.

In conclusion, we find two very different bodies of work on players’ strategies in centipede-like games: on the one hand we find idealized logical studies on games and strategies modelling interactive systems, and on the other there are experimental studies on players’ strategies and cognitive modelling of their reasoning processes. Both streams of research have been rather disconnected so far.

To the best of the knowledge of the authors, this article presents a first attempt to bridge the gap between the experimental studies, cognitive modeling, and logical studies of strategic reasoning. In particular, we investigate the question whether a logical model can be used to construct better computational cognitive models of strategic reasoning in games. In the next section we discuss some experimental studies on second-order reasoning of players and cognitive models of such reasoning. Section 3 builds up a logical framework to describe empirical reasoning of players. Finally, the last section provides a discussion with some pointers towards future work.

2 Marble Drop: experiments and cognitive model

This section presents a brief overview of the experimental work on Marble Drop games described in [MMRV10], and of the computational cognitive model (in the cognitive architecture ACT-R [And07]), which was developed in [MV10] to provide a model for second-order social reasoning in predicting the opponent’s moves in Marble Drop games.

2.1 Higher-order social cognition

One of the pinnacles of intelligent interaction is higher-order theory of mind, an agent’s ability to model recursively mental states of other agents, including the other’s model of the first agent’s mental state, and so forth. More precisely, zero-order theory of mind concerns world facts, whereas $k + 1$ -order reasoning models k -order reasoning of the other agent or oneself. For example, “Bob knows that Alice knows that he wrote a novel under pseudonym” ($K_{Bob}K_{Alice}p$) is a second-order attribution. Orders roughly correspond to the modal depth of a formula ¹

The authors in [MMRV10, MvRV10] have investigated higher-order social cognition in humans by means of two experiments. They conducted a behavioral experiment to investigate how well humans are able to apply first- and second-order reasoning. Even though behavioral measures can shed some light on the usage of strategies (see e.g. [HZ02]), they are too crude to go into the details of the actual reasoning steps. Johnson, Camerer, Sen, and Rymon’s study employed a novel measure to capture those details [JCSR02]. In their sequential bargaining experiment, participants had to bargain with one other player. The amount to bargain and the participant’s role in every

¹We use the term ‘higher-order social cognition’ instead of ‘higher-order theory of mind’. The reason to do this is that in the philosophical controversy between ‘theory-theory’ and ‘simulation-theory’, the term ‘theory of mind’ carries the unwanted connotation that ‘theory-theory’ is preferred.

round was hidden behind boxes. The participants had to click on a box to make elements of this information visible. This allowed Johnson et al. to record the information regarding what the participants select at a particular time within the bargaining (i.e., reasoning) process. A potential problem of this measure is that participants might feel disinclined to repeatedly check sets of information elements and rather develop an artificial strategy that involves fewer mouse clicks but puts a higher strain on working memory.

To avoid this problem, Meijering et al. choose to employ eye-tracking technology to investigate the details of higher-order social reasoning. They are currently conducting an eye-tracking study to investigate the reasoning steps during higher-order social reasoning [MvRV10]. The findings of this experiment, together with the behavioral results, help to determine the cognitive bounds on higher-order social reasoning. We give a short overview below and refer the reader to the full papers for more details.

2.1.1 Marble Drop games

Meijering et al. [MMRV10] presented participants with strategic games to investigate higher-order social reasoning. In these games, the path of a white marble that was about to drop could be manipulated by removing trapdoors (Figure 2). Experience with world-physics allowed players to see easily how the marble would run through a game, and which player could change the path of the white marble at each decision point in the game. In other words, higher-order social reasoning was embedded in a context that provided an insightful overview of the decisions and the consequences of these decisions.

Earlier, Hedden and Zhang [HZ02] and Flobbe et al. [FVHK08] had also presented participants with strategic games to investigate higher-order social reasoning, but the performance in those games was far from optimal with approximately 60% - 70% correct. The participants could either have had difficulties applying higher-order social reasoning or difficulties understanding the games.

The matrix games (Figure 1) presented in [HZ02] were very abstract, which could have made the games difficult to understand. Embedding the games in a context could have alleviated this problem. Some studies have shown that non-social reasoning can be facilitated by embedding it in a context. For example, for Wason's selection task it has been stated that a social rule-breaking context helps [WS71]; but see [MO91, SvL04]. More convincingly, subjects have been shown to win the game of tic-tac-toe more easily than its equivalent, Number Scrabble [Mic67, Sim79, Wei84]). The role of context and ecological validity in decision making also plays an important role in the work on 'simple heuristics that make us smart' [GT99]. Higher-order social reasoning, which seemed to be very demanding in matrix games, might also benefit from a context embedding.

Flobbe et al. [FVHK08] found some indirect evidence of a facilitative effect of embedding higher-order social reasoning in a context. However, the performance in their experiment was only slightly better than that in Hedden and Zhang's experiments. Flobbe et al. embedded Hedden and Zhang's matrix games in the context of a road game. The road games were played on a computer. The participants were presented with roads that had three junctions, which corresponded with the cell-transitions in the matrix games. At each junction either the participant or the computer decided to move ahead (i.e., continue the game) if there was a higher payoff to attain further in the game, or to turn right (i.e., end the game) if there was no higher payoff to attain further in the game. The participant and the computer alternately took seat in the driver's position; the one in the driver's seat made the decision.

The performance in the road games might not have been optimal because of the unnatural characteristic of the context in these games. The participants (and the computer) alternately changed

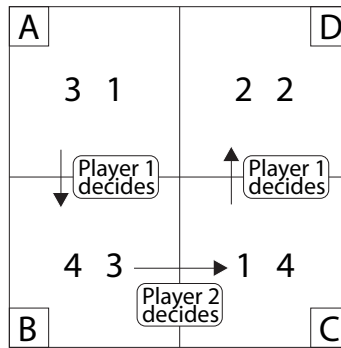


Figure 1: A schematic overview of a matrix game [HZ02]. The first number in each cell is Player 1’s payoff, the second Player 2’s payoff. The goal is to attain the highest possible payoff. Participants first had to predict what the other player would do at cell B before making a decision what to do at cell A: to stay and stop the game, or to move to cell B. In this example, Player 1 would have to predict that Player 2 will stay, because Player 1 will move to cell D if given a choice at cell C, leading to a lower payoff for Player 2, namely 2 instead of 4. Consequently, the rational decision for Player 1 is to move to cell B in the first move.

driver’s seat, which is not common practice in everyday life. Because this unnatural characteristic necessitates some (additional) explanation, the context in the road games was not insightful at first glance.

Meijering et al. [MMRV10] expected that the performance could be improved much further by embedding higher-order social reasoning in their context of a marble game, which was more intuitive and required less explanation. Importantly, these so-called Marble Drop games were game-theoretically equivalent to the matrix games of [HZ02] and the road games of [FVHK08]. All game types have the same extensive form, namely that of the Centipede game [Ros81] (see <http://www.ai.rug.nl/~leendert/Equivalence.pdf>). Consequently, an improvement in performance can be attributed to a context effect.

In Marble Drop games, the payoffs are color-graded marbles, instead of payoff numbers. Meijering et al. presented color-graded marbles instead of numbers to minimize the usage of numeric strategies other than first- and second-order reasoning. The color-graded marbles can easily be ranked according to preference, lighter marbles being less preferred than darker marbles. The ranking makes it possible to have payoff structures similar to those in matrix and road games. The sets of trapdoors in Marble Drop games correspond to the transitions, from one cell to another, in matrix games [HZ02], and to the junctions in road games [FVHK08].

Figure 2 depicts example games of Marble Drop. The goal is to let the white marble end up in the bin with the darkest color-graded marble of one’s own target color (orange or blue). Note that, for illustrative purposes, the color-graded marbles are replaced with codes: a1 - a4 represent the participant’s color-graded marbles and b1 - b4 represent the computer’s color-graded marbles (which are of another color); 1 - 4 being light to dark grades. (See <http://www.ai.rug.nl/~meijering/MarbleDrop.html> for the original Marble Drop games.) The diagonal lines represent the trapdoors.

In the example game in Figure 2a, participants need to remove the right trapdoor to attain the darkest color-graded marble of their color (a2). The game in Figure 2a is a zero-order game, because there is no other player to reason about. In first-order games (Figure 2b) participants need to reason about another player, the computer. The computer is programmed to let the white

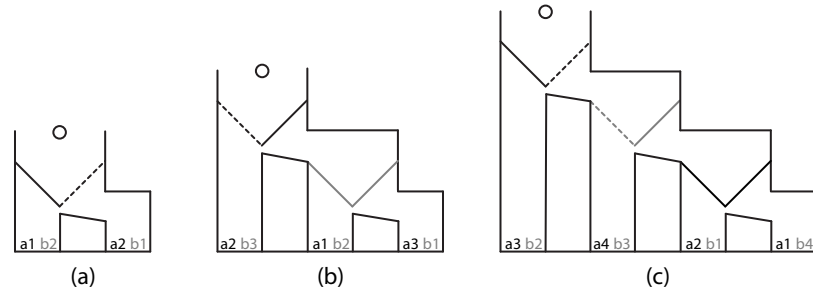


Figure 2: A zero-order (a), first-order (b) and second-order (c) Marble Drop game. The participant’s payoffs are represented by a1 - a4, the computer’s by b1 - b4, both in increasing order of value. The goal is to let the white marble end up in the bin with the highest attainable payoff. The diagonal lines represent trapdoors. At the first set of trapdoors, the participant decides which of both trapdoors to remove, at the second set the computer decides, and at the third set the participant again decides. The dashed lines represent the trapdoors that both players should remove to attain the highest payoff they can get.

marble end up in the bin with the darkest color-graded marble of its target color, which is different from the participant’s target color. Participants need to reason about the computer, because the computer’s decision at the second set of trapdoors affects at what bin a participant can end up.

In the example game in Figure 2b, if given a choice at the second set of trapdoors, the computer will remove the left trapdoor, because its marble in the second bin (b2) is darker than its marble in the third bin (b1). Consequently, the participant’s darkest marble in the third bin (a3) is unattainable. The participant should therefore remove the left trapdoor (of the first set of trapdoors), because the marble of their target color in the first bin (a2) is darker than the marble of their target color in the second bin (a1).

In a second-order game (Figure 2c) there is a third set of trapdoors at which the participants again decide which trapdoor to remove. They need to apply second-order reasoning, that is, reason about what the computer, at the second set of trapdoors, thinks that they, at the third set of trapdoors, think.

In Marble Drop games, half of the participants first had to predict what the opponent would do (at the second set of trapdoors) before deciding what to do at the first set of trapdoors. The participants were instructed that the opponent was rational, as were the participants in Hedden and Zhang’s and Flobbe et al.’s experiments².

Similar to the supporting role of scaffolding in the construction of a building, (instructional) scaffolding provides a supporting structure to learn a new concept or skill that is beyond the independent efforts of a student [WBR76]. Asking participants to make a prediction about the opponent before making a decision was thought to support second-order social reasoning, as decisions should be based on predictions, and making a prediction involves thinking about what the opponent thinks that the participant thinks. In matrix and road games, all the participants first had to give their prediction of what the opponent would do before making their own decision. Meijering et al. [MMRV10] set out to investigate whether embedding higher-order social reasoning in a context that allows for an insightful overview of the decisions and the consequences of these decisions might render such scaffolding obsolete.

²One group of participants in Hedden and Zhang’s experiments [HZ02] knew they were playing against a computer whereas another group did not. Hedden and Zhang found no difference in performance for the two groups.

2.2 Results

Twenty first-year Psychology students participated in exchange for course credit. The mean age of the included participants (12 female) was 21.15 years ($SE = 0.36$).

Whereas the participants in the matrix games [HZ02], and the road games [FVHK08] had difficulties applying second-order reasoning, in Marble Drop games [MMRV10], participants performed almost at ceiling: in 94% of the games, participants correctly applied second-order reasoning. In the former two game types, the performance ranged between 60 - 70%.

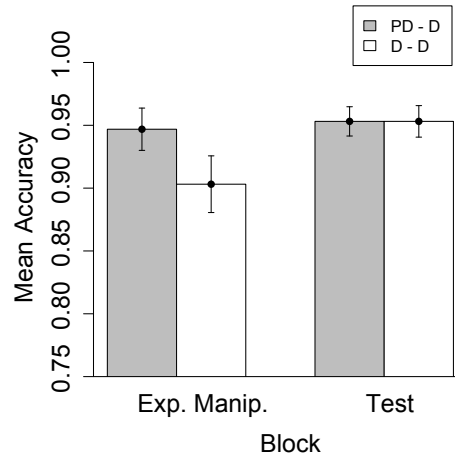


Figure 3: Mean proportion of games in which participants correctly applied second-order reasoning, presented separately for the PD-D and D-D groups in the experimental manipulation block and the test block. The standard errors are depicted above and below the means.

According to the results in [MMRV10], there is no need to scaffold second-order reasoning, given that it is embedded in an ecologically valid context. The performance of the participants that had to make a prediction before making a decision (the PD-D group) was slightly better in the first half (the experimental manipulation block) of the experiment, but in the second half (the test block), the participants that had to make a prediction (the PD-D group) and the participants that did not (the D-D group) performed equally well (Figure 3).

In sum, [MMRV10] demonstrated that adolescents are able to apply second-order reasoning if it is embedded in an ecologically valid context, and that such an embedding renders scaffolding unnecessary. Currently, [MvRV10] are doing an eye-tracking study to elaborate the steps in second-order reasoning during Marble Drop games. We will come back to this discussion in Section 2.5.

2.3 Computational cognitive models of game reasoning

A different perspective, that focuses on cognitive validity in developing formal models, is that of a cognitive architecture [And07]. Cognitive models developed within this framework aim to explain certain aspects of cognition by assuming only general cognitive principles. However, the current cognitive models that describe social interactions do not take higher-order strategic reasoning into account. For example, cognitive models of simple games exist in which it is important to know the opponent's behavior [LWW00, WLB06]. These cognitive models demonstrate that declarative memory is important in playing strategically. In [MV10] however, the authors are less interested

in how people adapt their strategy to an opposing strategy, but rather in studying the cognitive limitations of explicit second-order reasoning.

To provide a full model of second-order social reasoning, [MV10] implemented their model in the cognitive architecture ACT-R [And07]. ACT-R aspires to explain all of cognition using one theoretical framework. To achieve this, the heart of ACT-R consists of a procedural memory system, which contains condition-action pairs known as production rules. Besides the procedural module, ACT-R has designated modules for specific types of information. For example, the visual module processes visual information, whereas the declarative memory module processes declarative or factual information. Each module has a buffer that may contain one unit of information at a time. If the current contents of all buffers in the system match the conditions of a particular production rule, that rule fires and its actions are executed. Each action may refer to an operation in one of the modules.

Two modules of ACT-R deserve extra attention in the light of the model of second-order social reasoning in [MV10]: the *declarative memory module* and the *problem state module*. The *declarative memory module* represents long-term memory and stores information encoded in so-called *chunks* (i.e., knowledge structures). Each chunk in declarative memory has an activation value that determines the speed and success of its retrieval. Whenever a chunk is used, the activation value of that chunk increases. As the activation value increases, the probability of retrieval increases and the latency of retrieval decreases. Anderson [And07] provided a formalization of the mechanism that produces the relationship between the probability and speed of retrieval. If the activation value drops below a certain minimal value (the retrieval threshold), the related information is no longer accessible. In that case, the system will report a retrieval failure after a constant time factor. If the activation value is above the retrieval threshold, the information is accessible. Then, the higher the activation value, the faster the retrieval will be. As soon as a chunk is retrieved, it is put into the retrieval buffer. Each ACT-R module has a buffer that may contain one chunk at a time. On a functional level of description, the chunks that are stored in the various buffers are the knowledge structures the cognitive architecture is aware of.

The *problem state module* (sometimes referred to as the imaginal module) contains a buffer in which information can be temporarily stored. Typically, this information contains a subsolution to the problem at hand. In the case of a social reasoning task, this may be the outcome of a reasoning step that will be relevant in subsequent reasoning. Storing information in the problem state buffer is associated with a time cost (typically 200ms). The model that we present in Section 2.4 relies on the combination of the declarative module and the problem state buffer. That is, the model retrieves relevant information from memory and moves that information to the problem state buffer if new information is retrieved from memory that needs to be stored in the retrieval buffer.

2.4 A computational cognitive model of the Marble drop game

The ACT-R model proposed by [MV10] follows a backward induction strategy to predict the opponent's moves further on in the game. Hedden and Zhang [HZ02] provide a decision tree analysis of this process for their matrix version of the game. The model has knowledge on how to solve Marble Drop games for all possible distributions of payoffs over the bins of the marble run game. That is, the model stores chunks containing information on which payoffs to compare at each step. In addition, chunks representing the magnitudes of the payoff shades are stored in declarative memory, as well as chunks representing the location of the payoffs on the screen. Finally, chunks representing ordinal information are stored in declarative memory. This means that the model contains knowledge on the relative magnitudes of each combination of payoff values.

Because Van Maanen et al. implemented the backward induction strategy, the model starts a second-order game by comparing its own payoff values in bins 3 and 4. First, the model retrieves

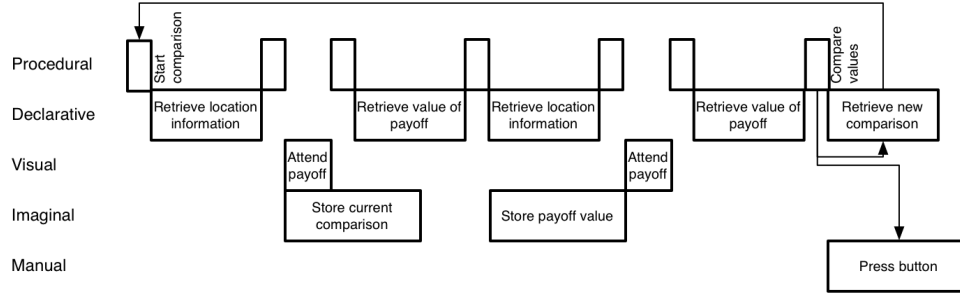


Figure 4: Flow-chart of the ACT-R model for solving Marble Drop game (figure from [MV10])

from declarative memory where the first of two payoffs is located on the screen (i.e., bin 4). If the model retrieves the location of bin 4, it attends bin 4 and tries to retrieve the value of the observed payoff. At the same time, the model frees the retrieval buffer for the upcoming payoff value and stores a chunk in the problem state buffer to represent the comparison that is currently made. Next, the model retrieves the location information for the other of the two payoffs that are currently compared. Again, the model frees the retrieval buffer (for the upcoming payoff value) and the payoff value of the first of two payoffs is stored in the problem state buffer. The problem state buffer can hold only one chunk at the same time and consequently the chunk that represents the current comparison is cleared from the problem state buffer. The location of the other payoff (i.e. bin 3) is attended and the corresponding value is retrieved from memory. Finally, the two payoff values are compared. The model tries to retrieve from declarative memory a chunk with the ordinal representation of both payoff values. Based on the outcome of this retrieval the model retrieves a next comparison.

For example, if the model's payoff value in bin 4 is greater than the model's payoff value in bin 3, the model will next compare the opponent's payoffs in bins 4 and 2. If the model's payoff value in bin 4 was less than the model's payoff value in bin 3, the model will next compare the opponent's payoffs in bins 3 and 2. The model continues to compare payoffs following the decision tree [HZ02] until it reaches the root of this tree. There, it decides its action based on the final comparison.

The model was tested against data from a Marble Drop task described by [MMRV10]. In the experiment the participants were asked to solve zero-order, first-order, and second-order Marble Drop problems. In all these conditions, participants were instructed to indicate the optimal first move as quickly as possible. That is, even in second-order games participants had to make only one choice. However, because the opponent always played rationally (and the participants were informed of this), there was always only one optimal choice.

As illustrated by figure 5, it turned out that the fit on the response time is very good ($R^2 = 1.0$; $RMSE = 0.42$ s). The fit on the accuracy data is slightly less ($RMSE = 0.067$, $R^2 = 0.2$), but this may be attributed to lack of data, making the estimated means less reliable. The model shows a very high proportion of correct responses, similar to the data of [MMRV10] (but contrary to [FVHK08]).

As the order of the Marble Drop reasoning problems increases, the model requires more time to respond. This is because more comparisons have to be made, and therefore more information has to be retrieved from declarative memory and stored in the problem state buffer. These steps take time, increasing the response time for higher-order reasoning problems. Because of the similar behavioral patterns between model and data, this study supports the view that participants in this

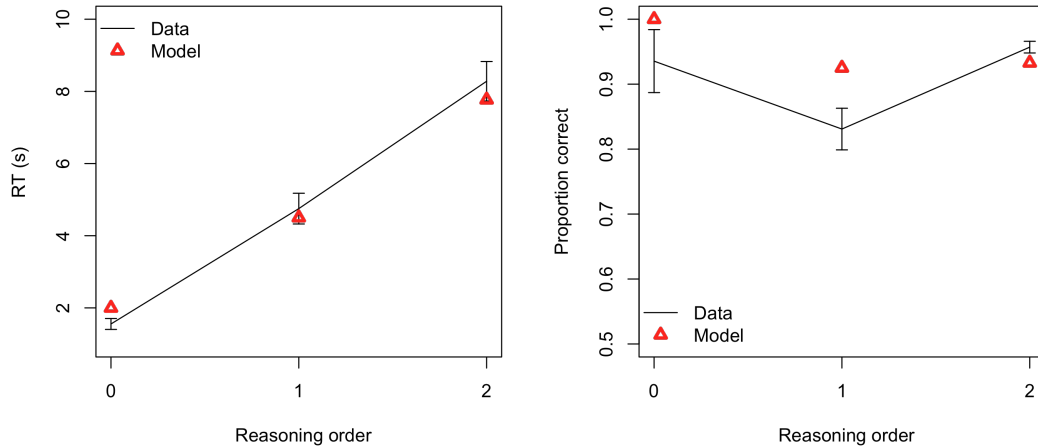


Figure 5: Model fit to data from [MMRV10]. Left: response time. Right: Accuracy (figure from [MV10])

task follow the same reasoning steps as the model does. That is, the hypothesis is supported that participants in a social reasoning game follow a decision tree to make the correct decision.

Another good validation of the model is to compare the sequence of spatial locations it attends with actual eye movements of participants during second-order social reasoning. Sequences of eye-movements (and more generally actions) help to determine the cognitive strategies involved in a task, and the similarity between the predicted and observed eye movements are indicative of the model’s validity [SA01].

2.5 Eye movements during second-order reasoning

Game-theoretically, participants are expected to use backward induction [OR94, Ros81]. In Marble Drop games, this would yield eye movements from right to left. However, the preliminary results from the eye-tracking study of [MvRV10] show opposite patterns: initially, participants seem to reason from left to right; they fixate at bins 1, 2, 3, and 4, in that order. Also, Figure 6 clearly shows that at position 1, the mean proportion of fixations is higher in bins 1 and 2, which correspond with the areas of interest (AOIs) A and B rather than in bins 3 and 4, which correspond with the AOIs C and D.

Accordingly, second-order reasoning seems to be context-driven. There is an evident left to right orientation in the Marble Drop context, as the white marble will run from left to right through a game for as long as both players continue the game, and the eye movements seem to follow that orientation.

Another finding that corroborates the idea of context-driven reasoning is that the proportion trends differ for some of the payoff structures (Figure 3). If participants had used backward induction, the eye movements would not vary, and occur independent of payoff structure. Differential proportion trends do fit context-driven reasoning, as it is susceptible to a *bottom-up* influence of orientation (left to right) and probably also features such as color-saliency (i.e., payoff value).

After an initial exploration of the context, participants will have to make some additional, backward comparisons if the intermediate reasoning steps have not yielded an optimal outcome (i.e., there is a darker, attainable marble in another bin than the bin that is the outcome of the last decision). Consider the payoff structure 3:1 (bin 1), 4:3 (bin 2), 1:2 (bin 3), and 2:4 (bin 4), with

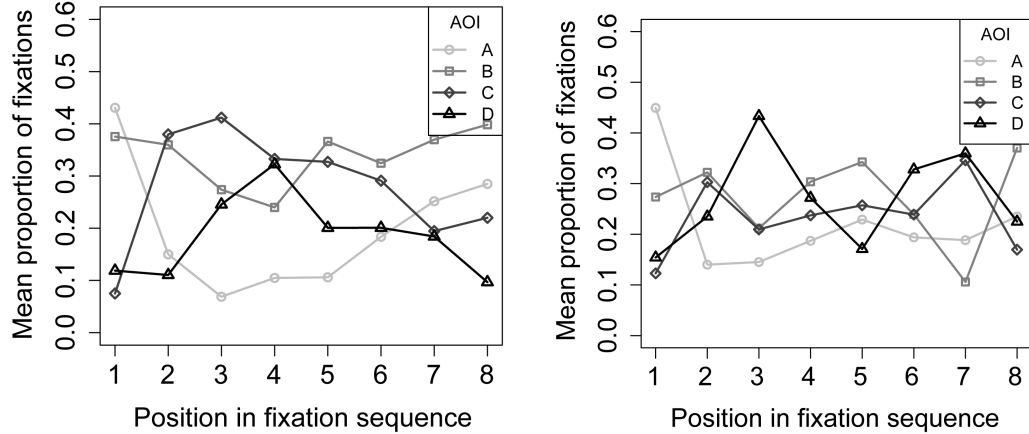


Figure 6: Mean proportion of fixations in bins 1 - 4, which correspond with the areas of interest (AOIs) A - D, as a function of position in the fixation sequence. The proportion trends differ for games in which a rational opponent would end the game and a rational participant would continue the game (a) and games in which both players would continue the game (b).

the first payoff value that of the participant and the second that of the opponent. A participant could reason forwardly: “I would go right (i.e., remove the right trapdoor) if the opponent ends the game at bin 2 (i.e., removes the left trapdoor)” and then “the opponent would end the game if I do not to continue the game from bin 3 to bin 4”. However, as the reasoning continues with “I would go to the right from bin 3 to bin 4”, a participant would have to reason backwards: “thus the opponent will not stop the game at bin 2” and therefore “if the value of my first payoff (at bin 1) is higher than that of my last payoff (at bin 4) I will stop the game, otherwise I will continue the game”.

3 A logical study

Following the lines of work in [RS08, PRS09], we now propose a language specifying strategies of players. This provides an elegant way to describe the empirical reasoning of the participants of the Marble Drop game (cf. Section 2.1.1), that has been found by the eye-tracking study in [MvRV10].

The basic ingredient that is needed for a logical system to model empirical reasoning of human agents, is to forego the usual assumption of *idealised* agents, but rather consider agents with limited computational and reasoning abilities. Though players with limited rationality are much more realistic to consider, for the time being, we only focus on *perfectly rational* players, whose only goal is to win the game. To model strategic reasoning of such resource-bounded players, we should note that these players are in general forced to strategize *locally* by selecting what part of the past history they choose to carry in their memory, and to what extent they can look ahead in their analysis. We consider the notion of *partial strategies* (formalised below) as a way to model such resource-bounded strategic reasoning.

3.1 Preliminaries

In this subsection, representations for extensive form games, game trees and strategies are presented, similar to those in [RS08, PRS09]. On the basis of these notations, reasoning strategies can be formalized in Subsection 3.2.

3.1.1 Extensive form games

Extensive form games are a natural model for representing finite games in an explicit manner. In this model, the game is represented as a finite tree where the nodes of the tree correspond to the game positions and edges correspond to moves of players. For this logical study, we will focus on game forms, and not on the games themselves, which come equipped with players' payoffs at the leaf nodes of the games. We present the formal definition below.

Let N denote the set of players; we use i to range over this set. For the time being, we restrict our attention to two player games, i.e. we take $N = \{1, 2\}$. We often use the notation i and \bar{i} to denote the players where $\bar{i} = 2$ when $i = 1$ and $\bar{i} = 1$ when $i = 2$. Let Σ be a finite set of action symbols representing moves of players, we let a, b range over Σ . For a set X and a finite sequence $\rho = x_1x_2 \dots x_m \in X^*$, let $last(\rho) = x_m$ denote the last element in this sequence.

3.1.2 Game trees

Let $\mathbb{T} = (S, \Rightarrow, s_0)$ be a tree rooted at s_0 on the set of vertices S and $\Rightarrow : (S \times \Sigma) \rightarrow S$ be a *partial* function specifying the edges of the tree. The tree \mathbb{T} is said to be finite if S is a finite set. For a node $s \in S$, let $\vec{s} = \{s' \in S \mid s \xrightarrow{a} s' \text{ for some } a \in \Sigma\}$. A node s is called a leaf node (or terminal node) if $\vec{s} = \emptyset$.

An **extensive form game tree** is a pair $T = (\mathbb{T}, \hat{\lambda})$ where $\mathbb{T} = (S, \Rightarrow, s_0)$ is a tree. The set S denotes the set of game positions with s_0 being the initial game position. The edge function \Rightarrow specifies the moves enabled at a game position and the turn function $\hat{\lambda} : S \rightarrow N$ associates each game position with a player. Technically, we need player labelling only at the non-leaf nodes. However, for the sake of uniform presentation, we do not distinguish between leaf nodes and non-leaf nodes as far as player labelling is concerned. An extensive form game tree $T = (\mathbb{T}, \hat{\lambda})$ is said to be finite if \mathbb{T} is finite. For $i \in N$, let $S^i = \{s \mid \hat{\lambda}(s) = i\}$ and let $frontier(\mathbb{T})$ denote the set of all leaf nodes of T .

A **play** in the game T starts by placing a token on s_0 and proceeds as follows: at any stage, if the token is at a position s and $\hat{\lambda}(s) = i$ then player i picks an action which is enabled for her at s , and the token is moved to s' where $s \xrightarrow{a} s'$. Formally a play in T is simply a path $\rho : s_0 a_0 s_1 \dots$ in \mathbb{T} such that for all $j > 0$, $s_{j-1} \xrightarrow{a_j} s_j$. Let $Plays(T)$ denote the set of all plays in the game tree T .

3.1.3 Strategies

A **strategy** for player i is a function μ^i which specifies a move at every game position of the player, i.e. $\mu^i : S^i \rightarrow \Sigma$. For $i \in N$, we use the notation μ^i to denote strategies of player i and $\tau^{\bar{i}}$ to denote strategies of player \bar{i} . By abuse of notation, we will drop the superscripts when the context is clear and follow the convention that μ represents strategies of player i and τ represents strategies of \bar{i} . A strategy μ can also be viewed as a subtree of T where for each node belonging to player i , there is a unique outgoing edge and for nodes belonging to player \bar{i} , every enabled move is included. Formally we define the strategy tree as follows: For $i \in N$ and a player i 's strategy $\mu : S^i \rightarrow \Sigma$, the strategy tree $T_\mu = (S_\mu, \Rightarrow_\mu, s_0, \hat{\lambda}_\mu)$ associated with μ is the least subtree of T satisfying the following property:

- $s_0 \in S_\mu$
- For any node $s \in S_\mu$,
 - if $\hat{\lambda}(s) = i$ then there exists a unique $s' \in S_\mu$ and action a such that $s \xrightarrow{a}_\mu s'$.
 - if $\hat{\lambda}(s) \neq i$ then for all s' such that $s \xrightarrow{a} s'$, we have $s \xrightarrow{a}_\mu s'$.

Let $\Omega^i(T)$ denote the set of all strategies for player i in the extensive form game tree T . A play $\rho : s_0 a_0 s_1 \dots$ is said to be consistent with μ if for all $j \geq 0$ we have $s_j \in S^i$ implies $\mu(s_j) = a_j$. A strategy profile (μ, τ) consists of a pair of strategies, one for each player.

3.1.4 Partial strategies

A partial strategy for player i is a partial function σ^i which specifies a move at some (and, not all) game positions of the player, i.e. $\sigma^i : S^i \rightarrow \Sigma$. Let \mathfrak{D}_{σ^i} denote the domain of the partial function σ^i . For $i \in N$, we use the notation σ^i to denote partial strategies of player i and $\pi^{\bar{i}}$ to denote partial strategies of player \bar{i} . When the context is clear, we refrain from using the superscripts. A partial strategy σ can also be viewed as a subtree of T where for some nodes belonging to player i , there is a unique outgoing edge and for other nodes belonging to player i as well as nodes belonging to player \bar{i} , every enabled move is included. Formally we define a partial strategy tree as follows: For $i \in N$ and a player i (partial) strategy $\sigma : S^i \rightarrow \Sigma$ the strategy tree $T_\sigma = (S_\sigma, \Rightarrow_\sigma, s_0, \hat{\lambda}_\sigma)$ associated with σ is the least subtree of T satisfying the following property:

- $s_0 \in S_\mu$
- For any node $s \in S_\mu$,
 - if $\hat{\lambda}(s) = i$ and $s \in \mathfrak{D}_\sigma$ then there exists a unique $s' \in S_\mu$ and action a such that $s \xrightarrow{a}_\mu s'$.
 - if $(\hat{\lambda}(s) = i$ and $s \notin \mathfrak{D}_\sigma)$ or $\hat{\lambda}(s) \neq i$ then for all s' such that $s \xrightarrow{a}_\mu s'$, we have $s \xrightarrow{a}_\mu s'$.

A partial strategy can be viewed as a set of total strategies. Given a partial strategy tree T_σ for a partial strategy σ for player i , a set of trees \widehat{T}_σ of total strategies can be defined as follows. A tree $T = (S, \Rightarrow, s_0, \hat{\lambda}) \in \widehat{T}_\sigma$ if and only if

- if $s \in T$ then for all $s' \in \vec{s}$, $s' \in T$ implies $s' \in T_\sigma$
- if $\hat{\lambda}(s) = i$ then there exists a unique $s' \in S_\sigma$ and action a such that $s \xrightarrow{a}_\sigma s'$.

Note that any total strategy is also viewed as a partial strategy, where the corresponding set of total strategies becomes a singleton set.

3.2 Strategy specifications

We now propose a syntax for specifying partial strategies and their compositions in a structural manner involving simultaneous recursion. The main case specifies, for a player, what conditions she tests for before making a move. The pre-condition for the move depends on observables that hold at the current game position as well as some simple finite past-time conditions and some finite look-ahead that each player can perform in terms of the structure of the game tree. Both the past-time and future conditions may involve some strategies that were or could be enforced by the players.

Below, for any countable set X , let $BPF(X)$ (the boolean, past and future combinations of the members of X) be sets of formulas given by the following syntax:

$$BPF(X) := x \in X \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \langle a^+ \rangle \psi \mid \langle a^- \rangle \psi.$$

where $a \in \Sigma$.

Formulas in $BPF(X)$ are interpreted at game positions. The formula $\langle a^+ \rangle \psi$ (respectively, $\langle a^- \rangle \psi$) talks about one step in the future (respectively, past). It asserts the existence of an a edge after (respectively, before) which ψ holds. Note that future (past) time assertions up to any bounded depth can be coded by iteration of the corresponding constructs. The “time free” fragment of $BPF(X)$ is formed by the boolean formulas over X . We denote this fragment by $Bool(X)$.

3.2.1 Syntax

Let $P^i = \{p_0^i, p_1^i, \dots\}$ be a countable set of observables for $i \in N$ and $P = \bigcup_{i \in N} P^i$. The syntax of strategy specifications is given by:

$$\text{Strat}^i(P^i) := [\psi \mapsto a]^i \mid \eta_1 + \eta_2 \mid \eta_1 \cdot \eta_2.$$

where $\psi \in \text{BPF}(P^i)$.

The idea is to use the above constructs to specify properties of strategies as well as to combine them to describe a **play** of the game. For instance the interpretation of a player i 's specification $[p \mapsto a]^i$ where $p \in P^i$, is to choose move “ a ” at every game position belonging to player i where p holds. At positions where p does not hold, the strategy is allowed to choose any enabled move. The strategy specification $\eta_1 + \eta_2$ says that the strategy of player i conforms to the specification η_1 or η_2 . The construct $\eta_1 \cdot \eta_2$ says that the strategy conforms to specifications η_1 and η_2 .

Let $\Sigma = \{a_1, \dots, a_m\}$, we also make use of the following abbreviation.

$$\bullet \text{ null}^i = [\top \mapsto a_1] + \dots + [\top \mapsto a_m].$$

It will be clear from the semantics (which is defined shortly) that any strategy of player i conforms to null^i , or in other words this is an empty specification. The empty specification is particularly useful for assertions of the form “there exists a strategy” where the property of the strategy is not of any relevance.

3.2.2 Semantics

Let $M = (T, V)$ where $T = (S, \Rightarrow, s_0, \hat{\lambda})$ is an extensive form game tree and $V : S \rightarrow 2^P$ a valuation function. The truth of a formula $\psi \in \text{BPF}(P)$ at the state s , denoted $M, s \models \psi$, is defined as follows:

- $M, s \models p$ iff $p \in V(s)$.
- $M, s \models \neg\psi$ iff $M, s \not\models \psi$.
- $M, s \models \psi_1 \vee \psi_2$ iff $M, s \models \psi_1$ or $M, s \models \psi_2$.
- $M, s \models \langle a^+ \rangle \psi$ iff there exists a s' such that $s \xrightarrow{a} s'$ and $M, s' \models \psi$.
- $M, s \models \langle a^- \rangle \psi$ iff there exists a s' such that $s' \xrightarrow{a} s$ and $M, s' \models \psi$.

Strategy specifications are interpreted on strategy trees of T . We assume the presence of two special propositions **turn**₁ and **turn**₂ that specify which player's turn it is to move, i.e. the valuation function satisfies the property

- for all $i \in N$, **turn** _{i} $\in V(s)$ iff $\lambda(s) = i$.

One more special proposition **root** is assumed to indicate the root of the game tree, that is the starting node of the game. The valuation function satisfies the property

- **root** $\in V(s)$ iff $s = s_0$.

Recall that a partial strategy σ of player i can be viewed as a set of total strategies of the player and each such strategy is a subtree of T .

The semantics of the strategy specifications are given as follows. Given the game tree $T = (S, \Rightarrow, s_0, \hat{\lambda})$ and a partial strategy specification $\eta \in \text{Strat}^i(P^i)$, we define a function $\llbracket \cdot \rrbracket_T : \text{Strat}^i(P^i) \rightarrow 2^{\Omega^i(T)}$, where each partial strategy specification is associated with a set of total strategy trees.

For any $\eta \in \text{Strat}^i(P^i)$, $\llbracket \cdot \rrbracket_T$ is defined inductively as follows:

- $\llbracket [\psi \mapsto a]^i \rrbracket_T = \Upsilon \in 2^{\Omega^i(T)}$ satisfying: $\mu \in \Upsilon$ iff μ satisfies the condition that, if $s \in S_\mu$ is a player i node then $M, s \models \psi$ implies $out_\mu(s) = a$.
- $\llbracket \eta_1 + \eta_2 \rrbracket_T = \llbracket \eta_1 \rrbracket_T \cup \llbracket \eta_2 \rrbracket_T$
- $\llbracket \eta_1 \cdot \eta_2 \rrbracket_T = \llbracket \eta_1 \rrbracket_T \cap \llbracket \eta_2 \rrbracket_T$

Above, $out_\mu(s)$ is the unique outgoing edge in μ at s . Recall that s is a player i node and therefore by definition there is a unique outgoing edge at s .

3.2.3 Response and future planning of players

Modelling a player’s response to the opponent’s play is one of the basic notions that we need to deal with while describing reasoning in games. To this end, we introduce one more construct in our language of pre-conditions $BPF(X)$. The idea is to model the phenomenon that if player \bar{i} has played according to π in the history of the game, player i responds following some strategy σ , say. We may also describe that since player \bar{i} may play according to π at a certain future point of the game (if it so happens that the game reaches that point), in anticipation to which player i can now play according to σ .

To model such scenarios we introduce the formula $\bar{i}?\zeta$ in the syntax of $BPF(P^i)$. The intuitive reading of the formula is “player \bar{i} is playing according to a partial strategy conforming to the specification ζ at the current stage of the game”, and the semantics is given by,

- $M, s \models \bar{i}?\zeta$ iff $\exists T'$ such that $T' \in \llbracket \zeta \rrbracket_T$ and $s \in T'$.

Note that this involves simultaneous recursion in the definitions of $BPF(X)$ and $Strat^i(P^i)$. The framework introduced by Ramanujam and Simon [RS08] has a more simpler version of $BPF(P^i)$, where only past formulas are considered, but they introduce an additional construct in the syntax of strategy specifications, viz. $\pi \Rightarrow \sigma$, which says that at any node player i plays according to the strategy specification σ if on the history of the play, all the moves made by \bar{i} conforms to π . The introduction of the formula $\bar{i}?\zeta$ in the language of $BPF(P^i)$ empowers us to model notions expressed by the specification $\pi \Rightarrow \sigma$. We leave the detailed technicalities involving our proposal as well as the comparative discussion with the related framework of [RS08] for future work.

3.3 Marble Drop game: a test case

We are now well-equipped to express the empirical reasoning performed by the participants of the Marble drop game described in Section 2.1.1. The game form is structurally equivalent to the Centipede game tree given in Figure 7.

Using the strategy specification language introduced in Section 3.2, we express the different reasoning methods of participants that have been validated by the experiments described in Section 2.5. The reasoning is carried out by an outside agent (participant) regarding the question:

How would the players 1 and 2 play in the game, under the assumptions that both players are rational (thus will try to maximize their utility), and that there is common knowledge of rationality among the players.

In particular, we encode three different types of reasoning. Note that we are not talking about why the players reason in some way, i.e. what considerations lead them to such reasoning, (e.g. in the marble drop game, the darker marble may lie at the first bin itself or may be at the third bin) but rather how they can go about playing the game. Combining the “why” and “how” at this level is something we leave for future work:

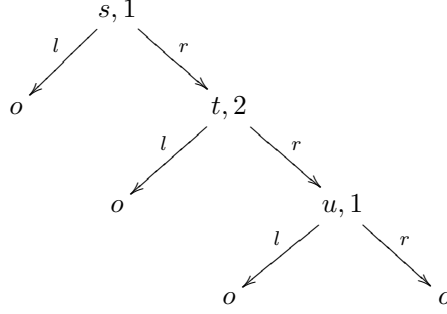


Figure 7: Centipede game tree

- forward reasoning: $\langle [1?[\mathbf{root} \wedge \mathbf{turn}_1 \mapsto r]^1 \wedge \langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_2 \mapsto r]^2, [2?[\langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_2 \mapsto r]^2 \wedge \langle r^- \rangle \langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_1 \mapsto l]^1 \rangle$.

if player 1 makes the move r at the root node, player 2 will respond with playing r , and if player 2 behaves like that, player 1 would play l .

- backward reasoning: $\langle [1?[\langle r^- \rangle \langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_1 \mapsto r]^1 \wedge \mathbf{turn}_2 \mapsto r]^2, [2?[\langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_2 \mapsto r]^2 \wedge \mathbf{root} \wedge \mathbf{turn}_1 \mapsto l]^1 \rangle$.

if player 1 makes the move r at the u node (if the game reaches there) , player 2 will play r when her turn comes, and if player 2 behaves like that, player 1 would play l at the start node.

- combined reasoning: $\langle [\mathbf{root} \wedge \mathbf{turn}_1 \mapsto r]^1, [1?[\langle r^- \rangle \langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_1 \mapsto r]^1 \wedge \langle r^- \rangle \mathbf{root} \wedge \mathbf{turn}_2 \mapsto r]^2 \rangle$.

player 1 would play l at the root node, and then player 2 will play r after that, since if the game reaches the u node, player 1 will play r .

The partial strategy profiles of the two players describe in each case what sort of reasoning players 1 and 2 might perform in the course of the game so as to gain maximum benefits. As apparent from the descriptions of such reasoning, this really captures the arbitrary as well as methodical reasoning that humans can do when confronted with such games, as exemplified (to a certain extent) in Section 2.5. In fact, the combined reasoning suggests that player 1 might make an arbitrary move at the beginning, which can often be the case in reality. As found in the eye-tracking study of [MvRV10], the human participants follow different procedures of reasoning (in particular, first a forward scan of the context, followed if necessary by backward reasoning), rather than the backward induction strategy prescribed by game theorists. These reasoning procedures are adequately described by the (partial) strategy specification language proposed in Section 3.2. This formal representation can provide the building blocks for a better cognitive model based on the ACT-R architecture, in order to construct precise computational cognitive models for the varied thought processes of human agents.

4 Discussion and future work

To put this first attempt at bridge-building between logic and experimental work on games and strategies into perspective, it may be fruitful to keep in mind the three levels of inquiry for cognitive

science that David Marr characterized [Mar82]:

- identification of the information-processing agents task as an input - output function: the computational level;
- specification of an algorithm which computes the function: the algorithmic level;
- physical/neural implementation of the algorithm specified: the implementation level.

Researchers aiming to answer the question what logical theories may contribute to the study of resource-bounded strategic reasoning could be disappointed when it turns out that logic is not the best vehicle to describe such reasoning at the implementation level. Still, logic surely makes a contribution at Marr's first computational level by providing a precise specification language for cognitive processes. Quite possibly, logic may also have a fruitful role to play in theories of resource-bounded strategic reasoning at the algorithmic level, in the construction of computational cognitive models in ACT-R. A first step in this direction was made in the previous subsection.

Future work would be to distinguish several possible reasoning strategies for resource-bounded agents in games. For example, Van Maanen et al. propose a new ACT-R model that predicts how humans perform in experiments with a dual task done in parallel to the Marble Drop game [MV10]. That ACT-R model presumes a reasoning strategy following the decision tree analysis of Hedden and Zhang [HZ02]. It would be interesting to also test the predictions of alternative models that correspond to different reasoning strategies, for example, the forward, backward and combined ones introduced in the previous section.

It would also be interesting to define reasoning strategies for games that consist of many more steps and construct corresponding ACT-R models to drive new experimental work.

The great advantage of coupling a strategy logic to ACT-R is that ACT-R already implements very precise, experimentally validated theories about human memory and cognitive bounds on reasoning processes. Thus, there is no need to add (possibly arbitrary) resource bounds in the logical language. The combined strengths of logic, coupled with cognitive modeling and ACT-R, will hopefully lead to an improved understanding of human resource-bounded reasoning in games.

From the logical perspective, providing a sound and complete system for strategic reasoning that models empirical human reasoning will be the essential next step. We would need to take players' preferences into consideration as well as intentions of other players. Evidently, reasoning about intentions is essential for forward induction and solution concepts like extensive-form rationalizability and refinements of sequential equilibrium.

Acknowledgements:

The authors would like to thank the anonymous referees for their insightful comments which helped in the discussions throughout the paper. They also thank Leendert van Maanen and Hedderick van Rijn for the helpful comments regarding the experimental work and cognitive architecture. The first author acknowledges NWO grant # 600.065.120.08N201 and the second and third authors acknowledge NWO grant # 227-80-001.

References

- [ACB07] H. Arló-Costa and C. Bicchieri. Knowing and supposing in games of perfect information. *Studia logica*, 86:353373, 2007.
- [And07] J.R. Anderson. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press, New York (NY), 2007.

- [Art09] S. Artemov. Knowledge-based rational decisions. Technical report, The CUNY Graduate Center, 2009.
- [Aum95] Robert J. Aumann. Backward induction and common knowledge of rationality. *Games Econ. Behav.*, 8(1):6–19, 1995.
- [Bic88] C. Bicchieri. Common knowledge and backward induction: a solution to the paradox. In *TARK '88: Proceedings of the 2nd conference on Theoretical aspects of Reasoning about Knowledge*, pages 381–393, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.
- [Bin96] Ken Binmore. A note on backward induction. *Games and Economic Behavior*, 17(1):135–137, November 1996.
- [Bon02] Giacomo Bonanno. Modal logic and game theory: two alternative approaches. *Risk, Decision and Policy*, 7(03):309–324, December 2002.
- [Bra07] A. Brandenburger. The power of paradox: Some recent developments in interactive epistemology. *International Journal of Game Theory*, 35:465–492, 2007.
- [BSZ09] A. Baltag, S. Smets, and J. Zvesper. Keep 'hoping' for rationality: A solution to the backward induction paradox. *Synthese*, 169(2):301–333, 2009.
- [Cam03] C. F. Camerer, editor. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, Princeton, 2003.
- [FVHK08] L. Flobbe, R. Verbrugge, P. Hendriks, and I. Krämer. Children's application of theory of mind in reasoning and language. *Journal of Logic, Language and Information*, 17:417–442, 2008. Special issue on formal models for real people, edited by M. Coughlan.
- [GT99] G. Gigerenzer and P.M. Todd. *Simple Heuristics that make us Smart*. Oxford University Press, New York, 1999.
- [HP09] J. Y. Halpern and R. Pass. Iterated regret minimization: A new solution concept. In C. Boutilier, editor, *Proceedings of IJCAI-09: 21st International Joint Conferences on Artificial Intelligence, Pasadena, CA, USA*, pages 153–158, 2009.
- [HvdHMMW03] B.P. Harrenstein, W. van der Hoek, J.-J. Ch. Meyer, and C. Witteveen. A modal characterization of Nash equilibrium. *Fundamenta Informaticae*, 57(2):281–321, 2003.
- [HZ02] T. Hedden and J. Zhang. What do you think I think you think? Strategic reasoning in matrix games. *Cognition*, 85:1–36, 2002.
- [JCSR02] E. J. Johnson, C. F. Camerer, S. Sen, and T. Rymon. Detecting failures of backward induction: Monitoring information search in sequential bargaining. *Journal of Economic Theory*, 104(1):16–47, 2002.
- [Jon80] A. Jones. *Game theory: Mathematical Models of Conflict*. John Wiley, 1980.
- [JT07] I. Juvina and N. A. Taatgen. Modeling control strategies in the n-back task. In *Proceedings of the 8th International Conference on Cognitive Modeling*, New York, NY, 2007. Psychology Press.

- [JvdH04] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63, 2004.
- [KMRW82] D. M. Kreps, P. Milgrom, J. Roberts, and R. Wilson. Rational cooperation in the finitely repeated prisoners' dilemma. *Journal of Economic Theory*, 27(2):245–252, 1982.
- [Lov05] Marsha C. Lovett. A strategy-based interpretation of Stroop. *Cognitive Science*, 29(3):493–524, 2005.
- [LWW00] C. Lebiere, D. Wallach, and R. West. A memory-based account of the prisoner's dilemma and other 2x2 games. In N.A. Taatgen and J. Aasman, editors, *Proceedings of Third International Conference on Cognitive Modeling*, pages 185–193, Veenendaal, 2000. Universal Press.
- [Mar82] D. Marr. *Vision*. Freeman and Company, New York, 1982.
- [Mic67] J. A. Michon. The game of JAM—an isomorph of Tic-Tac-Toe. *American Journal of Psychology*, 80(1):137–140, 1967.
- [MMRV10] B. Meijering, L.. van Maanen, H. van Rijn, and R. Verbrugge. The facilitative effect of context on second-order social reasoning. In *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society*, Cognitive Science Society, 2010.
- [MO91] K.I. Manktelow and D.E. Over. Social roles and utilities in reasoning with deontic conditionals. *Cognition*, 39(2):85–105, 1991.
- [MP92] R.D. McKelvey and T.R. Palfrey. An experimental study of the centipede game. *Econometrica*, 60(4):803–836, 1992.
- [MV10] L.. van Maanen and R. Verbrugge. A computational model of second-order social reasoning. In *Proceedings of the 10th International Conference on Cognitive Modeling*, 2010.
- [MvRV10] B. Meijering, H. van Rijn, and R. Verbrugge. The facilitative effect of context on second-order social reasoning. Technical report, University of Groningen, 2010. (in prep).
- [OR94] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.
- [Per10] A. Perea. Backward induction versus forward induction reasoning. *Games*, 1(3):168–188, 2010.
- [Pol45] G. Polya. *How to Solve It. A New Aspect of Mathematical Method*. Princeton University Press, Princeton, N.J., 1945.
- [PRS09] S. Paul, R. Ramanujam, and S. Simon. Stability under strategy switching. In *Proceedings of the 5th Conference on Computability in Europe (CiE 2009)*, LNCS 5635, pages 389–398. Springer, 2009.
- [Ros81] R.W. Rosenthal. Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economic theory*, 25(1):92–100, 1981.

- [RS08] R. Ramanujam and S. Simon. A logical structure for strategies. In *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, volume 3 of *Texts in Logic and Games*, pages 183–208. Amsterdam University Press, Amsterdam University Press, 2008.
- [SA01] D. D. Salvucci and J. R. Anderson. Automated eye-movement protocol analysis. *Human-Computer Interaction*, 16(1):39–86, 2001.
- [Sim79] H.A. Simon. Information processing models of cognition. *Annual Review of Psychology*, 30(1):363–396, 1979.
- [Sta96] R. Stalnaker. On the evaluation of solution concepts. *Theory and Decision*, 37:49–73, 1996.
- [SvL04] K. Stenning and M. van Lambalgen. A little logic goes a long way: basing experiment on semantic theory in the cognitive science of conditional reasoning. *Cognitive Science*, 28(4):481–529, 2004.
- [vW03] W. van der Hoek and M. Wooldridge. Time, knowledge and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75 (1):125–157, 2003.
- [WBR76] D. Wood, J. S. Bruner, and G. Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, 1976.
- [Wei84] J. S. Weitzenfeld. Valid reasoning by analogy. *Philosophy of Science*, 51(1):137–149, 1984.
- [WLB06] R.L. West, C. Lebiere, and D.J. Bothell. Cognitive architectures, game playing, and human evolution. In R. Sun, editor, *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, pages 103–123. Cambridge University Press, New York (NY), 2006.
- [WS71] P.C. Wason and D. Shapiro. Natural and contrived experience in a reasoning problem. *The Quarterly Journal of Experimental Psychology*, 23(1):63–71, 1971.

An extension of RB-ATL

Nguyen Hoang Nga

Abstract

RB-ATL is a logic to specify coalitional properties of multiagent systems where actions cost certain amount of resources. RB-ATL allows us to express properties such as within a limited amount of resources, a group of agents can produce a particular result but not another. This paper extends RB-ATL so that it is possible to express coalitional properties where we only consider the limitation over a subset of all resources.

1 Introduction

In previous work [2], we have presented a logic, namely RB-ATL, which allows expressing and reasoning about properties of coalitional ability under resource constraints. The logic extended ATL [3] where a bound over resources is added into each cooperation modality. Each resource bound, defined as a vector, specifies the upper bound over each resource available to (or willing to contribute by) an agent or a group of agents. This means it is not possible to express properties in RB-ATL where the upper bound over a subset of resources is of interest.

For example, let us consider a set of two resources: memory and network bandwidth. It is possible to express in RB-ATL the property that *Agents 1 and 2 can enforce p to become true without using more than 4 units of memory and 2 units of network bandwidth* by the formula $\langle\langle\{1, 2\}^{(4,2)}\rangle\rangle\top\mathcal{U}p$. However, if we would like to express the same property except we do not care about how much network bandwidth can be used, it is not possible to do so in RB-ATL unless the language allows infinite disjunction. For this reason, the property can only be written as $\bigvee_{n \geq 0} \langle\langle\{1, 2\}^{(4,n)}\rangle\rangle\top\mathcal{U}p$. In order to express such properties, in this paper, we extend RB-ATL by allowing the inclusion of an extra symbol ∞ in the resource bounds. The idea is that whenever no limit is required over a resource, we can set the bound for this resource as ∞ . We also show that the resulting logic RBATL^∞ is sound and complete.

The remainder of this paper is structured as follows. We first briefly recall the syntax and semantics of RB-ATL. Then, we introduce the syntax and semantics

of the extended logic RBATL^∞ . After that, we give a complete axiomatization followed by a sketch proof of the completeness. At the end is the section of related work and conclusion.

2 Resource-bounded ATL

As RBATL^∞ is based on RB-ATL [2], we first briefly recall RB-ATL. RB-ATL extended ATL in order to allow expressing properties of coalitional ability under limited amounts of resources. Bounds on resources are defined as vectors of numbers each of which corresponds to the bound on a type of resources. If we fix a finite set of resources r , then the set of bounds is defined as $\mathbb{B} = \mathbb{N}^{|r|}$. Given a set of propositions Φ and a finite set of agents N , formulas of RB-ATL are defined by the following syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \psi \mid \langle\langle A^b \rangle\rangle\bigcirc\varphi \mid \langle\langle A^b \rangle\rangle\varphi\mathcal{U}\psi \mid \langle\langle A^b \rangle\rangle\Box\varphi$$

where $p \in \Phi$, $A \subseteq N$ and $b \in \mathbb{B}$. Intuitively, $\langle\langle A^b \rangle\rangle\bigcirc\varphi$ says that the coalition A can co-operate (in one step) to make φ true without spending more than b amount of resources. Similarly, $\langle\langle A^b \rangle\rangle\varphi\mathcal{U}\psi$ means that the coalition A can co-operate (in many consecutive steps) to eventually make ψ true without spending more than b amount of resources while keeping φ true; and $\langle\langle A^b \rangle\rangle\Box\varphi$ says that the coalition A can co-operative from now on to guaranty that φ is true without spending more than b amount of resources.

For convenience, we only study the soundness and completeness of the normal form version of RB-ATL whose syntax is as follows:

$$\varphi ::= (\neg)p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid (\neg)\langle\langle A^b \rangle\rangle\bigcirc\varphi \mid (\neg)\langle\langle A^b \rangle\rangle\varphi\mathcal{U}\psi \mid (\neg)\langle\langle A^b \rangle\rangle\Box\varphi$$

For short, normal form RB-ATL are referred to as RB-ATL in the rest of this paper. We also use $\sim\varphi$ to denote the equivalent normal form formula of $\neg\varphi$.

Semantics of RB-ATL is defined in terms of Resource-bounded Concurrent Game Structures (RB-CGS) together with the notions of move, co-move, strategy and co-strategy. A Resource-bounded Concurrent Game Structure (RB-CGS) is a tuple $S = (n, Q, \Pi, \pi, d, c, \delta)$ where:

- $n \geq 1$ is the number of players (agents), we denote the set of players $\{1, \dots, n\}$ by N
- Q is a non-empty set of states
- Π is a finite set of propositional variables

- $\pi : Q \rightarrow \wp(\Pi)$ is a mapping which assigns each state in Q a subset of propositional variables
- $d : Q \times N \rightarrow \mathbb{N}$ is a mapping to indicate the number of available moves (actions) for each player $a \in N$ at a state $q \in Q$ such that $d(q, a) \geq 1$. At each state $q \in Q$, we denote the set of joint moves available for all players in N by $D(q)$. That is

$$D(q) = \{1, \dots, d(q, 1)\} \times \dots \times \{1, \dots, d(q, n)\}$$

- $c : Q \times N \times \mathbb{N} \rightarrow \mathbb{B}$ is a mapping to indicate the minimal amount of resources required by each move available to each agent at a specific state.
- $\delta : Q \times \mathbb{N}^n \rightarrow Q$ is a mapping which assigns the next state of the system after agents perform a joint move in $D(q)$ from a state.

From the definition of RB-CGSs, resource bounds are used to specify the costs of actions performed by agents. In order to express the spending of a subset of agents in one or many steps, we aggregate the spending of each agent in the subset. This type of aggregation is called parallel aggregation. Another type of aggregation is called serial where an agent (coalition) performs some (joint) action and then performs another (joint) action; the aggregative spending of the agent (coalition) after the two steps is the serial aggregation of the costs of the two (joint) actions. For simplicity, we assume in this paper that both types of aggregations are defined as addition.

Given an RB-CGS $S = (n, Q, \Pi, \pi, d, c, \delta)$, we denote the set of infinite sequences of states by Q^ω as usual. Let $\lambda = q_0q_1\dots \in Q^\omega$, we denote $\lambda[i] = q_i$ and $\lambda[i, j] = q_i\dots q_j$. A **move** for a coalition $A \subseteq N$ at a state $q \in Q$ is a tuple $\sigma_A = (\sigma_a)_{a \in A}$ such that $1 \leq \sigma_a \leq d(q, a)$. For convenience, we denote $D_A(q)$ to be the set of all moves for A at q . Furthermore, given $m \in D(q)$, we denote $m_A = (m_a)_{a \in A}$. Then we define the set of all possible outcomes by a move $\sigma_A \in D_A(q)$ at a state q as follows

$$out(q, \sigma_A) = \{q' \in Q \mid \exists m \in D(q) : m_A = \sigma_A \wedge q' = \delta(q, m)\}$$

The cost of a move $\sigma_A \in D_A(q)$ then is defined as $cost(q, \sigma_A) = \sum_{a \in A} c(q, a, \sigma_a)$.

A **strategy** for a coalition $A \subseteq N$ is a mapping F_A which associates each sequence $\lambda q \in Q^+$ to a move in $D_A(q)$. A computation $\lambda \in Q^\omega$ is consistent with F_A iff for all $i \geq 0$, $\lambda[i+1] \in out(\lambda[i], F_A(\lambda[0, i]))$. We denote by $out(q, F_A)$ the set of all such sequences λ starting from q , i.e. $q = \lambda[0]$. Given a bound $b \in \mathbb{B}$, a computation $\lambda \in out(q, F_A)$ is b -consistent with F_A iff, for every $i \geq 0$, $\sum_{j=0}^i cost(\lambda[j], F_A(\lambda[0, j])) \leq b$. We denote $out(q_0, F_A, b)$ the set of all such

sequences. Then, a strategy F_A is a b -strategy iff $out(q, F_A) = out(q, F_A, b)$ for any $q \in Q$.

Similarly to the case of moves and strategies, we define a **co-move** for a coalition $A \subseteq N$ as a mapping $\sigma_A^c : D_A(q) \rightarrow Q$ such that $\sigma_A^c(\sigma_A) \in out(q, \sigma_A)$ for any $\sigma_A \in D_A(q)$. We denote $D_A^c(q)$ be the set of all co-moves for A at a state $q \in Q$. A state q' is consistent with a co-move σ^c iff there is some move σ_A such that $\sigma^c(\sigma_A) = q'$. We define the set of consistent outcomes for a co-move σ^c by

$$out(q, \sigma^c) = \{q' \in Q \mid q' \text{ is consistent with } \sigma^c\}$$

Given a bound $b \in \mathbb{B}$, a state q' is b -consistent with a co-move σ^c at q iff there is some move $\sigma_A \in D_A(q)$ with $cost(q, \sigma_A) \leq b$ such that $\sigma^c(\sigma_A) = q'$. We denote the set of b -consistent outcomes for a co-move σ^c by

$$out(q, \sigma^c, b) = \{q' \in Q \mid q' \text{ is } b\text{-consistent with } \sigma^c \text{ at } q\}$$

A **co-strategy** for a coalition $A \subseteq N$ is a mapping F_A^c which assigns each sequence $\lambda q \in Q^+$ to a co-move in $D_A^c(q)$. We say a computation $\lambda \in Q^\omega$ is consistent with F_A^c iff, for all $i \geq 0$, $\lambda[i+1] \in out(\lambda[i], F_A^c(\lambda[0, i]))$. Let us define $out(q, F_A^c)$ to be the set of all such sequences where $q = \lambda[0]$. We say a computation $\lambda \in out(q, F_A^c)$ is b -consistent with F_A^c iff, for all $i \geq 0$, there is a sequence of moves $\sigma_A^0 \in D_A(\lambda[0]), \dots, \sigma_A^i \in D_A(\lambda[i])$ such that $\lambda_{j+1} = F_A^c(\lambda[0, j])(\sigma_A^j)$ for all $j = 0, \dots, i$ and $\sum_j cost(\lambda[j], \sigma_A^j) \leq b$. Let us denote $out(q, F_A^c, b)$ be the set of all such sequences where $q = \lambda[0]$.

The truth of a RB-ATL formula φ at a state q of a RB-CGS S is defined by induction on the structure of φ . We ignore the propositional cases, other cases are listed as follows:

- $S, q \models \langle\langle A^b \rangle\rangle \bigcirc \varphi$ iff there exists a b -strategy F_A such that for all $\lambda \in out(q, F_A)$, $S, \lambda[1] \models \varphi$ iff there is a move $\sigma_A \in D_A(q)$ such that for all $q' \in out(\sigma_A)$, $S, q' \models \varphi$
- $S, q \models \neg \langle\langle A^b \rangle\rangle \bigcirc \varphi$ iff there exists a co-strategy F_A^c such that for all $\lambda \in out(q, F_A, b)$, $S, \lambda[1] \models \sim \varphi$ iff there is a co-move $\sigma^c \in D_A^c(q)$ such that for all $\sigma_A \in D_A(q)$ and $cost(\sigma_A) \leq b$, $S, \sigma^c(\sigma_A) \models \sim \varphi$
- $S, q \models \langle\langle A^b \rangle\rangle \square \varphi$ iff there exists a b -strategy F_A for any $\lambda \in out(q, F_A)$, $S, \lambda[i] \models \varphi$ for all $i \geq 0$
- $S, q \models \neg \langle\langle A^b \rangle\rangle \square \varphi$ iff there exists a co-strategy F_A^c for any $\lambda \in out(q, F_A^c, b)$, $S, \lambda[i] \models \varphi$ for all $i \geq 0$

- $S, q \models \langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a b -strategy F_A such that for all $\lambda \in \text{out}(q, F_A)$, there is a position $i \geq 0$ such that $S, \lambda[i] \models \psi$ and $S, \lambda[j] \models \psi$ for all $j \in \{0, \dots, i-1\}$
- $S, q \models \neg \langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a co-strategy F_A^c such that for all $\lambda \in \text{out}(q, F_A, b)$, either $S, \lambda[i] \models \psi$ for all $i \geq 0$ or if there is a position $i \geq 0$ such that $S, \lambda[i] \models \psi$ then there exists $0 \leq j < i$ such that $S, \lambda[j] \models \sim \varphi$

3 Syntax and semantics of RBATL[∞]

We define the set of resource bounds with infinity as $\mathbb{B}^\infty = (\mathbb{N} \cup \{\infty\})^{|\mathcal{r}|}$. To make addition and comparison over resource bounds compatible with infinity, we extend them to include the case ∞ as follows:

$$\begin{aligned}
n + \infty &= n + \infty = \infty \\
\infty + \infty &= \infty \\
n &\leq \infty \\
\infty &\leq \infty
\end{aligned}$$

where $n \in \mathbb{N}$. Notice that costs of actions in RB-CGSs are still defined by \mathbb{B} while bounds appearing in RB-ATL formulas may include infinity. The syntax of (normal form) RBATL[∞] is defined as follows.

$$\varphi ::= (\neg)p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid (\neg)\langle\langle A^b \rangle\rangle \bigcirc \varphi \mid (\neg)\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi \mid (\neg)\langle\langle A^b \rangle\rangle \square \varphi$$

where $p \in \Phi$, $b \in \mathbb{B}^\infty$ and $A \subseteq N$. We define the $\langle\langle \emptyset^b \rangle\rangle \bigcirc$ modality as the dual one of $\langle\langle N^b \rangle\rangle \bigcirc$, i.e. $\langle\langle \emptyset^b \rangle\rangle \bigcirc \varphi \equiv \neg \langle\langle N^b \rangle\rangle \bigcirc \sim \varphi$. This modality is to say that a system of multiple agents cannot avoid some thing if they are not allowed to spend more than some b amount of resources.

The semantics of RBATL[∞] is also defined by means of RB-CSGs. However, we need to extend the notion of b -consistency of strategies to include the case where $b \in \mathbb{B}^\infty$. Given a bound $b \in \mathbb{B}^\infty$ and a strategy F_A , a computation $\lambda \in \text{out}(q, F_A)$ is b -consistent with F_A iff, for every $i \geq 0$, $\sum_{j=0}^i \text{cost}(\lambda[j], F_A(\lambda[0, i])) \leq b$. We denote $\text{out}(q_0, F_A, b)$ the set of all such sequences. Then, a strategy F_A is a b -strategy iff $\text{out}(q, F_A) = \text{out}(q, F_A, b)$ for any $q \in Q$. Similar extensions are also applied for the case of co-moves and co-strategies.

Given a RB-CGS $S = (n, Q, \Pi, \pi, d, c, \delta)$, the truth of a RBATL[∞] formula is defined inductively as follows where A is a non-empty coalition:

- $S, q \models p$ iff $p \in \pi(q)$
- $S, q \models \neg p$ iff $p \notin \pi(q)$

- $S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$
- $S, q \models \varphi \wedge \psi$ iff $S, q \models \varphi$ and $S, q \models \psi$
- $S, q \models \langle\langle A^b \rangle\rangle \bigcirc \varphi$ iff there exists a b -strategy F_A such that for all $\lambda \in \text{out}(q, F_A)$, $S, \lambda[1] \models \varphi$. In other words, it is equivalent to say that there is a move $\sigma_A \in D_A(q)$ such that for all $q' \in \text{out}(\sigma_A)$, $S, q' \models \varphi$
- $S, q \models \neg \langle\langle A^b \rangle\rangle \bigcirc \varphi$ iff there exists a co-strategy F_A^c such that for all $\lambda \in \text{out}(q, F_A, b)$, $S, \lambda[1] \models \sim \varphi$. In other words, it is equivalent to say that there is a co-move $\sigma^c \in D_A^c(q)$ such that for all $\sigma_A \in D_A(q)$ and $\text{cost}(\sigma_A) \leq b$, $S, \sigma^c(\sigma_A) \models \sim \varphi$
- $S, q \models \langle\langle \emptyset^b \rangle\rangle \bigcirc \varphi$ iff for every b -strategy F_N for all $\lambda \in \text{out}(q, F_A)$, $S, \lambda[1] \models \varphi$
- $S, q \models \neg \langle\langle \emptyset^b \rangle\rangle \bigcirc \varphi$ iff there is a b -strategy F_N such that for all $\lambda \in \text{out}(q, F_N)$, $S, \lambda[1] \models \sim \varphi$
- $S, q \models \langle\langle A^b \rangle\rangle \square \varphi$ iff there exists a b -strategy F_A for any $\lambda \in \text{out}(q, F_A)$, $S, \lambda[i] \models \varphi$ for all $i \geq 0$
- $S, q \models \neg \langle\langle A^b \rangle\rangle \square \varphi$ iff there exists a co-strategy F_A^c for any $\lambda \in \text{out}(q, F_A^c, b)$, $S, \lambda[i] \models \varphi$ for all $i \geq 0$
- $S, q \models \langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a b -strategy F_A such that for all $\lambda \in \text{out}(q, F_A)$, there is a position $i \geq 0$ such that $S, \lambda[i] \models \psi$ and $S, \lambda[j] \models \psi$ for all $j \in \{0, \dots, i-1\}$
- $S, q \models \neg \langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a co-strategy F_A^c such that for all $\lambda \in \text{out}(q, F_A, b)$, either $S, \lambda[i] \models \psi$ for all $i \geq 0$ or if there is a position $i \geq 0$ such that $S, \lambda[i] \models \psi$ then there exists $0 \leq j < i$ such that $S, \lambda[j] \models \sim \varphi$

4 Axiomatisation

In this section, we present an axiomatisation system of RBATL^∞ . We first introduce notations appearing in the axiomatisation. In the following, A, A_1, A_2 denotes non-empty coalitions, b, b_1, b_2 and $d \in \mathbb{B}^\infty$. We say that $b +^\infty d = e$ for any b, d and $e \in \mathbb{B}^\infty$ iff for every $i = 1, \dots, |r|$,

$$\begin{aligned} b_i + d_i &= e_i \text{ if } e_i \neq \infty \\ b_i &= d_i = \infty \text{ if } e_i = \infty \end{aligned}$$

Intuitively, $+\infty$ is used to split the usage of resources over multiple stages. As ∞ in some bound component means no constraint is required on the corresponding resource, we can also ignore the limitation on this resource in each stage by assigning ∞ to the corresponding component in each stage. For example, consider the bound $(2, 3, \infty)$, it can be split into $(1, 2, \infty)$ and $(1, 1, \infty)$ so that the bound on the third resource is ignored. Using $+\infty$ rather than $+$ makes sure that the number of ways to split of resource bounds is finite.

Hence, we define the following macros:

$$\begin{aligned}
\langle\langle A^b \rangle\rangle \circ \square \varphi &= \bigvee_{b_1+\infty b_2=b} \langle\langle A^{b_1} \rangle\rangle \circ \langle\langle A^{b_2} \rangle\rangle \circ \square \varphi \\
\neg \langle\langle A^b \rangle\rangle \circ \square \varphi &= \bigwedge_{b_1+\infty b_2=b} \neg \langle\langle A^{b_1} \rangle\rangle \circ \langle\langle A^{b_2} \rangle\rangle \circ \square \varphi \\
\langle\langle A^b \rangle\rangle \circ \varphi \mathcal{U} \psi &= \bigvee_{b_1+\infty b_2=b} \langle\langle A^{b_1} \rangle\rangle \circ \langle\langle A^{b_2} \rangle\rangle \circ \varphi \mathcal{U} \psi \\
\neg \langle\langle A^b \rangle\rangle \circ \varphi \mathcal{U} \psi &= \bigwedge_{b_1+\infty b_2=b} \neg \langle\langle A^{b_1} \rangle\rangle \circ \langle\langle A^{b_2} \rangle\rangle \circ \varphi \mathcal{U} \psi \\
\langle\langle \emptyset^b \rangle\rangle \circ \square \varphi &= \bigwedge_{b_1+\infty b_2=b} \langle\langle \emptyset^{b_1} \rangle\rangle \circ \langle\langle \emptyset^{b_2} \rangle\rangle \circ \square \varphi \\
\neg \langle\langle \emptyset^b \rangle\rangle \circ \square \varphi &= \bigvee_{b_1+\infty b_2=b} \langle\langle \emptyset^{b_1} \rangle\rangle \circ \langle\langle \emptyset^{b_2} \rangle\rangle \circ \square \varphi
\end{aligned}$$

Similar to the reason of introducing $+\infty$, we also define a zero bound $\bar{0}_b$ with respect to a bound b in \mathbb{B}^∞ as follows, for all $i = 1, \dots, |r|$

$$(\bar{0}_b)_i = \begin{cases} 0 & \text{if } b_i \neq \infty \\ \infty & \text{otherwise} \end{cases}$$

This means we ignore any component which is ∞ .

The axiomatisation system of RBATL^∞ is as follows:

Axioms

(PL) Tautologies of Propositional Logic

$$(L) \neg \langle\langle A^b \rangle\rangle \circ \perp$$

$$(T) \langle\langle A^b \rangle\rangle \circ \top$$

$$(B) \langle\langle A^b \rangle\rangle \circ \varphi \rightarrow \langle\langle A^d \rangle\rangle \circ \varphi$$

where $b \leq d$

$$(S) \langle\langle A_1^{b_1} \rangle\rangle \circ \varphi \wedge \langle\langle A_2^{b_2} \rangle\rangle \circ \psi \rightarrow \langle\langle (A_1 \cup A_2)^{b_1+b_2} \rangle\rangle \circ (\varphi \wedge \psi)$$

where $A_1 \cap A_2 = \emptyset$

$$(S_\emptyset) \langle\langle \emptyset^{b_1} \rangle\rangle \circ \varphi \wedge \langle\langle \emptyset^{b_2} \rangle\rangle \circ \psi \rightarrow \langle\langle \emptyset^{b_1} \rangle\rangle \circ (\varphi \wedge \psi)$$

where $b_1 \leq b_2$

$$(S_N) \langle\langle N^{b_1} \rangle\rangle \circ \varphi \wedge \langle\langle \emptyset^{b_2} \rangle\rangle \circ \psi \rightarrow \langle\langle N^{b_1} \rangle\rangle \circ (\varphi \wedge \psi)$$

where $b_1 \leq b_2$

$$\mathbf{(FP}_\square) \quad \langle\langle A^b \rangle\rangle \square \varphi \leftrightarrow \varphi \wedge (\langle\langle A^b \rangle\rangle \bigcirc \square \varphi \vee \langle\langle A^{\bar{0}b} \rangle\rangle \bigcirc (\langle\langle A^b \rangle\rangle \square \varphi))$$

$$\mathbf{(FP}_\mathcal{U}) \quad \langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi \leftrightarrow \psi \vee (\varphi \wedge (\langle\langle A^b \rangle\rangle \bigcirc \varphi \mathcal{U} \psi \vee \langle\langle A^{\bar{0}b} \rangle\rangle \bigcirc (\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi)))$$

$$\mathbf{(N}_\bigcirc) \quad \langle\langle \emptyset^b \rangle\rangle \bigcirc \varphi \leftrightarrow \neg \langle\langle N^b \rangle\rangle \bigcirc (\neg \varphi)$$

$$\mathbf{(N}_\square) \quad \langle\langle \emptyset^b \rangle\rangle \square \varphi \leftrightarrow \varphi \wedge \neg \langle\langle N^b \rangle\rangle \top \mathcal{U} \neg \varphi$$

$$\mathbf{(N}_\mathcal{U}) \quad \langle\langle \emptyset^b \rangle\rangle \varphi \mathcal{U} \psi \leftrightarrow \neg (\langle\langle N^b \rangle\rangle \neg \psi \mathcal{U} \neg (\varphi \vee \psi) \vee \langle\langle N^b \rangle\rangle \square \neg \psi)$$

Inference rules

$$\mathbf{(MP)} \quad \frac{\varphi, \varphi \rightarrow \psi}{\psi}$$

$$\mathbf{(\langle\langle A^b \rangle\rangle \bigcirc\text{-Monotonicity})} \quad \frac{\varphi \rightarrow \psi}{\langle\langle A^b \rangle\rangle \bigcirc \varphi \rightarrow \langle\langle A^b \rangle\rangle \bigcirc \psi}$$

$$\mathbf{(\langle\langle \emptyset^b \rangle\rangle \square\text{-Necessitation})} \quad \frac{\varphi}{\langle\langle \emptyset^b \rangle\rangle \square \varphi}$$

$$\mathbf{(\langle\langle A^b \rangle\rangle \square\text{-Induction})} \quad \frac{\theta \rightarrow (\varphi \wedge (\langle\langle A^b \rangle\rangle \bigcirc \square \varphi \vee \langle\langle A^{\bar{0}b} \rangle\rangle \bigcirc \theta))}{\theta \rightarrow \langle\langle A^b \rangle\rangle \square \varphi}$$

$$\mathbf{(\langle\langle A^b \rangle\rangle \mathcal{U}\text{-Induction})} \quad \frac{(\psi \vee (\varphi \wedge (\langle\langle A^b \rangle\rangle \bigcirc \varphi \mathcal{U} \psi) \vee \langle\langle A^{\bar{0}b} \rangle\rangle \bigcirc \theta)) \rightarrow \theta}{\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi \rightarrow \theta}$$

Proposition 1. *The above axiomatization system for RBATL^∞ is sound and complete.*

In the rest of this section, we provide a sketch proof of Proposition 1. As the proof of soundness is straightforward, it is ignored here. To prove the completeness, as usual, we will construct a model for any consistent formula φ_0 of RBATL^∞ . The constructed models are in the form of fixed-branch trees defined as follows.

Given a finite alphabet Θ , we denote the sets of finite words and infinite words of Θ by Θ^* and Θ^ω , respectively.

Definition 1. *A tree T is a subset of \mathbb{N}^* where for any $x \cdot c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$:*

- $x \in T$
- $x \cdot c' \in T$ for all $0 \leq c' \leq c$

Given a tree T , ϵ is the root of T . Nodes of T are elements of T . We define $\text{succ} : T \rightarrow 2^T$ as a function to return the successors of a node $x \in T$. Formally, $\text{succ}(x) = \{x \cdot c \in T \mid c \in \mathbb{N}\}$. The degree $d(x)$ of a node x is defined as the cardinality of $\text{succ}(x)$, i.e. $d(x) = |\text{succ}(x)|$. A node x is a leaf iff $d(x) = 0$. A node x is an interior node iff $d(x) > 0$.

Definition 2. Given a set Θ , a Θ -labeled tree is a pair (T, V) where T is a tree and $V : T \rightarrow \Theta$ is a mapping which labels each node of T with an element of Θ .

Given a finite set of agents $N = \{1, \dots, n\}$, for the purpose of constructing models for consistent formulas of RB-ATL, we are interested in a special form of Θ -labeled trees (T, V) where Θ is the set 2^Π of subsets of propositions and the degree of every node of T is fixed by some given number $k \in \mathbb{N}$, i.e. $\text{deg}(x) = k^n$ for all $x \in T$. Then, a 2^Π -labeled tree (T, V) with a fixed degree k^n can be considered as the skeleton of a model for RB-ATL formulas. We call a tree with a fixed degree k^n as a k^n -tree. Informally, each node of T is considered as a state. From each state $x \in T$, there are k^n transitions to its successors, namely from $x \cdot 0$ to $x \cdot k^n - 1$. We can name each transition from x to $x \cdot c$ by a tuple (a_1, \dots, a_n) where

1. $1 \leq a_i \leq k$
2. $\text{encode}((a_1, \dots, a_n)) = c$

Where $\text{encode} : \{1, \dots, k\}^n \rightarrow \{0, \dots, k^n - 1\}$ is a bijective function which is defined as

$$\text{encode}((x_1, \dots, x_n)) = (x_1 - 1)k^{n-1} + (x_2 - 1)k^{n-2} + \dots + (x_n - 1)$$

For convenience, we call the inverse function of encode as decode . Then, each transition from x to $x \cdot c$ can be considered as the effect of the joint action of n agents in N where agent i performs the action a_i for all $i \in \{1, \dots, n\}$ and $(a_1, \dots, a_n) = \text{decode}(c)$. Moreover, to become a model for RB-ATL formulas, we need to supply for each 2^Π -labeled k^n -tree (T, V) a costing function which defines the cost of each action of an agent at a node on the tree. We have the following definition.

Definition 3. A 2^Π -labeled k^n -costed-tree is a tuple (T, V, C) where (T, V) is a 2^Π -labeled k^n -tree and $C : T \times N \times \{1, \dots, k\} \rightarrow \mathbb{B}$ is a costing function.

Given a 2^Π -labeled k^n -costed-tree (T, V, C) , we define the corresponding RB-CGS $S_{(T, V, C)} = (n, T, \Pi, V, d, C, \delta)$ where $d(x, i) = k$ for all $x \in T$ and $i \in N$ and $\delta(x, (a_1, \dots, a_n)) = x \cdot \text{encode}((a_1, \dots, a_n))$. It is straightforward that $S_{(T, V, C)}$ is

well-defined. We shall write $(T, V, C), x \models \varphi$ for $S_{(T, V, C)}, x \models \varphi$ and $(T, V, C) \models \varphi$ for $(T, V, C), \epsilon \models \varphi$. Furthermore, we also have that in $S_{(T, V, C)}$, the available joint actions for any coalition A at any state are the same, i.e. $D_A(x) = D_A(x')$ for any $x, x' \in T$, hence we shall write Δ_A for $D_A(x)$.

Notice that when constructing the tree model for a consistent formula, we build k^n -costed-trees which are labeled by subsets of formulas rather than only a subset of propositional variables. However, we can consider them as models for RB-ATL formulas by restricting the labeling function V over the set of propositions, i.e. $V(t) \cap \Pi$. Finally, we define a simple tree as a tree which consists of only a root and its children.

The ingredients for labeling nodes of tree during the construction are defined in terms of closure of φ_0 .

Definition 4. *The closure $cl(\varphi_0)$ is the smallest set of formulas that satisfies the following closure condition:*

- All sub-formulas of φ including itself are in $cl(\varphi_0)$
- If $\langle\langle A^b \rangle\rangle \Box \varphi$ is in $cl(\varphi_0)$, then so are $\langle\langle A^{b_1} \rangle\rangle \circ \langle\langle A^{b_2} \rangle\rangle \Box \varphi$ for all $b_1 +_\infty b_2 = b$ and also $\langle\langle A^{0_b} \rangle\rangle \circ \langle\langle A^b \rangle\rangle \Box \varphi$
- If $\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$ is in $cl(\varphi_0)$, then so are $\langle\langle A^{b_1} \rangle\rangle \circ \langle\langle A^{b_2} \rangle\rangle \varphi \mathcal{U} \psi$ for all $b_1 +_\infty b_2 = b$ and also $\langle\langle A^{0_b} \rangle\rangle \circ \langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$
- If φ is in $cl(\varphi_0)$, then so is $\sim \varphi$
- $cl(\varphi_0)$ is also closed under finite positively boolean operator (\vee and \wedge) up to tautology equivalence.

Hence, $cl(\varphi_0)$ is finite. We denote $cl(\varphi_0)_\circ$ to be the set of all formulas of form $\langle\langle A^b \rangle\rangle \circ \varphi$ or $\neg \langle\langle A^b \rangle\rangle \circ \varphi$ in $cl(\varphi_0)$.

Then, the following three lemmas describe each step of the construction of the tree model. We only provide the proof of the last lemma.

Lemma 1. *Let $\Phi = \{\langle\langle A_1^{b_1} \rangle\rangle \circ \varphi_1, \dots, \langle\langle A_k^{b_k} \rangle\rangle \circ \varphi_k, \neg \langle\langle A^b \rangle\rangle \circ \varphi\}$ be a consistent set of formulas where:*

- All A_i are both non-empty and pair-wise disjoint
- $\cup_i A_i \subseteq A$
- $\sum_i b_i \leq b$

We have $\Psi = \{\varphi_1, \dots, \varphi_k, \sim \varphi\}$ is also consistent.

Lemma 2. Let $\Phi = \{\langle\langle A_1^{b_1} \rangle\rangle \circ \varphi_1, \dots, \langle\langle A_k^{b_k} \rangle\rangle \circ \varphi_k, \langle\langle \emptyset^{e_1} \rangle\rangle \circ \chi_1, \dots, \langle\langle \emptyset^{e_m} \rangle\rangle \circ \chi_m\}$ be a consistent set of formulas where:

- All A_i are both non-empty and pair-wise disjoint
- $\sum_i b_i \leq e_j$ for all j

We have $\Psi = \{\varphi_1, \dots, \varphi_k, \chi_1, \dots, \chi_m\}$ is also consistent.

We now use the above lemma to construct a simple tree which is locally consistent for a consistent set of formulas.

Definition 5. A tree (T, V, C) is locally consistent if and only if for any interior node $t \in T$:

1. If $\langle\langle A^b \rangle\rangle \circ \varphi \in V(t)$, then there is a move σ_A such that $C(t, A, \sigma_A) \leq b$ and for any $c \in \text{out}(\sigma_A)$ we have $\varphi \in V(c)$
2. If $\neg\langle\langle A^b \rangle\rangle \circ \varphi \in V(t)$, then for any move σ_A with $C(t, A, \sigma_A) \leq b$, there exists $c \in \text{out}(\sigma_A)$ where $\sim\varphi \in V(c)$

Lemma 3. Let Φ be a finite consistent set of formulas, Φ_\circ the subset of Φ which contains all formulas of the forms $\langle\langle A^b \rangle\rangle \circ \varphi$ or their negations from Φ and k some number where $|\Phi_\circ| < k$, there is a simple k^n -costed-tree (T, V, C) which is locally consistent such that $V(\epsilon) = \Phi$.

Proof. Firstly, we have $\neg\langle\langle N^b \rangle\rangle \circ \varphi$ and $\neg\langle\langle \emptyset^b \rangle\rangle \circ \varphi$ are equivalent to $\langle\langle \emptyset^b \rangle\rangle \circ \sim\varphi$ and $\langle\langle N^b \rangle\rangle \circ \sim\varphi$, respectively. Therefore, we only consider the case when Φ_\circ does not contain formulas of the form $\neg\langle\langle N^b \rangle\rangle \circ \varphi$ and $\neg\langle\langle \emptyset^b \rangle\rangle \circ \varphi$.

Assume that

$$\begin{aligned} \Phi_\circ = & \{ \langle\langle A_1^{b_1} \rangle\rangle \circ \varphi_1, \dots, \langle\langle A_m^{b_m} \rangle\rangle \circ \varphi_m \} \cup \\ & \{ \neg\langle\langle B_1^{d_1} \rangle\rangle \circ \psi_1, \dots, \neg\langle\langle B_l^{d_l} \rangle\rangle \circ \psi_l \} \cup \\ & \{ \langle\langle \emptyset^{e_1} \rangle\rangle \circ \chi_1, \dots, \langle\langle \emptyset^{e_h} \rangle\rangle \circ \chi_h \} \end{aligned}$$

where all A_i are non-empty, all B_i are both non-empty and not equal to the grand coalition N . We define a vector $\max \in \mathbb{B}$ where each component of \max is the maximal bound except infinity of the corresponding resource appearing in Φ_\circ . In the case that there is no maximal bound, then the component of \max is set to 0. For example, assume that $|r| = 2$ and $\Phi_\circ = \{ \langle\langle \{1, 2\}^{(2,2)} \rangle\rangle \circ p, \langle\langle \{1\}^{(3,\infty)} \rangle\rangle \circ p \}$, then $\max = (3, 2)$; in another case, if $\Phi_\circ = \{ \langle\langle \{1\}^{(3,\infty)} \rangle\rangle \circ p \}$ then $\max = (3, 0)$.

Then, we define a function $\text{deinf} : \mathbb{B}^\infty \rightarrow \mathbb{B}$ which removes infinity from a bound as follows: $\text{deinf}(b) = b'$ where for all $i = 1, \dots, |r|$

$$(b')_i = \begin{cases} (b)_i & \text{if } (b)_i \neq \infty \\ \max_i + 1 & \text{otherwise} \end{cases}$$

Let e be a bound of resources such that $e > \text{deinf}(e_i)$ for all $i \in \{1, \dots, h\}$.

We construct a tree with a root labelled by Φ and k^n children, each is denoted by $c = \text{encode}(a_1, \dots, a_n)$ where $a_i \in \{1, \dots, k\}$. Intuitively, we allow each agent i to perform k different actions and the special action k for each agent will be considered as the costless idle-action. We shall denote $c(i) = a_i$ for the action performed by agent i with the corresponding outcome c . In the following, we define the labeling function $V(c)$ for each node c and the cost function $C(\epsilon, i, a)$ for each agent i and action $a \in \{1, \dots, k\}$.

For each $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$ wherein $A_p \neq \emptyset$, φ_p is added to $V(c)$ whenever $c(i) = p$ for all $i \in A_p$. Let \min_{A_p} be the smallest number in A_p , we assign the cost of action p performed by \min_{A_p} to be b_p , i.e. $C(\epsilon, \min_{A_p}, p) = \text{deinf}(b_p)$. For actions of other agents i in A_p , we assign $C(\epsilon, i, p) = \bar{0}$.

After considering all $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$, for all other unassigned-cost actions, i.e. actions $a > m$ but $a < k$ for all agents, we simply set their costs to be e . The action k performed by all agents is defined to associate with the cost 0. We denote $C(c) = \sum_{i \in N} C(\epsilon, i, c(i))$. Then, for each $\langle\langle \emptyset_p^{e_p} \rangle\rangle \circ \chi_p \in \Phi_\circ$, χ_p is added to $V(c)$ whenever $C(c) \leq e_p$.

Finally, we will add at most one formula from the negation formulas of Φ_\circ to $V(c)$. We denote $C(c, A) = \sum_{i \in A} C(\epsilon, i, c(i))$. For each c , let $\Phi_\circ^-(c) = \{\neg \langle\langle B^d \rangle\rangle \circ \psi \in \Phi_\circ \mid C(c, B) \leq d\} = \{\neg \langle\langle B_{i_1}^{d_{i_1}} \rangle\rangle \circ \psi_{i_1}, \dots, \neg \langle\langle B_{i_c}^{d_{i_c}} \rangle\rangle \circ \psi_{i_c}\}$ where $i_1 < i_2 < \dots < i_c$. Let $I = \{i \mid m < c(i) \leq m + l_c\}$ and $j = \sum_{i \in I} (c(i) - 1 - m) \bmod l_c + 1$. Consider $\neg \langle\langle B_{i_j}^{d_{i_j}} \rangle\rangle \circ \psi_{i_j}$: if $N \setminus B_{i_j} \subseteq I$, then $\sim \psi_{i_j}$ is added into $V(c)$.

We now need to show that our simple tree is locally consistent. In the first step, we show that all labels are consistent. It is obvious that $V(\epsilon) = \Phi$ is consistent.

Let us firstly consider every child c of the root where $\sim \psi_q \in V(c)$ from some negation formula in Φ_\circ . This will imply that there will be no $\chi \in V(c)$ from the formulas of the form $\langle\langle \emptyset^b \rangle\rangle \circ \chi$ in Φ_\circ . The reason is that because some $\sim \psi_q \in V(c)$, there must be some agent performing an action $a \in \{m + 1, \dots, m + l_c\}$ as otherwise $I = \emptyset$ and the condition $N \setminus B_{i_j} \subseteq I$ fails since $B_{i_j} \neq N$. We know that the cost of this action is e , then $C(c) \geq e$, therefore, no χ will be added into $V(c)$.

When there is no $\varphi \in V(c)$ from the formulas of the form $\langle\langle A^b \rangle\rangle \circ \varphi$ in Φ_\circ , the proof is trivial as there is only one $\sim \psi_q \in V(c)$. If there are some $\varphi_p \in V(c)$ where $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$, then for each p , $c(i) = p < m$ for all $i \in A_p$. Hence, all A_p are pair-wise disjoint. This simply shows that the set of $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$ where $\varphi_p \in V(c)$ and $\sim \langle\langle B_q^{b_q} \rangle\rangle \circ \psi_q$ satisfies the conditions of Lemma 1. Therefore, $V(c)$ is consistent.

Now, we consider every child c of the root where there is no $\sim \psi \in V(c)$ from some negation formula in Φ_\circ .

When there is no $\varphi \in V(c)$ from the formulas of the form $\langle\langle A^b \rangle\rangle \circ \varphi$ in Φ_\circ , the proof is trivial as there are only some $\chi_q \in V(c)$. If there are some $\varphi_p \in V(c)$ where $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$ and $A_p \neq \emptyset$, then for each p , $c(i) = p < m$ for all $i \in A_p$. Hence, all A_p are pair-wise disjoint. For any $\chi_q \in V(c)$ by some $\langle\langle \emptyset^{e_q} \rangle\rangle \circ \chi_q \in \Phi_\circ$, we have that $e_q \geq C(c) \geq \sum_p b_p$. This simply shows that the set of $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$ where $\varphi_p \in V(c)$ and $\langle\langle \emptyset_q^{e_q} \rangle\rangle \circ \chi_q$ satisfies the conditions of Lemma 2. Therefore, $V(c)$ is consistent.

Let us now check the conditions of local consistency on the newly built tree.

For $\langle\langle A_p^{b_p} \rangle\rangle \circ \varphi_p \in \Phi_\circ$, it is straightforward that the move σ_{A_p} where all agents in A_p performs action $p < m$ has cost equal to b_p and for any $c \in \text{out}(\sigma_{A_p})$, $\varphi_p \in V(c)$.

For $\neg\langle\langle B_p^{d_p} \rangle\rangle \circ \psi_p \in \Phi_\circ$ and σ being an arbitrary move of agents in B_p of which cost is at most equal to d_p , we will point out an output $c \in \text{out}(\sigma)$ where $\sim\psi \in V(c)$ and the actions of agents out of B_p are within $m+1$ and $m+l$ which always cost e amount of resources. Even though we do not know the exact actions of agents out of B_p , the costs of those unspecified actions are known to be e . Hence, we can determine the set $\Phi_\circ^-(c) = \{\neg\langle\langle B_{i_1}^{d_{i_1}} \rangle\rangle \circ \psi_{i_1}, \dots, \neg\langle\langle B_{i_{l_c}}^{d_{i_{l_c}}} \rangle\rangle \circ \psi_{i_{l_c}}\}$ as well as l_c . It is obvious that $\neg\langle\langle B_p^{d_p} \rangle\rangle \circ \psi_p \in \Phi_\circ^-(c)$, then $p = i_r$ for some $1 \leq r \leq l_c$. Let σ_i be the action performed by agent i in B_p , we define $c(i) = \sigma_i$ for all $i \in B_p$. Let $I' = \{i \in B_q \mid m < c(i) \leq m + l_c\}$ and $j' = \sum_{i \in I'} (c(i) - 1 - m) \bmod l_c$. We select an arbitrary $i' \notin B_p$ and set $c(i') = m + (r - 1 - j') \bmod l_c + 1$. For all other $i \notin B_p$, let $c(i) = m + 1$. Then, we have $I = \{i \mid m < c(i) \leq m + l_c\} = (N \setminus B_p) \cup I'$. Therefore, $\sum_{i \in I} (c(i) - 1 - m) \bmod l_c + 1 = \sum_{i \in I' \cup \{i'\}} (c(i) - 1 - m) \bmod l_c + 1 = (j' + c(i') - 1 - m) \bmod l_c + 1 = (r - 1) \bmod l_c + 1 = r$, and $N \setminus B_p \subseteq I$ because $I = (N \setminus B_p) \cup I'$. Hence $\sim\psi_p \in V(c)$. \square

Let us consider an example of building such a locally consistent tree. Consider a system of 2 agents, i.e. $N = \{1, 2\}$, 1 resource, i.e. $|r| = 1$, and the following set Φ_\circ of RB-ATL formulas.

$$\Phi_\circ = \{\langle\langle 1^1 \rangle\rangle \circ p, \langle\langle 2^\infty \rangle\rangle \circ (p \rightarrow q), \neg\langle\langle 1^2 \rangle\rangle \circ q, \neg\langle\langle 2^2 \rangle\rangle \circ p, \langle\langle \emptyset^2 \rangle\rangle \circ (\neg q)\}$$

It is easy to see that $\max = 2$ and we can pick $e = 3$. We now construct a simple tree which is locally consistent and the root is labeled by Φ_\circ . As $|\Phi_\circ| = 5$, let us consider the number of actions for each agent $k = 6$. Then, the set of outcomes is $O = \{(i, j) \mid 1 \leq i, j \leq 6\}$.

Consider the formula $\langle\langle 1^1 \rangle\rangle \circ p \in \Phi_\circ$, we add to the label of every $V((1, j))$ the formula p , for any $1 \leq j \leq 6$. The cost of action 1 of agent 1 is 1.

Consider the formula $\langle\langle 2^\infty \rangle\rangle \circ (p \rightarrow q) \in \Phi_\circ$, we add to the label of every $V((i, 2))$ the formula $p \rightarrow q$, for any $1 \leq i \leq 6$. The cost of action 2 of agent 2 is $\max + 1 = 3$.

As we mean the action 6 for both agents to be the idle action, we simply assign the cost 0 for 6 of both agents. Then we assign the cost $e = 3$ for all cost-unassigned actions of both agents. After this step, we add $\neg q$ to every outcome (i, j) of which the total cost of i and j is no more than 2.

We have the assignment of labels $V((i, j))$ for every $1 \leq i, j \leq 6$ so far as in the following table where each column (row) corresponds to an action of agent 1 (2) together with its cost.

AC	1^1	2^3	3^3	4^3	5^3	6^0
1^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2^3	$\{p, p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$
3^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
4^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
5^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
6^0	$\{p, \neg q\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\neg q\}$

Let us consider the negation formulas in Φ_\circ . We take each outcome into account to decide whether one of the sub-formulas of the negation formulas in Φ_\circ is included in the label of the outcome.

We consider the outcome $c = (1^1, 1^3)$, then $\Phi_\circ(c) = \{\neg\langle\langle 1^2 \rangle\rangle \circ q\}$. Then $l_c = 1$, $I = \{i \mid 2 < c(i) \leq 2 + 1\} = \emptyset$. Therefore, as $N \setminus \{1\} \notin I$, $\neg q$ is not included in $V(c)$.

We consider the outcome $c = (1^1, 3^3)$, then $\Phi_\circ(c) = \{\neg\langle\langle 1^2 \rangle\rangle \circ q\}$. Then $l_c = 1$, $I = \{i \mid 2 < c(i) \leq 2 + 1\} = \{2\}$. Therefore, as $N \setminus \{1\} \subseteq I$, $\neg q$ is included in $V(c)$.

We consider the outcome $c = (3^3, 6^0)$, then $\Phi_\circ(c) = \{\neg\langle\langle 2^2 \rangle\rangle \circ p\}$. Then $l_c = 1$, $I = \{i \mid 2 < c(i) \leq 2 + 1\} = \{1\}$. Therefore, as $N \setminus \{2\} \subseteq I$, $\neg p$ is included in $V(c)$.

We can apply the similar argument, and obtain the following table.

AC	1^1	2^3	3^3	4^3	5^3	6^0
1^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
2^3	$\{p, p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$	$\{p \rightarrow q\}$
3^3	$\{p, \neg q\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\neg q\}$
4^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
5^3	$\{p\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
6^0	$\{p, \neg q\}$	$\{\}$	$\{\neg p\}$	$\{\}$	$\{\}$	$\{\neg q\}$

In the following, Γ is the finite set of all maximal consistent sets of formulas from $cl(\varphi_0)$. As $cl(\varphi_0)$ is finite, Γ is also finite. We extend the construction for satisfying eventuality formulas which are in forms of $\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi$ and $\langle\langle A^b \rangle\rangle \square \varphi$ by the following lemma. We also omit the proof.

Firstly, we say that a formula $\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi (\neg \langle\langle A^b \rangle\rangle \square \varphi)$ is realised from a node t of a Γ -labeled tree (T, V, C) if there exists a strategy (co-strategy) F_A such that for all $\lambda \in \text{out}(t, F_A, b)$ ($\lambda \in \text{out}(t, F_A^c, b)$), there is some i such that $\psi \in V(\lambda[i])$ and $\varphi \in V(\lambda[j])$ for all $j \in \{0, i-1\}$ ($\sim \varphi \in V(\lambda[i])$).

Lemma 4. *For each formula $\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi (\neg \langle\langle A^b \rangle\rangle \square \varphi)$ and $x \in \Gamma$, there is finite Γ -labeled k^n -costed-tree (T, V, C) where:*

- $k = |\text{cl}(\varphi_0)_\circ| + 1$
- (T, V, C) is locally consistent
- $V(\epsilon) = x$
- If $\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi \in x$ ($\neg \langle\langle A^b \rangle\rangle \square \varphi \in x$) then (T, V, C) realises $\langle\langle A^b \rangle\rangle \varphi \mathcal{U} \psi (\neg \langle\langle A^b \rangle\rangle \square \varphi)$ from ϵ

Then, the final construction is as follows. For each consistent set x in Γ and an eventual formula φ of $\text{cl}(\varphi_0)$, we have a finite tree $(T_{x,\varphi}, V_{x,\varphi}, C_{x,\varphi})$ which realises φ with the root having label x . Let the eventual formulas in $\text{cl}(\varphi_0)$ be listed as $\varphi_0^e, \dots, \varphi_m^e$. In the following, we have the definition of the final tree.

Definition 6. *The final tree $(T_{\varphi_0}, V_{\varphi_0}, C_{\varphi_0})$ is constructed inductively as follows.*

- Initially, select an arbitrary $x \in \Gamma$ such that $\varphi_0 \in x$. As that formula is consistent, x must exist. Let $(T_{x,\varphi_0^e}, V_{x,\varphi_0^e}, C_{x,\varphi_0^e})$ be the initial tree.
- Given the tree constructed so far and the last used eventual formula φ_i^e . Then, for every leaf labelled by $y \in \Gamma$ of the currently constructed tree, we replace it with the tree $(T_{y,\varphi_j^e}, V_{y,\varphi_j^e}, C_{y,\varphi_j^e})$ where $j = i + 1$ if $i < m$ or $j = 0$ if otherwise.

Let S_{φ_0} be the model which is based on $(T_{\varphi_0}, V_{\varphi_0}, C_{\varphi_0})$. We have the following truth lemma which completes the proof of completeness of the axiomatization system for RBATL^∞ . The proof of this lemma is omitted in this paper.

Lemma 5. *For every node t of $(T_{\varphi_0}, V_{\varphi_0}, C_{\varphi_0})$ and every formula $\varphi \in \text{cl}(\varphi_0)$, if $\varphi \in V_{\varphi_0}(t)$ then $S_{\varphi_0}, t \models \varphi$.*

5 Related work and Conclusion

The extended logic RBATL^∞ is an useful tool which allows us to flexibly express properties in multiagent systems where available amount of resources may affect

the ability of the agents. Rather than specifying upper-bounds on every resource in a property, we can set up limitation over some resources which are of interest. A sound and complete axiomatization of RBATL^∞ is also presented in this paper.

Recently, there have been several efforts on logics for expressing resource-bounded properties for multi-agent systems. In [4], the authors extend the temporal logic CTL (and CTL*) where agents not only consume but also produce resources. However, such logics only express properties of single agent system. In [1], the authors extend ATL in order to express properties of coalitional abilities of bounded-memory multi-agent systems. The limitation of memory in the logic is set up by restricting the size of strategy mappings and the length of history in each strategy mapping. Both extensions have no axiomatization result.

In the future, we will consider the satisfiability and model-checking problem of RBATL^∞ .

Acknowledgments

I gratefully acknowledge Natasha Alechina for supporting and giving me valuable comments and suggestions about the proofs. I also thank anonymous referees for careful reading, corrections and suggestions on the text.

References

- [1] Thomas Ågotnes and Dirk Walther. A logic of strategic ability under bounded memory. *Journal of Logic, Language and Information*, 18(1):55–77, 2009.
- [2] Natasha Alechina, Brian Logan, Nguyen Hoang Nga, and Abdur Rakib. Resource-bounded alternating-time temporal logic. In Wiebe van der Hoek, Gal Kaminka, Yves Lesperance, Michael Luck, and Sandip Sen, editors, *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, Toronto, Canada, May 2010. IFAAMAS, IFAAMAS. (to appear).
- [3] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM (JACM)*, 49(5):672–713, 2002.
- [4] Nils Bulling and Berndt Farwer. RTL and RTL*: Expressing abilities of resource-bounded agents. In J. Dix, M. Fisher, and P. Novak, editors, *Proceedings 10th International Workshop Computational Logic in Multi-Agent Systems*, pages 2–19, 2009.

Abduction for (non-omniscient) agents

Fernando Soler-Toscano and Fernando R. Velázquez-Quesada

Abstract

Among the non-monotonic reasoning processes, abduction is one of the most important. Usually described as the process of looking for explanations, it has been recognized as one of the most commonly used in our daily activities. Still, the traditional definitions of an abductive problem and an abductive solution mention only theories and formulas, leaving agency out of the picture.

Our work proposes a study of abductive reasoning from an epistemic and dynamic perspective, making special emphasis on non-ideal agents. We begin by exploring what an abductive problem is in terms of an agent's information, and what an abductive solution is in terms of the actions that modify it. Then we explore the different kinds of abductive problems and abductive solutions that arise when we consider agents whose information is not closed under logical consequence, and agents whose reasoning abilities are not complete.

1 Abductive reasoning

Beyond the obvious facts that he has at some time done manual labour, that he takes snuff, that he is a Freemason, that he has been in China, and that he has done a considerable amount of writing lately, I can [get] nothing else.

*Sherlock Holmes
The Red-Headed League*

Among the non-monotonic reasoning processes, abduction [1] is one of the most important. Usually described as the process of *looking for an explanation*, it has been recognized as one of the most commonly used in our daily activities. Observing that Mr. Wilson's right cuff is very shiny for five inches and the left one has a smooth patch near the elbow, Holmes assumes that he (Mr. Wilson) has done a considerable amount of writing lately. Given the symptoms A and B , a doctor suspects that the patient suffers from C . Karen knows that when it rains, the grass gets wet, and that the grass is wet right now; then, she suspects that it has rained.

But though traditional examples of abductive reasoning are given in terms of an agent's information and its changes, classical definitions of an abductive problem and its solutions are given in terms of theories and formulas, without mentioning the agent's information and how it is modified.

The present work proposes a study of abductive reasoning from an epistemic and dynamic perspective. After recalling the classical definitions of an abductive problem and an abductive solution (the rest of the current section), we explore what an abductive problem is in terms of the agent's information, and what an abductive solution is in terms of the actions that modify it (Section 2). Then we focus on non-ideal agents, analyzing not only the cases that arise when the agent's information is not closed under logical consequence (Section 3) but also those that arise when the agent's reasoning abilities are not complete (Section 4). We finish with a summary, proposing lines for further work (Section 5).

In this paper we will use the term *information* in the most general sense, with the notions of *knowledge* or *belief* being particular instances that impose further restrictions, like truth or consistency. Moreover, though we will use formulas in *Epistemic Logic* (EL; [9]) and *Dynamic Epistemic Logic* style (DEL; [5]), we will not commit ourselves to any particular semantic model. The main goal of this work is to explore the possibilities and concepts that emerge from a dynamic epistemic analysis of abductive reasoning.

1.1 The classical approach to abduction

Traditionally, it is said that there is an abductive problem when there is a formula χ that is not predicted by the current theory Φ . Recently, it has been observed that, even if the theory does not entail χ , it might entail its negation. Following [1], we can identify two basic abductive problems.

Definition 1.1 (Abductive problem). Let Φ and χ be a theory and a formula, respectively, in some language \mathcal{L} . Let \vdash be a consequence relation on \mathcal{L} .

The pair (Φ, χ) is a *novel abductive problem* when neither χ nor $\neg\chi$ are consequences of Φ , i.e., when

$$\Phi \not\vdash \chi \quad \text{and} \quad \Phi \not\vdash \neg\chi$$

The pair (Φ, χ) is an *anomalous abductive problem* when, though χ is not a consequence of Φ , $\neg\chi$ is, i.e., when

$$\Phi \not\vdash \chi \quad \text{and} \quad \Phi \vdash \neg\chi$$

Traditionally, a solution for an abductive problem (Φ, χ) is a formula ψ that, together with Φ , entails χ . This solves the problem because now the theory is strong enough to *explain* χ . The *anomalous* case requires an extra initial step, since adding directly such ψ will make the theory to entail both χ and $\neg\chi$. The agent should perform first a *theory revision* that stop $\neg\chi$ from being a consequence of Φ . Here are the formal definitions.

Definition 1.2 (Abductive solution).

- Given a *novel* abductive problem (Φ, χ) , the formula ψ is an *abductive solution* if

$$\Phi, \psi \vdash \chi$$

- Given an *anomalous* abductive problem (Φ, χ) , the formula ψ is an *abductive solution* if it is possible to perform a theory revision to get a *novel* problem (Φ', χ) for which ψ is a solution.

In some cases, Definition 1.2 is too weak since it allows trivial solutions, like χ itself. Again, following [1], it is possible to make a further classification.

Definition 1.3 (Classification of abductive solutions). Let (Φ, χ) be an abductive problem. An abductive solution ψ is

<i>consistent</i> if	$\Phi, \psi \not\vdash \perp$
<i>explanatory</i> if	$\psi \not\vdash \chi$
<i>minimal</i> if, for every other abductive solution φ ,	$\psi \vdash \varphi$ implies $\varphi \vdash \psi$

The consistency requirement discards those ψ inconsistent with Φ and the explanatory requirement discards χ itself. Minimality works as the Occam's razor, asking for the solution ψ to be logically equivalent to any other solution it implies.

2 From an agent's perspective

Most of the examples of abductive reasoning involve an agent and its information. It is Holmes who observes that Mr. Wilson's right cuff is very shiny; it is a doctor who observes the symptoms A and B ; it is Karen who observes that the grass is wet. So when does an agent has an abductive problem (Φ, χ) ? By interpreting Φ as the agent's information, we get the following definitions. We use formulas in *EL* style, where $\text{Inf } \varphi$ is read as " φ is part of the agent's information".

Definition 2.1 (Subjective abductive problem). Let χ be a formula.

We say that an agent has a *novel* χ -*abductive problem* when neither χ nor $\neg\chi$ are part of her information, i.e., when the following formula holds:

$$\neg\text{Inf } \chi \wedge \neg\text{Inf } \neg\chi \quad (1)$$

We say that an agent has an *anomalous* χ -*abductive problem* when χ is not part of her information but $\neg\chi$ is, i.e., when the following formula holds:

$$\neg\text{Inf } \chi \wedge \text{Inf } \neg\chi \quad (2)$$

So an agent has a χ -abductive problem when χ is not part of her information. What about an abductive solution? Definition 1.2 states that ψ is a solution to a novel problem if, when added to the theory Φ , we get a theory that entails χ . But a theory is actually closed under logical consequence, so ψ is a solution if, when added to the theory, makes χ part of the theory too. The *anomalous* case needs another step, since a revision is required first.

We have identified Φ with the agent's information. Then, a solution for the subjective *novel* case is a formula ψ that, when added to the agent's information, makes the agent informed about χ . This highlights the fact the requisites of a solution involve an *action*; an action that changes the agent's information by adding ψ to it. In the subjective *anomalous* case, the action was already clear, since the theory should be modified. But now we can see that the requisites for this case involves *two* actions: removing a piece of information and then incorporating a new one.

We will express changes in the agent's information by using formulas in *DEL* style. In particular, formulas of the form $\langle \text{Add}_\phi \rangle \varphi$ will be read as " ϕ can be added to the agent's information and, after that, φ is the case", and formulas of the form $\langle \text{Rem}_\phi \rangle \varphi$ will be read as " ϕ can be removed from the agent's information and, after that, φ is the case".

Definition 2.2 (Subjective abductive solution). Suppose an agent has a *novel* χ -abductive problem, that is, $\neg \text{Inf } \chi \wedge \neg \text{Inf } \neg \chi$ holds. A formula ψ is an *abductive solution* to this problem if, when added to the agent's information, the agent becomes informed about χ . In a formula,

$$\langle \text{Add}_\psi \rangle \text{Inf } \chi$$

Now suppose the agent has an *anomalous* χ -abductive problem, that is, $\neg \text{Inf } \chi \wedge \text{Inf } \neg \chi$ holds. A formula ψ is an *abductive solution* to this problem if the agent can revise her information to remove $\neg \chi$ from it and, after it, the incorporation of ψ makes χ part of her information. In a formula,

$$\langle \text{Rem}_{\neg \chi} \rangle (\neg \text{Inf } \neg \chi \wedge \langle \text{Add}_\psi \rangle \text{Inf } \chi)$$

What about the further classification for abductive solutions? We can also provide formulas that characterize them.

Definition 2.3 (Classification of subjective abductive solutions). Suppose an agent has a χ -abductive problem. A formula ψ is a(n)

- *consistent* abductive solution if it is a solution and can be added to the agent's information without making the latter inconsistent:

$$\langle \text{Add}_\psi \rangle (\text{Inf } \chi \wedge \neg \text{Inf } \perp)$$

- *explanatory* abductive solution if it is a solution and it does not imply χ , that is, it only *complements* the agent's information to produce χ :

$$\neg(\psi \rightarrow \chi) \wedge \langle \text{Add}_\psi \rangle \text{Inf } \chi$$

- *minimal* abductive solution if it is a solution and, for any φ , if φ is a solution that becomes part of the agent's information after ψ is added, then ψ also becomes part of the agent's information after φ is added.

$$\langle \text{Add}_\psi \rangle \text{Inf } \chi \wedge (\langle \text{Add}_\varphi \rangle \text{Inf } \chi \wedge \langle \text{Add}_\psi \rangle \text{Inf } \varphi) \rightarrow \langle \text{Add}_\varphi \rangle \text{Inf } \psi$$

3 A non-omniscient agent

In the classical definition of an abductive problem, the set of formulas Φ is understood as a *theory*, usually assumed to be closed under logical consequence, as we mentioned before. If this is the case, then we have actually revised an omniscient case.

But our agent does not need to be ideal. And if the agent's information is not closed under logical consequence, then we should make a difference between the information she actually has, her *explicit* information, and what follows logically from it, her *implicit information* [12; 11; 15].

3.1 Abductive problems

A non-omniscient agent has an abductive problem whenever χ is not part of her *explicit* information. As a consequence, the modality Inf in Definition 2.1 becomes Inf_{Ex} . But now each of our two abductive problems splits into four, according to the agent's *implicit* information (Inf_{Im}) about χ and $\neg\chi$. These eight cases include inconsistent situations in which the agent is implicitly informed about both χ and $\neg\chi$. They can be discarded under particular interpretations of the agent's information, like *knowledge* or *consistent beliefs*, but we have chosen to keep them here for the sake of generality.

Still, not all these cases are possible. We have said that *implicit* information is what follows logically from the *explicit* one, so explicit information itself should be implicit information, that is,

$$\text{Inf}_{\text{Ex}} \varphi \rightarrow \text{Inf}_{\text{Im}} \varphi$$

By assuming this formula, we can drop the cases in which some formula is in the agent's explicit information, but not in her implicit one.

Definition 3.1 (Non-omniscient abductive problems). A non-omniscient agent can face six different abductive problems, each one of them characterized by a formula in Table 1.

$\neg\text{Inf}_{\text{Ex}}\chi \wedge \neg\text{Inf}_{\text{Ex}}\neg\chi \wedge$	{	$\neg\text{Inf}_{\text{Im}}\chi \wedge \neg\text{Inf}_{\text{Im}}\neg\chi$	(1.1)
		$\text{Inf}_{\text{Im}}\chi \wedge \neg\text{Inf}_{\text{Im}}\neg\chi$	(1.2)
		$\neg\text{Inf}_{\text{Im}}\chi \wedge \text{Inf}_{\text{Im}}\neg\chi$	(1.3)
		$\text{Inf}_{\text{Im}}\chi \wedge \text{Inf}_{\text{Im}}\neg\chi$	(1.4)
$\neg\text{Inf}_{\text{Ex}}\chi \wedge \text{Inf}_{\text{Ex}}\neg\chi \wedge$	{	$\neg\text{Inf}_{\text{Im}}\chi \wedge \text{Inf}_{\text{Im}}\neg\chi$	(2.3)
		$\text{Inf}_{\text{Im}}\chi \wedge \text{Inf}_{\text{Im}}\neg\chi$	(2.4)

Table 1: Abductive problems for non-omniscient agents.

Let us review each one of the *novel* cases. In case **(1.1)**, the *truly novel* one, the agent lacks explicit and implicit information about both χ and $\neg\chi$; the formula χ is a real novelty for her. But in case **(1.2)**, the *not implicit novelty* one, though the agent does not have explicit information about neither χ nor $\neg\chi$, she has implicit information about χ . In other words, χ is a novelty for the agent's explicit information, but not for her implicit one since χ follows logically from what she explicitly has. In case **(1.3)**, the *implicitly anomaly* one, the agent lacks explicit information about both χ and $\neg\chi$, but implicitly she is informed about $\neg\chi$. Finally we have the *implicitly inconsistent* case, **(1.4)**, in which the agent lacks explicit information about both χ and $\neg\chi$, but has an implicit inconsistency.

Now for the *anomalous* cases. Case **(2.3)** is the *truly anomalous* one: the agent has both explicit and implicit information about $\neg\chi$, and lacks both explicit and implicit information about χ . In the remaining one **(2.4)**, called *anomaly with implicit inconsistency*, though the agent has explicit information about $\neg\chi$ but not about χ , the latter follows from her explicit information.

Omniscience as a particular case An agent is omniscient when she has explicitly all her implicit information. With this extra requirement, expressed by the formula $\text{Inf}_{\text{Im}}\varphi \rightarrow \text{Inf}_{\text{Ex}}\varphi$, cases **(1.2)**, **(1.3)**, **(1.4)** and **(2.4)** can be discarded since explicit and implicit information do not coincide. This leaves us only with cases **(1.1)** and **(2.3)**; exactly the two cases of Definition 2.1.

3.2 Abductive solutions

We have defined non-omniscient χ -abductive problems as situations in which the agent is not *explicitly* informed about χ . Accordingly, for defining a solution, we will look for an action (or a sequence of them) that makes the agent *explicitly* informed about χ , without having neither implicit nor explicit information about $\neg\chi$. We will focus on cases **(1.1)**, **(1.2)**, **(1.3)** and **(2.3)**, leaving the inconsistent ones, **(1.4)** and **(2.4)**, for future work.

Consider the *truly novel* case **(1.1)**: the agent lacks explicit and implicit information about both χ and $\neg\chi$. Then, just like in the omniscient case, a solution is a formula ψ that when added to the agent's *explicit* information makes the agent explicitly informed about χ .

Now consider the *not implicit novelty* case **(1.2)**: though the agent does not have χ explicitly, she has it *implicitly*. A solution for case **(1.1)**, adding some ψ , would also work here, but the agent does not really need this external interaction, since a non-omniscient agent has another possibility: she can make the implicit χ explicit by performing the adequate *reasoning steps*. And this gives us new possibilities not only this case. For example, in **(1.1)**, the agent does not need a ψ that makes χ explicit after added: a ψ that makes χ *implicit* is also a solution, since now she can make χ explicit by only reasoning. In fact, there are several strategies for solving each one of the abductive problems, but for simplicity we will focus on the most representative one for each one of them.

In case **(1.3)**, reasoning will only make the anomaly explicit. But then the agent will be in the *truly anomaly* case **(2.3)**, which can be solved by revising the agent's information to remove $\neg\chi$ from the explicit and implicit part, and then adding a ψ that makes χ part of her explicit information.

In the following definition, formulas of the form $\langle\alpha\rangle\varphi$ indicates that the agent can perform some reasoning step α after which φ is the case.

Definition 3.2 (Non-omniscient abductive solutions). Solutions for consistent non-omniscient abductive problems are provided in Table 2.

Case	Solution
(1.1)	A formula ψ such that $\langle\text{Add}_\psi\rangle\text{Inf}_{\text{Ex}}\chi$
(1.2)	A reasoning α such that $\langle\alpha\rangle\text{Inf}_{\text{Ex}}\chi$
(1.3)	A reasoning α and a formula ψ such that $\langle\alpha\rangle(\text{Inf}_{\text{Ex}}\neg\chi \wedge \langle\text{Rem}_{\neg\chi}\rangle(\neg\text{Inf}_{\text{Im}}\neg\chi \wedge \langle\text{Add}_\psi\rangle\text{Inf}_{\text{Ex}}\chi))$
(2.3)	A formula ψ such that $\langle\text{Rem}_{\neg\chi}\rangle(\neg\text{Inf}_{\text{Im}}\neg\chi \wedge \langle\text{Add}_\psi\rangle\text{Inf}_{\text{Ex}}\chi)$

Table 2: Solutions for consistent non-omniscient abductive problems.

Note how actions take us from some abductive problem to another. In case **(1.3)**, the proper reasoning will take the agent to case **(2.3)** from which, by applying the proper revision, the agent will reach case **(1.1)**, where a new piece of information is needed. The flowchart of Figure 1 shows this.

Classification of abductive solutions The extra requisites of Definition 2.3 can be adapted in this non-omniscient case. For the consistency and the explanatory requirements there are no important changes: we just require for the agent's *implicit* (and therefore her explicit information too) to be consistent at the end of the sequence of actions ($\neg\text{Inf}_{\text{Im}}\perp$), and for the formula ψ to not imply χ ($\neg(\psi \rightarrow \chi)$) in the cases in which it is needed

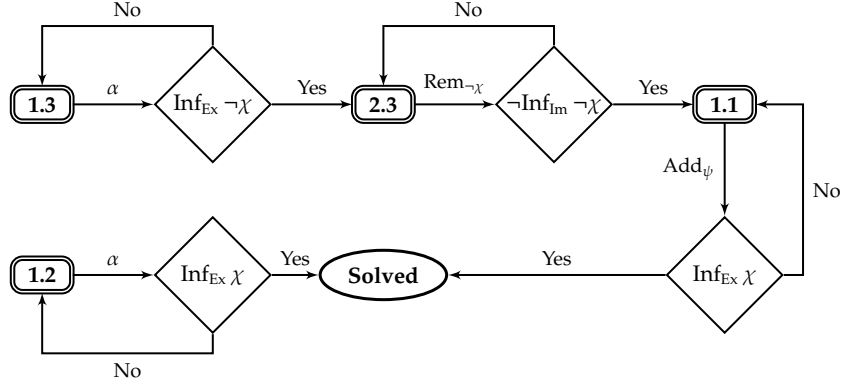


Figure 1: Flowchart of abductive solutions for non-omniscient agents.

((1.1), (1.3) and (2.3)). The minimality requirement now gives us more options. We can define it over the action $\langle \text{Add}_\psi \rangle$, looking for the weakest formula ψ , but it can also be defined over the action $\langle \text{Rem}_{\neg\chi} \rangle$, looking for the revision that removes the smallest amount of information. It can even be defined over the action $\langle \alpha \rangle$, looking for the shortest reasoning chain.

4 A non-dynamically-omniscient agent

Even though the agents of the previous section are non-omniscient, there is still an idealization about them. We have defined the agent's implicit information as what follows logically from her explicit information, but a more 'real' agent does not need to be *dynamically omniscient* in the sense that she does not need to have complete reasoning abilities. In other words, she may not be able to derive all logical consequences of her explicit information. This difference is important, because then a solution for a χ -abductive problem does not need to be as strong as a formula that, when added, also informs explicitly the agent about χ ; it can also be some formula that, when added, allow the agent to *derive* χ .

4.1 Abductive problems

Now we can make a further refinement. We can distinguish between what follows logically from the agent's explicit information, the *objective* implicit information Inf_{Im} , and what the agent can actually derive, the *subjective* implicit information Inf_{Der} . In other words, $\text{Inf}_{\text{Der}} \varphi$ holds when the agent can perform a sequence of reasoning steps that make φ explicit information. In particular, an empty sequence of reasoning steps makes explicit the information that is already explicit, so we assume

$$\text{Inf}_{\text{Ex}} \varphi \rightarrow \text{Inf}_{\text{Der}} \varphi$$

Though not *complete*, we can assume that the agent's reasoning abilities are *sound*. This makes subjective implicit information part of objective implicit one, giving us

$$\text{Inf}_{\text{Der}} \varphi \rightarrow \text{Inf}_{\text{Im}} \varphi$$

Each one of the six abductive problems of Table 1 turns into four cases, according to whether the agent can derive or not what follows logically from her explicit information, that is, according to whether $\text{Inf}_{\text{Der}} \chi$ and $\text{Inf}_{\text{Der}} \neg \chi$ hold or not. Our two assumptions allow us to discard some of the cases, leaving us with the following.

Definition 4.1 (Extended abductive problems). A non-omniscient agent without complete reasoning abilities can face eleven different abductive problems, each one of them characterized by a formula in Table 3.

$\neg \text{Inf}_{\text{Ex}} \chi \wedge \neg \text{Inf}_{\text{Ex}} \neg \chi \wedge \left\{ \neg \text{Inf}_{\text{Der}} \chi \wedge \neg \text{Inf}_{\text{Der}} \neg \chi \right\} \wedge \neg \text{Inf}_{\text{Im}} \chi \wedge \neg \text{Inf}_{\text{Im}} \neg \chi$	(1.1.a)
$\neg \text{Inf}_{\text{Ex}} \chi \wedge \neg \text{Inf}_{\text{Ex}} \neg \chi \wedge \left\{ \begin{array}{l} \neg \text{Inf}_{\text{Der}} \chi \wedge \neg \text{Inf}_{\text{Der}} \neg \chi \\ \text{Inf}_{\text{Der}} \chi \wedge \neg \text{Inf}_{\text{Der}} \neg \chi \end{array} \right\} \wedge \text{Inf}_{\text{Im}} \chi \wedge \neg \text{Inf}_{\text{Im}} \neg \chi$	(1.2.a) (1.2.b)
$\neg \text{Inf}_{\text{Ex}} \chi \wedge \neg \text{Inf}_{\text{Ex}} \neg \chi \wedge \left\{ \begin{array}{l} \neg \text{Inf}_{\text{Der}} \chi \wedge \neg \text{Inf}_{\text{Der}} \neg \chi \\ \neg \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg \chi \end{array} \right\} \wedge \neg \text{Inf}_{\text{Im}} \chi \wedge \text{Inf}_{\text{Im}} \neg \chi$	(1.3.a) (1.3.c)
$\neg \text{Inf}_{\text{Ex}} \chi \wedge \neg \text{Inf}_{\text{Ex}} \neg \chi \wedge \left\{ \begin{array}{l} \neg \text{Inf}_{\text{Der}} \chi \wedge \neg \text{Inf}_{\text{Der}} \neg \chi \\ \text{Inf}_{\text{Der}} \chi \wedge \neg \text{Inf}_{\text{Der}} \neg \chi \\ \neg \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg \chi \\ \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg \chi \end{array} \right\} \wedge \text{Inf}_{\text{Im}} \chi \wedge \text{Inf}_{\text{Im}} \neg \chi$	(1.4.a) (1.4.b) (1.4.c) (1.4.d)
$\neg \text{Inf}_{\text{Ex}} \chi \wedge \text{Inf}_{\text{Ex}} \neg \chi \wedge \left\{ \neg \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg \chi \right\} \wedge \neg \text{Inf}_{\text{Im}} \chi \wedge \text{Inf}_{\text{Im}} \neg \chi$	(2.3.c)
$\neg \text{Inf}_{\text{Ex}} \chi \wedge \text{Inf}_{\text{Ex}} \neg \chi \wedge \left\{ \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg \chi \right\} \wedge \text{Inf}_{\text{Im}} \chi \wedge \text{Inf}_{\text{Im}} \neg \chi$	(2.4.d)

Table 3: Abductive problems with subjective/objective implicit information.

4.2 Abductive solutions

Recall that different abductive problems in Table 1 can have the same solution. For example, though abductive problem **(1.2)**, the *non-implicit novelty* case, can be solved by means of reasoning steps (see Table 2), we mentioned that it can also be solved like case **(1.1)**. But the further refinement that we have just done really makes them different. In case **(1.2.b)**, χ is subjective implicit information, so the agent can derive it and solve the problem by only reasoning. Nevertheless, this is not possible in **(1.2.a)** since χ is objective but not subjective implicit information. The agent cannot derive χ ; she needs a formula that, when added to her explicit information, makes χ explicit (like in the truly novel case); or, more interesting, she can *extend her reasoning abilities* with a formula/rule that allows her to derive χ .

The same happens with other cases; consider those derived from (1.3). In (1.3.c) the anomaly will be detected so the agent can start with a revision of her information. But in (1.3.a) the anomaly cannot be derived, so we have an objective but not subjective anomaly. The agent cannot detect and, moreover, cannot derive the anomaly, so a better approach for a solution is to solve (1.3.a) as a novel abductive problem. In fact, we can say that (1.3.a) is an *objective anomaly* but a *subjective novelty*.

Just like actions of reasoning, revision and addition can take us from one abductive problem of Table 1 to another, they also allows us to move between the abductive problem of Table 3. Again, we will focus on the consistent cases, discarding (1.4.*) and (2.4.d).

Definition 4.2 (Extended abductive solutions). Solutions for consistent extended abductive problems are provided in Table 4. It should be read as a transition table that provides actions and conditions that should hold in order to move from one abductive problem to another. There are six operations/conditions which are described below, from left to right.

- Action $\langle \text{Add}_{\psi} \rangle$ consists in adding ψ to the agent's explicit information. The aim is to make the agent explicitly informed about χ .
- Action $\langle \text{Add}_{\psi/\alpha} \rangle$ extends the agent's explicit information by adding a formula ψ or some inference resource α (e.g., a rule) with the aim to provide the agent with enough information so she can derive χ .
- Action $\langle \alpha \rangle$ consists on the application of reasoning steps. The goal here is to make χ explicit.
- Action $\langle \text{Add}_{\psi/\alpha} \rangle$ is just as before. The goal is that after the action, the agent should be able to derive $\neg\chi$.
- Action $\langle \alpha \rangle$ is just as before, this time with the aim to make $\neg\chi$ explicit.
- Action $\langle \text{Rem}_{\neg\chi} \rangle$ removes $\neg\chi$ from the agent's explicit information, but the goal here is to remove it from her implicit information as well.

Case	$\langle \text{Add}_{\psi} \rangle \text{Inf}_{\text{Ex}} \chi$	$\langle \text{Add}_{\psi/\alpha} \rangle \text{Inf}_{\text{Der}} \chi$	$\langle \alpha \rangle \text{Inf}_{\text{Ex}} \chi$	$\langle \text{Add}_{\psi/\alpha} \rangle \text{Inf}_{\text{Der}} \neg\chi$	$\langle \alpha \rangle \text{Inf}_{\text{Ex}} \neg\chi$	$\langle \text{Rem}_{\neg\chi} \rangle \neg \text{Inf}_{\text{Im}} \neg\chi$
(1.1.a)	Solved	—	—	—	—	—
(1.2.a)	—	(1.2.b)	—	—	—	—
(1.2.b)	—	—	Solved	—	—	—
(1.3.a)	—	—	—	(1.3.c)	—	—
(1.3.c)	—	—	—	—	(2.3.c)	—
(2.3.c)	—	—	—	—	—	(1.1.a)

Table 4: Solutions for consistent extended abductive problems.

Table 4 establishes a natural path to solve a consistent extended abductive problem in. The longest path corresponds to case **(1.3.a)** in which the agent does not have explicit information about neither χ nor $\neg\chi$ and, though $\neg\chi$ follows logically from her explicit information, she cannot derive it. A sequence of actions to solve this problem is, first, to provide the agent with enough information so she can derive $\neg\chi$, turning this case into **(1.3.c)**. Then, after reasoning to derive $\neg\chi$ she will have an explicit anomaly, case **(2.3.c)**. From here she needs to revise her information to remove $\neg\chi$ from it and, once she has done this and reached case **(1.1.a)**, she needs to extend her information with some ψ that will make her be explicitly informed about χ .

4.3 Collapsing the cases

We have taken the perspective of an outsider. From a subjective point of view, the agent does not need to solve an anomaly that she cannot detect. What guides the process of solving an abductive problem is explicit information and what she can derive from it, that is, the subjective implicit one. In other words, unaccessible inconsistencies should not matter!

We can simplify the solution of abductive problem by observing that some problems in Table 3 are in fact indistinguishable for the agent. Without further external interaction, she can only access her explicit information and eventually what she can derive from it; the rest, the implicit information that is not derivable is not relevant. For example, abductive problems **(1.{1,2,3,4}.a)** are in fact the same from the agent's point of view, and she will try to solve them in the same way. By reducing Table 3 according to the agent's subjective information we get:

$\neg\text{Inf}_{\text{Ex}} \chi \wedge \neg\text{Inf}_{\text{Ex}} \neg\chi \wedge$	$\left\{ \begin{array}{ll} \neg\text{Inf}_{\text{Der}} \chi \wedge \neg\text{Inf}_{\text{Der}} \neg\chi & \mathbf{(1.\{1,2,3,4\}.a)} \\ \text{Inf}_{\text{Der}} \chi \wedge \neg\text{Inf}_{\text{Der}} \neg\chi & \mathbf{(1.\{2, 4\}.b)} \\ \neg\text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg\chi & \mathbf{(1.\{3, 4\}.c)} \\ \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg\chi & \mathbf{(1.4.d)} \end{array} \right.$
$\neg\text{Inf}_{\text{Ex}} \chi \wedge \text{Inf}_{\text{Ex}} \neg\chi \wedge$	$\left\{ \begin{array}{ll} \neg\text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg\chi & \mathbf{(2.3.c)} \\ \text{Inf}_{\text{Der}} \chi \wedge \text{Inf}_{\text{Der}} \neg\chi & \mathbf{(2.4.d)} \end{array} \right.$

Note how these classes correspond to abductive problems in Table 1 in which Inf_{Der} appears in place of Inf_{Im} . Then the abductive solutions in Table 3 can be considered the *objective solutions* for the eleven objectively different abductive problems. But if we consider only the information accessible to the agent then the *subjective paths* she follows to solve abductive problems are similar to the Figure 1.

5 Summary and future work

We have presented definitions of *novel* and *anomalous* abductive problems from a subjective perspective. We have focused not only on omniscient agents, but also on those whose information is not closed under logical consequence and those whose reasoning abilities are not complete. Moreover, we have also provided definitions for what an abductive solution is for each one of these problems, identifying actions that allow the agent to move from one problem to another and the conditions that such actions should verify in order to provide an abductive solution.

Our work is just an initial exploration on abductive reasoning for (non-omniscient) agents, and there are many aspects yet to be studied. To begin with, we can still make a further refinement in our notions of information, this time relative to the explicit one. Among our explicit information, there are things that are supported by the rest of our information. Consider, for example, the quadratic formula for solving quadratic equations: even if we do not know or forget the formula, we can derive it if we know the process of completing squares. But we also have information that is not supported by the rest; things that, if not observed, we would not have. Consider our initial example of Mr. Wilson having a very shiny right cuff: Holmes did not know him before, so there is no way he could have derived this information about his cuff without observing him. And in fact, the intuitive idea of abductive reasoning is closer to situations of this kind in which we try to find support (justification) for facts that, being observed, have become part of our explicit information.

But we can also look for a more concrete form of what we already have. First, we have talked about information, but we can study more specific notions, like *knowledge* and *beliefs*, by asking for more specific requirements, like truth or consistency. Moreover, the real definite step will be given by providing a semantic model in which we can represent not only the introduced notions of information (explicit, subjective implicit, objective implicit) in their knowledge and belief versions, but also provide a concrete definition for the discussed actions that modify them: adding external information, reasoning in order to extend the explicit one or removing part of it. Our current efforts are oriented to *dynamic epistemic* approaches, following not only ideas like public announcements [14; 6] and revision [3; 2], but also the finer grained notions of dropping [4] and inference [7].

And finally we can look at, possibly, the most important question in abductive reasoning. We know now what an abductive solution is but, how can we find them? In other words, given a particular semantic model representing an agent's information, can we provide a procedure that returns 'the set of abductive solutions'? And though an abductive solution can be a formula that, when assumed, provides the agent with explicit information about the observed χ , the most interesting solutions are those that allow

the agent to derive χ , linking tightly the search for solutions with the agent's reasoning tools. This emphasizes the need of a semantic model that allows us to represent an agent's inference dynamics.

Even more: an important topic in abductive reasoning is the selection of the best explanation (e.g., [13] and [10]). Beyond logical requisites to avoid triviality, the definition of a suitable criteria to model the agent's preference for solutions is still an open problem. By using Kripke frames, we can provide some criteria based in a *plausibility* measure among accessible worlds [3; 2]. And at last (but not least), once the agent has selected her 'best' explanation, what shall she do with it?

Acknowledgements We thank Johan van Benthem and Ángel Nepomuceno-Fernández for their valuable comments and suggestions.

References

- [1] A. Aliseda. *Abductive Reasoning. Logical Investigations into Discovery and Explanation*, volume 330 of *Synthese Library Series*. Springer, 2006.
- [2] A. Baltag and S. Smets. A qualitative theory of dynamic interactive belief revision. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory (LOFT7)*, volume 3 of *Texts in Logic and Games*, pages 13–60. AUP, 2008.
- [3] J. van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007.
- [4] J. van Benthem and F. R. Velázquez-Quesada. Inference, promotion, and the dynamics of awareness. Technical Report PP-2009-43, ILLC, Universiteit van Amsterdam, 2009.
- [5] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library Series*. Springer, 2007.
- [6] J. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, ILLC, Universiteit van Amsterdam, 1999. ILLC Dissertation Series DS-1999-01.
- [7] D. Grossi and F. R. Velázquez-Quesada. *Twelve Angry Men: A study on the fine-grain of announcements*. In X. He, J. F. Horty, and E. Pacuit, editors, *LORI*, volume 5834 of *LNCS*, pages 147–160. Springer, 2009.
- [8] J. Y. Halpern, editor. *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, March 1986*, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

- [9] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca, N.Y., 1962.
- [10] J. Hintikka. What is abduction? The fundamental problem of contemporary epistemology. *Transactions of the Charles S. Peirce Society*, 34(3):503–533, 1998.
- [11] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In Halpern [8], pages 325–340.
- [12] H. J. Levesque. A logic of implicit and explicit belief. In *Proc. of AAAI-84*, pages 198–202, Austin, TX, 1984.
- [13] P. Lipton. *Inference to the Best Explanation*. Routledge, New York, 1991.
- [14] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, Tennessee, USA, 1989. Oak Ridge National Laboratory, ORNL/DSRD-24.
- [15] M. Y. Vardi. On epistemic logic and logical omniscience. In Halpern [8], pages 293–305.

Dynamic Epistemic Logic for Implicit and Explicit Beliefs

Fernando R. Velázquez-Quesada

Abstract

The *dynamic* turn in Epistemic Logic is based on the idea that notions of information should be studied together with the actions that modify them. Dynamic epistemic logics have explored how knowledge and beliefs change as consequence of, among others, acts of observation and upgrade. Nevertheless, the omniscient nature of the represented agents has kept finer actions outside the picture, the most important being the action of *inference*.

Following proposals for representing non-omniscient agents, recent works have explored how *implicit* and *explicit knowledge* change as a consequence of acts of observation, inference, consideration and even forgetting. The present work proposes a further step towards a common framework for representing finer notions of information and their dynamics. We propose a combination of existing works in order to represent *implicit* and *explicit beliefs*. Then, after adapting definitions for the actions of *upgrade* and *retraction*, we discuss the action of *inference on beliefs*, analyzing its differences with respect to inference on knowledge and proposing a rich system for its representation.

1 Introduction

Epistemic Logic [21] and its possible worlds semantics is a powerful and compact framework for representing an agent's information. Their *dynamic* versions [14] have emerged to analyze not only information in its knowledge and belief versions, but also the actions that modify them. Nevertheless, agents represented in this framework are *logically omniscient*, that is, their information is closed under logical consequence. This property, useful in some applications, hides finer reasoning actions that are crucial in some others, the most important being that of *inference*.

Based on the awareness approach of [17], several works have explored dynamics of information for non-omniscient agents. In a *propositional dynamic logic (PDL)* style, some of them have explored how the act of inference modifies an agent's *explicit* knowledge [15; 22]. In a *dynamic epistemic* style, some others have explored how the acts of observation, inference, consideration and forgetting affect *implicit* and *explicit knowledge* [5; 18; 10; 13].

The present work follows the previous ones, now focussing on the notion of *beliefs*. We combine approaches of the existing literature, proposing a setting for representing the notions of *implicit* and *explicit belief* (Section 2). Then we look into the dynamics of these notions; first, by adapting existing proposals to define the actions of *explicit upgrade* (*explicit revision*) and *retraction* (Section 3), and second, by discussing the action of *inference on beliefs* and its differences with *inference on knowledge*, and by proposing a rich system for its representation (Section 4).

2 Modelling implicit and explicit beliefs

This section recalls a framework for implicit and explicit information and a framework for beliefs. By combining them, we will get our model for representing implicit/explicit beliefs. But before going into their details, we recall the framework on which all the others are based.

Epistemic Logic. The frameworks of this section are based on that of *Epistemic Logic* (*EL*; [21]). Given a set of atomic propositions \mathbf{P} , the *EL* language extends the propositional one with formulas of the form $\Box\varphi$: “the agent is informed about φ ”. Though there are several possibilities, the classical semantic model for *EL*-formulas are *Kripke models*, tuples $M = \langle W, R, V \rangle$ with W a non-empty set of *possible worlds*, $V : W \rightarrow \wp(\mathbf{P})$ an atomic valuation function indicating which atomic propositions are true at each world, and $R \subseteq (W \times W)$ an accessibility relation indicating which worlds the agent considers possible from each one of them.

Formulas are evaluated on *pointed models* (M, w) with M a Kripke model and $w \in W$ a given evaluation point. Boolean connectives are interpreted as usual; the key clause is the one for $\Box\varphi$, indicating that *the agent is informed about φ at w iff φ is true in all the worlds the agent considers possible from w* :

$$(M, w) \models \Box\varphi \quad \text{iff} \quad \text{for all } u \in W, Rwu \text{ implies } (M, u) \models \varphi$$

2.1 Implicit and explicit information

Non-omniscient agents. The formula $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$ is valid in Kripke models: the agent’s information is closed under logical consequence. This becomes obvious when we realize that each possible world stands for a maximally consistent set of formulas. So if both $\Box(\varphi \rightarrow \psi)$ and $\Box\varphi$ hold at world w , both $\varphi \rightarrow \psi$ and φ are true in all worlds R -reachable from w . But then ψ also holds in all such worlds, and therefore $\Box\psi$ holds at w . Usually the discussion revolves around whether this is a reasonable assumption for ‘real’ agents. Even computational agents may not have this property, since they may lack of resources (space and/or time) to derive all the logical consequences of their information [1].

One of the most influential solutions to this *omniscience problem* is *awareness logic* [17]. This approach follows the idea of making a difference between *implicit (potential) information*, what the agent can eventually get, and *explicit information*, what the agent actually has [23; 25; 24; 27]. The main observation is that, in order to have explicit information about some formula φ , besides having it as implicit information, the agent should be *aware* of φ .

Syntactically, awareness logic extends the *EL* language with formulas of the form $A\varphi$: “the agent is aware of φ ”. Semantically, it extends Kripke models with a function A that assigns a set of formulas to the agent in each possible world. The new formulas are evaluated in the following way:

$$(M, w) \models A\varphi \quad \text{iff} \quad \varphi \in A(w)$$

Implicit information about φ is defined as $\Box\varphi$, but explicit information is defined as $\Box\varphi \wedge A\varphi$. Although implicit information is still closed under logical consequence, explicit information is not. This follows from the fact that, different from the possible worlds, the A -sets *do not need* to have any closure property; in particular, $\{\varphi \rightarrow \psi, \varphi\} \subseteq A(w)$ does not imply $\psi \in A(w)$.

Agents with reasoning abilities. Still, though a ‘real’ agent’s information does not need to be closed under logical consequence, it does not need to be static. The more interesting approach for us is that in which the agent can extend her explicit information by the adequate actions. But, which are these actions and what does the agent needs in order to perform them?

In [15], the author proposes a framework in which the actions available to the agent are different rules (e.g., modus ponens, conjunction elimination), each one of them represented by a relation between worlds that should be faithful to the rule’s spirit (e.g., the modus ponens relation should connect worlds with an implication and its antecedent with worlds augmented with the consequent). This yields an agent that does not need to be omniscient, but still is able to perform inferences.

In [22] the author goes one step further: a rule cannot be used by an agent unless the rule itself is also part of her explicit information. For example, for two worlds to be connected by the modus ponens relation, the initial one should have not only an implication and its antecedent, but also the *modus ponens* rule itself.

The combination of the mentioned ideas have produced models for representing implicit and explicit *knowledge* ([5; 28; 13; 18; 10] among others). But the notion of *belief* is different, as we discuss in the next subsection.

2.2 Modelling beliefs

The KD45 approach. For modelling knowledge in *EL*, it is usually asked for the accessibility relation R to be at least *reflexive* (making $\Box\varphi \rightarrow \varphi$ valid:

if the agent knows φ , then φ is true), and often to be also *transitive* and *euclidean* (giving the agent full positive and negative introspection). Beliefs can be represented in a similar way, now asking for R to satisfy weaker properties, the crucial one following the idea that, though beliefs do not need to be true, we can expect them to be *consistent*. This is achieved by asking for the relation to be *serial*, making the D axiom $\neg\Box\perp$ valid. Full introspection is usually assumed, yielding the classical $KD45$ approach.

Belief as what is most plausible. But beliefs are different from knowledge. Intuitively, we do not believe something because it is true in all possible situations; we believe it because it is true in those we consider most likely to be the case [19; 26]. This idea has led the development of variants of Kripke models [12; 4; 3]. Here we recall the *plausibility models* of [3].

A *plausibility model* is a Kripke model in which the accessibility relation, denoted now by \leq , is interpreted as a *plausibility* relation ordering possible worlds. This relation is assumed to be a preorder (a reflexive and transitive relation). Moreover, since the idea is to define the agent's beliefs as what is true in the most plausible worlds from the evaluation point, \leq should satisfy an important extra property: for any possible world w , the set of worlds that are better than w among those comparable to it should have maximal worlds. In order to state this property formally, denote by V_w the set of worlds comparable to w (its comparability class: $V_w := \{u \mid w \leq u \text{ or } u \leq w\}$) and by $\text{Max}_{\leq}(U)$ the set of \leq -maximal worlds of U ($\text{Max}_{\leq}(U) := \{v \in U \mid \text{for all } u \in U, u \leq v\}$). Then, in a plausibility model, the accessibility relation \leq is asked to be a *locally well-preorder*: a reflexive and transitive relation such that, for each comparability class V_w and for every non-empty $U \subseteq V_w$, $\text{Max}_{\leq}(U) \neq \emptyset$. Note how the existence of maximal elements in every $U \subseteq V_w$ implies the already required reflexivity, but also *connectedness* inside V_w . In particular, if two worlds w_2 and w_3 are more plausible than a given w_1 ($w_1 \leq w_2$ and $w_1 \leq w_3$), then these two worlds should be \leq -related ($w_2 \leq w_3$ or $w_3 \leq w_2$ or both).

Interestingly, the agent's indistinguishability relation can be derived from the plausibility one. If two worlds are \leq -related, then though the agent considers one of them more plausible than the other, she cannot discard one of them when the other one is given. In other words, worlds that are \leq related are in fact epistemically indistinguishable.

For the language we have two options.¹ We can extend the propositional language with formulas of the form $B\varphi$, semantically interpreted as

$$(M, w) \Vdash B\varphi \quad \text{iff} \quad \text{for all } u \in W, u \in \text{Max}_{\leq}(R_{\leq}(w)) \text{ implies } (M, u) \Vdash \varphi, \\ \text{where } R_{\leq}(w) := \{u \in W \mid w \leq u\}.$$

¹In fact, the mentioned works, [12; 4; 3], use the notion of *conditional* belief as the primitive one, rather than plain belief. We have chosen to stick with the notion of plain belief through the present notes, leaving an analysis of the notions of implicit/explicit conditional beliefs for further work.

The second option is to use a standard modal language with $[\leq]$ standing for the relation \leq , and then define beliefs in terms of it. Given the properties of \leq (in particular, reflexivity, transitivity and connectedness), it is not hard to see that φ is true in the most plausible worlds from w iff w can see a better world from which all successors are φ worlds. This yields the following definition for “the agent believes φ ”:

$$B\varphi := \langle \leq \rangle [\leq] \varphi$$

2.3 Combining the models

Our framework for representing implicit and explicit beliefs combines the mentioned ideas. The language has two components: formulas and rules. Formulas are given by a propositional language extended, first, with modalities $\langle \leq \rangle$ and $\langle \simeq \rangle$, and second, with formulas of the form $A \varphi$ and $R \rho$, where φ is a formula and ρ a rule. Rules, on the other hand, are pairs consisting of a set of formulas, the rule’s premises, and a single formula, the rule’s conclusion. The formal definition of our language is as follows.

Definition 2.1 (Language \mathcal{L}). Given a set of atomic propositions P , formulas φ and rules ρ of the *plausibility-access* language \mathcal{L} are given, respectively, by

$$\begin{aligned} \varphi &::= p \mid A \varphi \mid R \rho \mid \neg \varphi \mid \varphi \vee \psi \mid \langle \simeq \rangle \varphi \mid \langle \leq \rangle \varphi \\ \rho &::= (\{\varphi_1, \dots, \varphi_{n_\rho}\}, \psi) \end{aligned}$$

where $p \in P$. Formulas of the form $A \varphi$ are read as “the agent has acknowledged that formula φ is true”, and formulas of the form $R \rho$ as “the agent has acknowledged that rule ρ is truth-preserving”. For the modalities, $\langle \leq \rangle \varphi$ is read as “there is a more plausible world where φ holds”, and $\langle \simeq \rangle \varphi$ as “there is an epistemically indistinguishable world where φ holds”. Other boolean connectives as well as the box modalities $[\simeq]$ and $[\leq]$ are defined as usual. We denote by \mathcal{L}_f the set of formulas of \mathcal{L} , and by \mathcal{L}_r its set of rules.

Though rules are usually presented as schemas, our rules are defined as particular instantiations (e.g., the rule $(\{p \wedge q\}, p)$ is different from the rule $(\{q \wedge r\}, q)$). Since they will be applied in a generalized modus ponens form (if the agent has all the premises, she can derive the conclusion), using concrete formulas avoids details of instantiation, therefore facilitating the definition. When dealing with them, the following definitions will be useful.

Definition 2.2. Given a rule ρ , we will denote its *set of premises* by $\text{pm}(\rho)$, its *conclusion* by $\text{cn}(\rho)$, and its *translation* (an implication whose antecedent is the finite conjunction of ρ ’s premises and whose consequent is ρ ’s conclusion) by $\text{tr}(\rho)$.

For the semantic model, we will extend the described plausibility models with two functions.

Definition 2.3 (Plausibility-access model). With P the set of atomic propositions, a *plausibility-access (PA) model* is a tuple $M = \langle W, \leq, V, A, R \rangle$ where $\langle W, \leq, V \rangle$ is a plausibility model over P and

- $A : W \rightarrow \wp(\mathcal{L}_f)$ is the *access set function*, assigning to the agent a set of formulas of \mathcal{L} in each possible world,
- $R : W \rightarrow \wp(\mathcal{L}_r)$ is the *rule set function*, assigning to the agent a set of rules of \mathcal{L} in each possible world.

Functions A and R can be seen as valuations with a particular range, assigning to the agent a set of formulas and a set of rules at each possible world, respectively. Moreover, recall that two worlds that are \leq related are epistemically indistinguishable, so we define \simeq as the union of \leq and its converse ($\simeq := \leq \cup \geq$): the agent cannot distinguish between two worlds if she considers one of them more plausible than the other.

A *pointed plausibility-access model* (M, w) is a plausibility-access model with a distinguished world $w \in W$.

Here it is important to emphasize our interpretation of the A -sets. Different from [17] and [10], we do not interpret them as “the formulas the agent is aware of at world w ”, but rather as “the formulas the agent has acknowledged as true at world w ”, closer to the ideas in [15; 22; 18].

Now the semantic evaluation. The modalities $\langle \leq \rangle$ and $\langle \simeq \rangle$ are interpreted via their correspondent relation in the usual way, and formulas of the form $A\varphi$ and $R\rho$ are interpreted with our two new functions.

Definition 2.4 (Semantic interpretation). Let (M, w) be a pointed PA model with $M = \langle W, \leq, V, A, R \rangle$. Atomic propositions and boolean operators are interpreted as usual. For the remaining cases,

$$\begin{aligned} (M, w) \Vdash A\varphi & \quad \text{iff } \varphi \in A(w) \\ (M, w) \Vdash R\rho & \quad \text{iff } \rho \in R(w) \\ (M, w) \Vdash \langle \leq \rangle \varphi & \quad \text{iff there is a } u \in W \text{ such that } w \leq u \text{ and } (M, u) \Vdash \varphi \\ (M, w) \Vdash \langle \simeq \rangle \varphi & \quad \text{iff there is a } u \in W \text{ such that } w \simeq u \text{ and } (M, u) \Vdash \varphi \end{aligned}$$

For characterizing valid formulas, an important observation is that a locally well-preorder is a *locally connected and conversely well-founded pre-order* [3]. Then, by standard results on canonicity and modal correspondence (Chapter 4 of [11]), the axiom system of Section 2.6 of [3] (Table 1) is also sound and (weakly) complete for our language \mathcal{L} with respect to ‘non-standard’ plausibility-access models: those in which \leq is reflexive, transitive and locally connected (axioms T_\leq , 4_\leq and LC) and \simeq the symmetric extension of \leq (axioms T_\simeq , 4_\simeq , B_\simeq and Inc). But such models also have the *finite model* property (with respect to formulas in our language), so completeness

<i>Prop</i>	$\vdash \varphi$ for φ a propositional tautology	<i>MP</i>	If $\vdash \varphi \rightarrow \psi$ and $\vdash \varphi$, then $\vdash \psi$
K_{\leq}	$\vdash [\leq](\varphi \rightarrow \psi) \rightarrow ([\leq]\varphi \rightarrow [\leq]\psi)$	K_{\simeq}	$\vdash [\simeq](\varphi \rightarrow \psi) \rightarrow ([\simeq]\varphi \rightarrow [\simeq]\psi)$
$Dual_{\leq}$	$\vdash \langle \leq \rangle \varphi \leftrightarrow \neg[\leq]\neg\varphi$	$Dual_{\simeq}$	$\vdash \langle \simeq \rangle \varphi \leftrightarrow \neg[\simeq]\neg\varphi$
Nec_{\leq}	If $\vdash \varphi$, then $\vdash [\leq]\varphi$	Nec_{\simeq}	If $\vdash \varphi$, then $\vdash [\simeq]\varphi$
T_{\leq}	$\vdash [\leq]\varphi \rightarrow \varphi$	T_{\simeq}	$\vdash [\simeq]\varphi \rightarrow \varphi$
4_{\leq}	$\vdash [\leq]\varphi \rightarrow [\leq][\leq]\varphi$	4_{\simeq}	$\vdash [\simeq]\varphi \rightarrow [\simeq][\simeq]\varphi$
		B_{\simeq}	$\vdash \varphi \rightarrow [\simeq]\langle \simeq \rangle \varphi$
<i>LC</i>	$\langle \langle \simeq \rangle \varphi \wedge \langle \simeq \rangle \psi \rangle \rightarrow \langle \langle \simeq \rangle (\varphi \wedge \langle \leq \rangle \psi) \vee \langle \simeq \rangle (\psi \wedge \langle \leq \rangle \varphi) \rangle$		
<i>Inc</i>	$\langle \leq \rangle \varphi \rightarrow \langle \simeq \rangle \varphi$		

Table 1: Axiom system for \mathcal{L} with respect to plausibility-access models.

with respect to plausibility-access models follows from the fact that every strict preorder is conversely well-founded.

Note how the axiom system does not have axioms for formulas of the form $A\varphi$ and $R\rho$. This is because, as mentioned before, such formulas are simply special *atoms* for the dedicated *valuation functions* A and R . Moreover, we have not asked for the A - and R -sets to have any special closure property and there is no restriction in the way they interact with each other.² Just like axiom systems for Epistemic Logic do not require special axioms describing the behaviour of atomic propositions (unless, of course, they have special properties, like q being true every time p is, characterized by $p \rightarrow q$), our system does not require special axioms for these special atoms. More precisely, in the canonical model construction, we only need to define access and rule sets in the proper way:

$$A(w) := \{\varphi \in \mathcal{L}_f \mid A\varphi \in w\} \quad R(w) := \{\rho \in \mathcal{L}_r \mid R\rho \in w\}$$

Then, formulas of the form $A\varphi$ and $R\rho$ also satisfy the crucial *Truth Lemma*, and completeness follows. Again, see Chapter 4 of [11] for details.

2.3.1 Implicit and explicit beliefs

It is time to define the notions of implicit and explicit beliefs. Our definitions, shown in Table 2, combine ideas from [3], [18] and [10]. Note how the agent believes the formula φ (the rule ρ) implicitly iff φ ($\text{tr}(\rho)$) is true in the most plausible worlds, but in order to believe it explicitly, the agent should also acknowledge φ (ρ) as true (truth-preserving) in these ‘best’ worlds.

Explicit beliefs are implicit beliefs, witness the following validities:

$$B_{\text{Ex}}\varphi \rightarrow B_{\text{Im}}\varphi \quad B_{\text{Ex}}\rho \rightarrow B_{\text{Im}}\rho$$

²In [17], the authors explore and characterize several closure properties of A -sets.

The agent <i>implicitly</i> believes formula φ	$B_{\text{Im}}\varphi := \langle \leq \rangle [\leq] \varphi$
The agent <i>explicitly</i> believes formula φ	$B_{\text{Ex}}\varphi := \langle \leq \rangle [\leq] (\varphi \wedge A\varphi)$
The agent <i>implicitly</i> believes rule ρ	$B_{\text{Im}}\rho := \langle \leq \rangle [\leq] \text{tr}(\rho)$
The agent <i>explicitly</i> believes rule ρ	$B_{\text{Ex}}\rho := \langle \leq \rangle [\leq] (\text{tr}(\rho) \wedge R\rho)$

Table 2: Implicit and explicit beliefs about formulas and rules.

A possibly more interesting point is the following. An agent in [17; 10] is non-omniscient due to lack of attention; she does not need to be *aware of* every formula. On the other hand, our agent is aware of all formulas, but still she is non-omniscient because she does not need to be *aware that* a formula is true. This may seem a small difference, but the interpretation of the A sets determines the reasonable operations over them. An agent can become aware of any formula at any time, so any formula can be added to the A -sets without further requirement [10]. On the other hand, it is a stretch to assume that an agent can recognize as true any formula at any moment; it is more reasonable to ask for some derivation device, which in this work will be a *rule application* [15; 22; 18].

We finish this section by mentioning some properties of implicit and explicit beliefs about formulas. (Rules behave in a similar way.) Implicit beliefs are closed under logical consequence: if the most plausible worlds satisfy both $\varphi \rightarrow \psi$ and φ , then they also satisfy ψ . But explicit beliefs do not need to have this property because the A -sets do not need to have any closure property: having φ and $\varphi \rightarrow \psi$ does not guarantee to have ψ .

Though \leq is reflexive, neither implicit nor explicit beliefs have to be true because the real world does not need to be among the most plausible ones. Nevertheless, reflexivity makes implicit (and therefore explicit) beliefs consistent. Every world has at least one \leq -successor, so $\neg B_{\text{Im}}\perp$ is valid.

Implicit beliefs are positively and negatively introspective. This is the case because the notion of ‘most plausible worlds’ is global inside the same comparability class. For positive introspection, if the set of maximal worlds contains only φ -worlds ($B_{\text{Im}}\varphi$), so does the maximal from the maximal ones ($B_{\text{Im}}B_{\text{Im}}\varphi$). And for negative introspection, if there is a $\neg\varphi$ -world u in the maximal worlds, ($\neg B_{\text{Im}}\varphi$), then u is also in those maximal from the maximal ones ($B_{\text{Im}}\neg B_{\text{Im}}\varphi$). But this does not extend to explicit beliefs, again because the A -sets do not need to have any closure property. Having φ does not guarantee to have $B_{\text{Ex}}\varphi$ (so $B_{\text{Ex}}\varphi \rightarrow B_{\text{Ex}}B_{\text{Ex}}\varphi$ is not valid), and *not* having φ does not guarantee to have $\neg B_{\text{Ex}}\varphi$ (so $\neg B_{\text{Ex}}\varphi \rightarrow B_{\text{Ex}}\neg B_{\text{Ex}}\varphi$ is not valid).

3 Dynamics part one: upgrade and retraction

We have a framework for representing implicit and explicit beliefs. We now look at their *dynamics* by introducing two actions that modify them.

3.1 Explicit upgrade

The χ -upgrade operation [4; 3] modifies the plausibility relation \leq to put the χ -worlds at the top, therefore *revising* the agent's beliefs. Here we have two possibilities, depending on whether it also adds χ to the A-sets (*explicit upgrade*) or not (*implicit upgrade*). Here is the definition of the first case.

Definition 3.1 (Explicit upgrade). Let $M = \langle W, \leq, V, \mathbf{A}, \mathbf{R} \rangle$ be a PA model and χ a formula in \mathcal{L} . The PA model $M_{\chi\uparrow^+} = \langle W, \leq', V, \mathbf{A}', \mathbf{R} \rangle$ differs from M in the plausibility relation and in the access set function:

$$\begin{aligned} \leq' &:= (\leq; \chi?) \cup (\neg\chi?; \leq) \cup (\neg\chi?; \simeq; \chi?) \\ \mathbf{A}'(w) &:= \mathbf{A}(w) \cup \{\chi\} \quad \text{for every } w \in W \end{aligned}$$

Note how the *upgrade* operation is functional: for every model M it returns one and only one model $M_{\chi\uparrow^+}$.

The new plausibility relation is given in a PDL style: we have $w \leq' u$ iff (1) $w \leq u$ and u is a χ -world, or (2) w is a $\neg\chi$ -world and $w \leq u$, or (3) $w \simeq u$, w is a $\neg\chi$ -world and u is a χ -world. There are other possible definitions for \leq' [4; 3], and the chosen one, so-called *radical upgrade*, is just an example of what can be defined.

The operation preserves models in the intended class.

Proposition 1. *If M is a PA model, then so is $M_{\chi\uparrow^+}$.*

We extend the language to express the effect of an explicit upgrade; formulas of the form $\langle \chi \uparrow^+ \rangle \varphi$ are read as “it is possible to perform an explicit χ -upgrade after which φ holds”. There is no precondition for this action (the agent can perform an explicit upgrade whenever she wants), so the semantic interpretation is as follows.

Definition 3.2. Let (M, w) be a pointed PA model:

$$(M, w) \Vdash \langle \chi \uparrow^+ \rangle \varphi \quad \text{iff} \quad (M_{\chi\uparrow^+}, w) \Vdash \varphi$$

Note how the operation puts on top those worlds that are χ -worlds in the original model, but they do not need to be χ -ones after the upgrade. The plausibility relation changes, therefore changing the truth-value of formulas containing the modalities for \leq and/or \simeq and, in particular, changing the agent's beliefs. This is not strange at all, and in fact it corresponds to the

well-known Moore-like sentences (“ p is the case and you do not know it”) in *Public Announcement Logic* that become false after being announced, and therefore cannot be known.

Nevertheless, the operation behaves as expected for propositional formulas. The operation does not change valuations, so if χ is purely propositional, the operation will put current χ -worlds on top, and they will still be χ -worlds after the operation so the agent will believe χ .

The validities in our new language can be axiomatized by using *reduction axioms*, valid formulas that indicate how to translate a formula with the new modality $\langle \chi \uparrow^+ \rangle$ into a provably equivalent one without them. Then, completeness follows from the completeness of the basic system. We refer to [8] for an extensive explanation of this technique.

Theorem 1. *The axiom system of Table 1 together with axioms and rules of Table 3 (with \top the always true formula) provide a sound and (weakly) complete axiom system for formulas in the language \mathcal{L} plus the explicit upgrade modality with respect to plausibility-access models.*

$\vdash \langle \chi \uparrow^+ \rangle p \leftrightarrow p$	$\vdash \langle \chi \uparrow^+ \rangle A\chi \leftrightarrow \top$
$\vdash \langle \chi \uparrow^+ \rangle \neg\varphi \leftrightarrow \neg\langle \chi \uparrow^+ \rangle \varphi$	$\vdash \langle \chi \uparrow^+ \rangle A\varphi \leftrightarrow A\varphi \quad \text{for } \varphi \neq \chi$
$\vdash \langle \chi \uparrow^+ \rangle (\varphi \vee \psi) \leftrightarrow (\langle \chi \uparrow^+ \rangle \varphi \vee \langle \chi \uparrow^+ \rangle \psi)$	$\vdash \langle \chi \uparrow^+ \rangle R\rho \leftrightarrow R\rho$
$\vdash \langle \chi \uparrow^+ \rangle \langle \leq \rangle \varphi \leftrightarrow \langle \leq \rangle (\chi \wedge \langle \chi \uparrow^+ \rangle \varphi) \vee (\neg\chi \wedge \langle \leq \rangle \langle \chi \uparrow^+ \rangle \varphi) \vee (\neg\chi \wedge \langle \simeq \rangle (\chi \wedge \langle \chi \uparrow^+ \rangle \varphi))$	
$\vdash \langle \chi \uparrow^+ \rangle \langle \simeq \rangle \varphi \leftrightarrow \langle \simeq \rangle \langle \chi \uparrow^+ \rangle \varphi$	
From $\vdash \varphi$ infer $\vdash \langle \chi \uparrow^+ \rangle \varphi$	

Table 3: Axioms and rules for explicit upgrade formulas.

The reduction axioms simply indicate how each kind of formula is affected by the explicit upgrade operation. For example, $\langle \chi \uparrow^+ \rangle p \leftrightarrow p$ states that atomic propositions are not affected, and both $\langle \chi \uparrow^+ \rangle A\chi \leftrightarrow \top$ and $\langle \chi \uparrow^+ \rangle A\varphi \leftrightarrow A\varphi$ for $\varphi \neq \chi$ together state that χ and only χ is added to the **A**-sets. The interesting axiom is the one for the plausibility modality $\langle \leq \rangle$. It is obtained with techniques from [9], and simply translates the three-cases *PDL* definition of the new plausibility relation: after an upgrade with χ there is a \leq -reachable world where φ holds iff before the operation (1) there is a \leq -reachable χ -world that will become φ after the upgrade, or (2) the current is a $\neg\chi$ -world that can \leq -reach another that will turn into a φ -one after the operation, or (3) the current is a $\neg\chi$ -world that can \simeq -reach another that is χ and will become φ after the upgrade. Similar reduction axioms have been presented in [9] in the context of preference upgrade.

3.2 Retraction

But there are also situations in which the agent simply retracts some explicit belief, that is, she does not acknowledge it as true anymore. This is achieved simply by removing the formula from the \mathbf{A} -sets.

Definition 3.3 (Retraction). Let $M = \langle W, \leq, V, \mathbf{A}, \mathbf{R} \rangle$ be a PA model and χ a formula in \mathcal{L} . The PA model $M_{-\chi} = \langle W, \leq, V, \mathbf{A}', \mathbf{R} \rangle$ differs from M just in the access set function, given for every $w \in W$ as

$$\mathbf{A}'(w) := \mathbf{A}(w) \setminus \{\chi\}$$

Again, the *retraction* operation is functional. Moreover, it does not modify \leq , so it preserves plausibility models.

This operation is represented in the language by formulas of the form $\langle -\chi \rangle \varphi$, read as “it is possible to retract χ and after it φ holds”. Just like an upgrade, no precondition is needed.

Definition 3.4. Let (M, w) be a pointed PA model:

$$(M, w) \Vdash \langle -\chi \rangle \varphi \quad \text{iff} \quad (M_{-\chi}, w) \Vdash \varphi$$

Our definition behaves as we intend, witness the validity of $\langle -\chi \rangle \neg B_{\text{Ex}} \chi$: after retracting χ , the agent will not believe it explicitly.

For an axiom system, we can use reduction axioms again. In this case the axioms are simpler since only \mathbf{A} -sets are affected.

Theorem 2. *The axiom system of Table 1 together with axioms and rules of Table 4 (with \perp the always false formula) provide a sound and (weakly) complete axiom system for formulas in the language \mathcal{L} plus the retraction modality with respect to plausibility-access models.*

$\vdash \langle -\chi \rangle p \leftrightarrow p$	$\vdash \langle -\chi \rangle \mathbf{A} \chi \leftrightarrow \perp$
$\vdash \langle -\chi \rangle \neg \varphi \leftrightarrow \neg \langle -\chi \rangle \varphi$	$\vdash \langle -\chi \rangle \mathbf{A} \varphi \leftrightarrow \mathbf{A} \varphi \quad \text{for } \varphi \neq \chi$
$\vdash \langle -\chi \rangle (\varphi \vee \psi) \leftrightarrow (\langle -\chi \rangle \varphi \vee \langle -\chi \rangle \psi)$	$\vdash \langle -\chi \rangle \mathbf{R} \rho \leftrightarrow \mathbf{R} \rho$
$\vdash \langle -\chi \rangle \langle \leq \rangle \varphi \leftrightarrow \langle \leq \rangle \langle -\chi \rangle \varphi$	
$\vdash \langle -\chi \rangle \langle \simeq \rangle \varphi \leftrightarrow \langle \simeq \rangle \langle -\chi \rangle \varphi$	
From $\vdash \varphi$ infer $\vdash \langle -\chi \rangle \varphi$	

Table 4: Axioms and rules for retraction formulas.

Here the key axioms are $\langle -\chi \rangle \mathbf{A} \chi \leftrightarrow \perp$ and $\langle -\chi \rangle \mathbf{A} \varphi \leftrightarrow \mathbf{A} \varphi$ for $\varphi \neq \chi$, stating that χ and only χ is removed from the \mathbf{A} -sets. Similar reduction axioms have been presented in [10] in the context of dynamics of awareness.

4 Dynamics part two: inference on beliefs

We now turn into the main part of our work. In this section we analyze *rule-based inference on beliefs*. We start by recalling the case of rule-based inference on *knowledge* [18].

The definition of implicit and explicit knowledge are simpler than those for beliefs since they depend directly on all the worlds the agent considers possible. The agent knows φ implicitly when it holds in all the worlds she considers possible, and she knows φ explicitly when she also recognizes it as true in all such worlds. The definitions of explicit knowledge about a rule ρ is given in a similar way.

$$\begin{aligned} K_{\text{Im}}\varphi &:= [\simeq] \varphi & K_{\text{Ex}}\varphi &:= [\simeq] (\varphi \wedge A \varphi) \\ K_{\text{Im}}\rho &:= [\simeq] \text{tr}(\rho) & K_{\text{Ex}}\rho &:= [\simeq] (\text{tr}(\rho) \wedge R \rho) \end{aligned}$$

The action of *inference on knowledge* with rule σ is defined as an operation that adds σ 's conclusion to the A -set of those worlds where the agent knows explicitly σ and its premises ($K_{\text{Ex}}\sigma \wedge K_{\text{Ex}}\text{pm}(\sigma)$). More precisely, if M is a plausibility-access model with access set function A , then the operation of σ -inference on knowledge produces the model $M_{\hookrightarrow_{\sigma}^K}$, differing from M just in the access set function A' , which is given by

$$A'(w) := \begin{cases} A(w) \cup \{\text{cn}(\sigma)\} & \text{if } (M, w) \Vdash K_{\text{Ex}}\sigma \wedge K_{\text{Ex}}\text{pm}(\sigma) \\ A(w) & \text{otherwise} \end{cases}$$

A new modality $\langle \hookrightarrow_{\sigma}^K \rangle$ is introduced to express the effects of this operation, and its semantic definition is given by

$$(M, w) \Vdash \langle \hookrightarrow_{\sigma}^K \rangle \varphi \quad \text{iff} \quad (M, w) \Vdash K_{\text{Ex}}\sigma \wedge K_{\text{Ex}}\text{pm}(\sigma) \quad \text{and} \quad (M_{\hookrightarrow_{\sigma}^K}, w) \Vdash \varphi$$

But take a closer look at the inference on knowledge operation. What it actually does is to discard all worlds where $K_{\text{Ex}}\sigma \wedge K_{\text{Ex}}\text{pm}(\sigma)$ holds, and replace them with copies that are almost identical, the only difference being their A -sets that, after the operation, will have $\text{cn}(\sigma)$. And this is reasonable because, under the assumption that knowledge is true information, inference based on a known (therefore truth-preserving) rule with known (therefore true) premises is simply deductive reasoning: the premises are true and the rule preserves the truth, so the conclusion *should* be true. In fact, inference based on a known rule with known premises is the act of recognizing two things. First, since the applied rule is truth-preserving and its premises are true, its conclusion *must* be true; and second, situations where the premises are true but the conclusion is not *are not possible*.

The case of beliefs is different, as suggested in [6]. An inference on beliefs is based on a rule that is believed to be truth-preserving, but that it is not necessarily so. Even though it is reasonable to consider a situation in which the premises and the conclusion hold, the agent should not discard a situation where the premises hold but the conclusion does not.

Our proposal is the following. An inference on beliefs should create two copies of each world where the rule and the premises are believed: an exact copy of the original one, and another extending it by adding the rule's conclusion to it. But not only that. The agent believes that the rule is truth-preserving and the premises are true, so the extended world should be more plausible than the 'conclusionless' one.

But, how to create copies of a possible world? We can use the *action models* and *product update* of the so-called *BMS* approach [2].

4.1 Plausibility-access action models

The main idea behind *action models* [2] is that actions can be represented with a model similar to that used for representing the static situation. In other words, just as the agent can be uncertain about which one is the real world, she can also be uncertain about which action has taken place. Then, the uncertainty of the agent *after an action* is a combination of her uncertainty about the situation *before the action* and her uncertainty *about the action* itself.

This idea has been extended in two different directions: in order to deal with plausibility models [3] and in order to deal with non-omniscient multi-agent situations [10]. Our proposal combines and extends these two ideas, now with the aim of dealing with single-agent inference on beliefs. We start by defining the structures that will represent this kind of actions.

Definition 4.1 (Plausibility-access action model). A *plausibility-access action model* is a tuple $A = \langle S, \preceq, \text{Pre}, \text{Pos}_A, \text{Pos}_R \rangle$ where

- $\langle S, \preceq, \text{Pre} \rangle$ is a plausibility action model [3] with S a finite non-empty set of *events*, \preceq a *plausibility* relation on S (with the same requirements as those for a plausibility-access model) and $\text{Pre} : S \rightarrow \mathcal{L}_f$ a *precondition* function indicating the requirement for each event to be executed.
- $\text{Pos}_A : (S \times \wp(\mathcal{L}_f)) \rightarrow \wp(\mathcal{L}_f)$ is the *new access set* function, which will allow us to define the *access set* of the agent in the model that will result from applying this action.
- $\text{Pos}_R : (S \times \wp(\mathcal{L}_r)) \rightarrow \wp(\mathcal{L}_r)$ is the *new rule set* function, which will allow us to define the *rule set* of the agent in the model that will result from applying this action.

Just as before, the plausibility relation \preceq defines an equivalence relation by putting it together with its converse: $\approx := \preceq \cup \succeq$. A pointed plausibility-access action model (A, s) has a distinguished event $s \in S$.

Examples of plausibility-access action models will be shown in Section 4.2. But first we will define the plausibility-access model that results from an action model application as well as the formula that will represent this operation and its semantic interpretation.

Definition 4.2 (Product update). Let $M = \langle W, \leq, V, A, R \rangle$ be a PA model and $A = \langle S, \leq, \text{Pre}, \text{Pos}_A, \text{Pos}_R \rangle$ be a PA action model. The PA model $M \otimes A = \langle W', \leq', V', A', R' \rangle$ is given by

- $W' := \{(w, s) \in (W \times S) \mid (M, w) \models \text{Pre}(s)\}$
- $(w_1, s_1) \leq' (w_2, s_2)$ iff $(s_1 < s_2 \text{ and } w_1 \simeq w_2)$ or $(s_1 \approx s_2 \text{ and } w_1 \leq w_2)$
- $V'(w, s) := V(w)$
- $A'(w, s) := \text{Pos}_A(s, A(w))$
- $A'(w, s) := \text{Pos}_R(s, R(w))$

Note how the set of worlds of the new plausibility-access model is given by the restricted cartesian product of W and S ; a pair (w, s) will be a world in the new model iff event s can be executed at world w . The new plausibility order follows the so-called ‘Action-priority’ rule [3], making (w_2, s_2) more plausible than (w_1, s_1) iff either s_2 is strictly more plausible than s_1 and w_1, w_2 are indistinguishable, or else s_1, s_2 are indistinguishable and w_2 is more plausible than w_1 .

Now, for the valuations of the new worlds. First, a new world inherits the atomic valuation of its static component, that is, an atom p holds at (w, s) iff p holds at w . The cases for access sets gives us full generality: the access set of world (w, s) is given by the function Pos_R with the event s and the access set of w as parameters [10]. The case for rule sets is similar.

It is not hard to verify that the *product update* operation preserves plausibility-access models.

Proposition 2. *If M is a plausibility-access model and A a plausibility-access action model, then $M \otimes A$ is a plausibility-access model.*

In order to express how product updates affect the agents’ information, we extend our language with modalities for each pointed plausibility-access action model (A, s) , allowing us to build formulas of the form $\langle A, s \rangle \varphi$, whose semantic interpretation is given below.

Definition 4.3. Let (M, w) be a pointed PA model and let (A, s) be a pointed PA action model with Pre its precondition function.

$$(M, w) \models \langle A, s \rangle \varphi \quad \text{iff} \quad (M, w) \models \text{Pre}(s) \text{ and } (M \otimes A, (w, s)) \models \varphi$$

4.2 Plausibility-access action models for basic inference

The action of inference on knowledge can be represented with plausibility-access action models.

Definition 4.4 (Inference on knowledge). Let σ be a rule. The action of *inference on knowledge* is given by the pointed *PA* action model $(A_{\hookrightarrow_\sigma^k}, s)$ whose definition (left) and relevant diagram (right) are given by

$$\begin{array}{l} \bullet S := \{s\} \quad \bullet \leq := \{(s, s)\} \quad \bullet \text{Pre}(s) := K_{\text{Ex}}\sigma \wedge K_{\text{Ex}}\text{pm}(\sigma) \\ \bullet \text{Pos}_A(s, X) := X \cup \{\text{cn}(\sigma)\} \quad \bullet \text{Pos}_R(s, Y) := Y \end{array} \quad \begin{array}{c} \hline \curvearrowright \\ s \quad X \cup \{\text{cn}(\sigma)\} \\ \hline \end{array}$$

But now we can represent more. Following our previous discussion, here is the action model for basic inference on beliefs.

Definition 4.5 (Basic inference on beliefs). Let σ be a rule. The action of *basic inference on belief* is given by the pointed *PA* action model $(A_{\hookrightarrow_\sigma^b}, s_1)$ whose definition is

$$\begin{array}{l} \bullet S := \{s_1, s_2\} \\ \bullet \leq := \{(s_1, s_1), (s_1, s_2), (s_2, s_2)\} \\ \bullet \begin{cases} \text{Pre}(s_1) := \text{Pre}_{B\sigma} \\ \text{Pre}(s_2) := \text{Pre}_{B\sigma} \end{cases} \end{array} \quad \bullet \begin{cases} \text{Pos}_A(s_1, X) := X \\ \text{Pos}_A(s_2, X) := X \cup \{\text{cn}(\sigma)\} \\ \text{Pos}_R(s_1, Y) := Y \\ \text{Pos}_R(s_2, Y) := Y \end{cases}$$

The precondition is that the agent believes explicitly the rule and its premises, that is,

$$\text{Pre}_{B\sigma} := B_{\text{Ex}}\sigma \wedge B_{\text{Ex}}\text{pm}(\sigma)$$

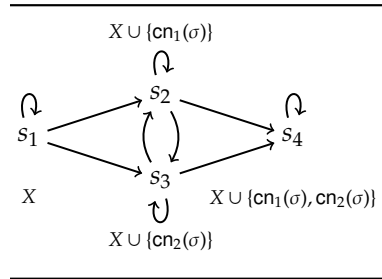
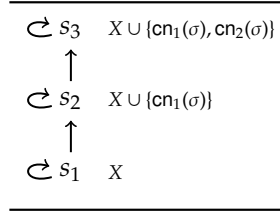
$$\begin{array}{c} \hline \curvearrowleft s_2 \quad X \cup \{\text{cn}(\sigma)\} \\ \uparrow \\ \curvearrowleft s_1 \quad X \\ \hline \end{array}$$

The relevant diagram appears on the right.

4.3 Extended inference: an exploration

Plausibility-access action models allow us to represent more than what we have discussed. As observed in [3], a plausibility relation generates a Grove's system-of-spheres, that is, several layers of possible events ordered according to their plausibility. The presented action models for basic inference on beliefs are just those models with two layers, each one of them having just one event, and with the most plausible one being the extended one. But we do not have to restrict ourselves to such kind of inferences.

Action models with more than two layers allow us to represent inference based on rules with more than one conclusion. The action model on the right has three layers, each one containing one event. Event s_1 preserves access sets, s_2 extends them with the first conclusion and s_3 extends them with both conclusions.



And we can do more by using layers with more than one world, like the action model on the left that allows the agent to have $cn_2(\sigma)$ without having $cn_1(\sigma)$.

So far our examples have one characteristic in common. The *new access set* function is monotone, reflecting the optimism of the agent with respect to the conclusion: events that extend A-sets are always more plausible.

Definition 4.6 (Plausibility-access action models for optimistic inference). Plausibility-access action models in which, for every event s_1, s_2 ,

$$s_1 \preceq s_2 \text{ implies } \text{Pos}_A(s_1, X) \subseteq \text{Pos}_A(s_2, X)$$

are called action models for *optimistic inference*.

But then we can also consider the opposite case. Models with an *anti-monotone new access set* function reflect the pessimism of the agent with respect to the conclusion: events that extend A-sets are always less plausible.

Definition 4.7 (Plausibility-access action models for pessimistic inference). Plausibility-access action models in which, for every event s_1, s_2 ,

$$s_1 \preceq s_2 \text{ implies } \text{Pos}_A(s_1, X) \supseteq \text{Pos}_A(s_2, X)$$

are called action models for *pessimistic inference*.

Of course these two classes do not cover all possibilities. Plausibility-access action models allow us to represent many different and complex inferences whose detailed study has to be left for further work.

4.4 Brief discussion on completeness

The reduction axioms of [3] are inherited by our system. In particular, the following one states the way the plausibility relation changes:

$$\langle A, s \rangle \langle \leq \rangle \varphi \leftrightarrow \left(\text{Pre}(s) \wedge \left(\bigvee_{s \approx s'} \langle A, s' \rangle \varphi \vee \bigvee_{s \approx s''} \langle A, s'' \rangle \varphi \right) \right)$$

But when looking for reduction axioms for access and rule set formulas, Pos_A and Pos_R pose a problem. The reason is that they allow the new access and rule sets to be arbitrary sets. Compare this with other product update definitions. The one of [7] can change the atomic valuation, but the set of worlds in which a given atomic proposition will be true should be given by a formula of the language; the one of [16] can change the relation in a point-wise way, but the new relation is given in terms of the previous ones by using only regular operations. Our current efforts focus on particular definitions expressive enough to describe our desired inferences and restricted enough to get the needed reduction axioms.

5 Conclusions and further work

We have presented a framework for representing implicit and explicit beliefs. We have also provided representations of three actions that modify them, starting with those of *explicit upgrade* and *retraction* but, more important, discussing intuitive ideas and proposing a rich framework for representing the action of *inference on beliefs*.

There are parts of this work that deserve further exploration, the most appealing being the study of the different kind of inferences that we can represent with plausibility-access action models. We have defined those for inference on knowledge and basic inference on beliefs, and we have briefly explored some others, but our structures can represent much more. Another interesting extension is to look at dynamics of rules, that is, to look for reasonable actions that extend not only the rules the agent knows [28], but also the rules she believes. These two studies will not be complete without the appropriate axiom system for our product update definition. Finally we mention a third direction: the study of a multi-agent setting, including not only the addition of more agents to the picture, but also the analysis of implicit/explicit versions of multi-agent notions, like common knowledge and common beliefs.

References

- [1] T. Ågotnes and N. Alechina, editors. *Special issue on Logics for Resource Bounded Agents*, 2009. *Journal of Logic, Language and Information*, 18(1).

- [2] A. Baltag, L. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicious. SEN-R9922, CWI, Amsterdam, 1999.
- [3] A. Baltag and S. Smets. A qualitative theory of dynamic interactive belief revision. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Logic and the Foundations of Game and Decision Theory (LOFT7)*, volume 3 of *Texts in Logic and Games*, pages 13–60. AUP, 2008.
- [4] J. van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007.
- [5] J. van Benthem. Merging observation and access in dynamic logic. *Journal of Logic Studies*, 1(1):1–17, 2008.
- [6] J. van Benthem. Logic, mathematics, and general agency. In P. Bour, M. Rebuschi, and L. Rollet, editors, *Festschrift for Gerhard Heinzmann*. Laboratoire d’histoire des sciences et de la philosophie, Nancy, 2009.
- [7] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.
- [8] J. van Benthem and B. Kooi. Reduction axioms for epistemic actions. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic (Technical Report UMCS-04-09-01)*, pages 197–211. University of Manchester, 2004.
- [9] J. van Benthem and F. Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 17(2):157–182, 2007.
- [10] J. van Benthem and F. R. Velázquez-Quesada. Inference, promotion, and the dynamics of awareness. PP-2009-43, ILLC, Universiteit van Amsterdam, 2009.
- [11] P. Blackburn, M. de Rijke, and I. Venema. *Modal Logic*. Cambridge University Press, 2001.
- [12] O. Board. Dynamic interactive epistemology. *Games and Economic Behavior*, 49(1):49–80, 2004.
- [13] H. van Ditmarsch, A. Herzig, J. Lang, and P. Marquis. Introspective forgetting. *Synthese (KRA)*, 169(2):405–423, 2009.
- [14] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library Series*. Springer, 2007.
- [15] H. N. Duc. *Resource-Bounded Reasoning about Knowledge*. PhD thesis, Institut für Informatik, Universität Leipzig, Leipzig, Germany, 2001.
- [16] J. van Eijck and Y. Wang. Propositional dynamic logic as a logic of belief revision. In W. Hodges and R. J. G. B. de Queiroz, editors, *WoLLIC*, volume 5110 of *LNCS*, pages 136–148. Springer, 2008.
- [17] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

- [18] D. Grossi and F. R. Velázquez-Quesada. *Twelve Angry Men: A study on the fine-grain of announcements*. In X. He, J. F. Horty, and E. Pacuit, editors, *LORI*, volume 5834 of *LNCS*, pages 147–160. Springer, 2009.
- [19] A. Grove. Two modellings for theory change. *Journal of Philosophical Logic*, 17(2):157–170, 1988.
- [20] J. Y. Halpern, editor. *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, March 1986*, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.
- [21] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca, N.Y., 1962.
- [22] M. Jago. Rule-based and resource-bounded: A new look at epistemic logic. In T. Ågotnes and N. Alechina, editors, *Proceedings of the Workshop on Logics for Resource-Bounded Agents*, pages 63–77, Malaga, Spain, 2006.
- [23] K. Konolige. Belief and incompleteness. Technical Report 319, SRI, 1984.
- [24] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In Halpern [20], pages 325–340.
- [25] H. J. Levesque. A logic of implicit and explicit belief. In *Proc. of AAAI-84*, pages 198–202, Austin, TX, 1984.
- [26] K. Segerberg. The basic dynamic doxastic logic of AGM. In M.-A. Williams and H. Rott, editors, *Frontiers in Belief Revision*, number 22 in Applied Logic Series, pages 57–84. Kluwer Academic Publishers, 2001.
- [27] M. Y. Vardi. On epistemic logic and logical omniscience. In Halpern [20], pages 293–305.
- [28] F. R. Velázquez-Quesada. Inference and update. *Synthese (KRA)*, 169(2):283–300, 2009.

A Construction of Logic-Constrained Functions with Respect to Awareness

Susumu Yamasaki

Department of Computer Science, Okayama University, Japan
email: yamasaki@momo.cs.okayama-u.ac.jp

Abstract

A logic-constrained function is motivated by modelling the behaviour of the electronic device (like a mobile phone) with respect to evocation caused by awareness. This paper presents an analysis to develop a logic-constrained function system. We firstly have the contradiction removal procedure for a proof system which is expected with *negation as failure* rule (as denoting *unawareness*) to derive the constraint (supposedly as *awareness*) with reference to the (abstract) functions like λ -terms. Though the system deriving constraints can be the logic programming system (Alferes, J.J. et al.) of coherence principle, a limited reasoning (with negation as failure) is treated in this paper to cope with inconsistency (contradiction) caused by classical negation, in terms of contradiction removal facility. We then have a description of the logic-constrained function. As a logic-constrained function, we have an outlook on a literal-constrained term, an extended λ -term, where the literal may be derivable from a proof system. The term conversions are defined such that a system originating from awareness may have abstract function applications, based on the literal-constrained term.

1 Introduction

As a ubiquitous system, we can assume the electronic device (like a mobile phone) in place of the PC in distributed environments through internets or wireless communications. (i) The device is expected to hold functional and objective knowledge abstractly. (ii) The device as a resource is bounded to be left alone until it is evoked and of use. The evocation of the device can be supported by constraint awareness of (a set of) states (as in [11]) which some predicate (with or without classical negation) may denote. That is, the evocation may be interpreted to be realized by an awareness of states (i.e., of a predicate). (iii) As long as we are concerned with awareness of a predicate for such a set of states, it is implementable as derivability from a proof system. (iv) In a proof system, negation as failure is applicable to unawareness, owing to derivability of the designated predicate. (Note

that the closed world assumption is broader than negation as failure with reference to unawareness.)

Therefore we have an illustrative structure on logical awareness as in Table 1.

Derivability basis		Evocation		Device
A proof system	\Rightarrow	Awareness	\Rightarrow	Function construction

Table 1: Logical awareness

We then have technical problems: (1) Function constructions constrained by predicates (constructions of logic-constrained function) are to be modelled for the mobile phone evoked by awareness of a state set. (2) If we regard awareness as caused by derivability from a proof system, paraconsistency should be denied. If a complementary pair of predicates with and without negation is derivable, one of them must be removed.

Regarding the problems, abstract functions have been deeply studied in the λ -calculus such that there were many compact textbooks among which we can see the one ([9]). From knowledge views on function applications, an analysis of the function (application) constrained by logic concerning awareness is also a problem. For the function to conceive logical awareness, the logic-constrained function may be constructed, where the logical formula (which constrains functions) is often interpreted as: procedure or process ([12]) and state set ([11]) possibly inducing space and/or time notions, such that the logical formula can be concerned with awareness of some agent. In this paper, the notion of the state set is adopted. The denotation of a state set is described by a predicate (with or without classical negation) which may be derived (deduced) from a proof system. As regards derivability, *negation as failure* in computational logic ([12]), the inference rule “ $\neg p \Rightarrow \sim p$ (or denoted as *not p*)” has been well studied. Unawareness may be defined in terms of negation as failure.

Motivated by the above problems, this paper is to analyze the logic-constrained function construction, where: (a) the logic programming system derives the literals with and without classical negation, by means of reasoning containing negation as failure, and (b) a logic-constrained function is defined by means of a combination of λ -terms with the logical formulae.

Regarding the logic programming system, the negative literals are as significant as positive ones, with reference to the closed world assumption, the default and/or negation as failure rules ([2, 17]). As an analysis, we need the negation as failure to denote unawareness with a contradiction-free reasoning. As before pointed out for the second problem, a complementary

pair of literals with and without classical negation causes a contradiction even with respect to awareness. That is, the contradiction is not allowed for logic-constrained function: the contradiction is to be removed from awareness view such that reasoning to remove contradiction must be restricted. Concerning the logic-constrained function, the established λ -calculus is fundamental and may be extended to the one where the (λ -)term constrained by logical formulae is definable. This paper focuses on the definable terms generally denoting the logic-constrained functions, where the logic-constrained function is a λ -term with respect to awareness expressed by a logical formula. The awareness is regarded as caused by derivability from a proof system, while negation as failure for derivability supposedly denotes unawareness.

The paper of the analysis for logic-constrained functions is organized for the problem as motivations described here. In Section 2, the logic programming system is reviewed from the propositional logic version, where an idea of the contradiction removal is presented. Section 3 presents a contradiction removal procedure and its soundness is shown. In Section 4, we have an outlook on an extended λ -term with the literal (possibly containing classical negation) and its conversions. Section 5 gives comments regarding logic-constraints on the λ -term.

2 Logical Expressions

We now review the terminologies in propositional logic programming, as a proof system to possibly derive literals with and without classical negation. Negation as failure is to denote an *unawareness*, while contradiction caused by a pair of derivable literals (*awareness*) with and without classical negation must be removed by a limited reasoning.

- (1) A set of symbols to stand for propositions is assumed.
- (2) Two kinds of negation sign are taken: the classical negation “ \neg ” and the negation as failure “ \sim ”.
- (3) A literal l is either an atom a , or a classical negation $\neg a$ of an atom a . An atom is an expression consisting of a symbol to denote a proposition.

Classical negation vs. negation as failure

The class of extended logic programs in the propositional logic is treated, where two kinds of negation may be contained. As a theory for the *literal-constrained term* which would be mentioned later, an extended logic program (ELP, for short) is a set of clauses of the form

$$l \leftarrow l_1, \dots, l_m, \sim l_{m+1}, \dots, \sim l_n \quad (0 \leq m \leq n),$$

where l and l_i are literals, and “ \sim ” stands for the negation as failure (NAF, for short). The literal $\sim l$ is called a negation-as-failure literal. The literal l of the clause is said to be its head, and the literal sequence $l_1, \dots, l_m, \sim l_{m+1}, \dots, \sim l_n$ is its body. The classical negation $\neg l$ means

- (i) $\neg a$ if $l = a$, and
- (ii) a if $l = \neg a$,

for an atom a . The pair of literals a and $\neg a$ is said to be complementary.

The expressions $L, L_1, \dots, L_m, \dots, M, M_1, \dots, M_n, \dots$ are reserved to denote literals or negation-as-failure literals. The expressions $\alpha, \alpha_1, \dots, \alpha_m, \dots, \beta, \beta_1, \dots, \beta_n, \dots$ are reserved to denote sequences of literals or negation-as-failure literals.

A goal is an expression of the form $\leftarrow l_1, \dots, l_m, \sim l_{m+1}, \dots, \sim l_n$ ($0 \leq m \leq n$), where l_i are literals. The goal of the form $\leftarrow \sim m_1, \dots, \sim m_q$ ($q \geq 0$) is said to be a negative goal. The negative goal is the empty clause, denoted by \square , if it contains no literal. For the ELPs, we must note a well established paraconsistent reasoning (which is regarded as a proof procedure to possibly derive literals) equipped with *coherence principle* ([1]):

“For any objective literal $\neg l$, if $\neg l$ is entailed by the semantics, then $\sim l$ is also entailed.”

In the paper, a proof procedure with coherence principle is given, while some condition for the program to be consistent is shown.

We have an outlook on SLD resolution and negation as failure, where we can see [12, 17] among so many papers.

SLD resolution applied to goals is a rule to derive

$$\leftarrow L_1, \dots, L_{i-1}, M_1, \dots, M_n, L_{i+1}, \dots, L_m$$

from a goal $\leftarrow L_1, \dots, L_{i-1}, l, L_{i+1}, \dots, L_m$ and a clause $l \leftarrow M_1, \dots, M_n$ in the given ELP. That is, a literal l of a goal may be replaced by the body M_1, \dots, M_n of a clause $l \leftarrow M_1, \dots, M_n$, whose head is just the literal l .

Negation as failure is a rule to infer a negation-as-failure literal $\sim l$, when the literal l is not derived by some proof procedure. We refine negation as failure in relation to SLD resolution such that the succeeding and failing (derivations) of a goal (with reference to a given ELP) are defined recursively as follows:

- (i) The goal $\leftarrow l$ succeeds if $\leftarrow l$ is reduced to \square by applying finitely many SLD resolutions.
- (ii) The goal $\leftarrow l$ fails if one of following conditions holds:
 - (a) there is no clause whose head is the literal l .
 - (b) there are goals $\leftarrow \alpha_1, L_1, \beta_1, \dots, \leftarrow \alpha_n, L_n, \beta_n$ ($n \geq 1$), derived by SLD resolutions for the goal $\leftarrow l$ such that all the goals $\leftarrow L_1, \dots, \leftarrow L_n$ fail.
 - (c) the goal $\leftarrow \neg l$ succeeds.
- (iii) The goal $\leftarrow \sim l$ succeeds if the goal $\leftarrow l$ fails.
- (iv) The goal $\leftarrow \sim l$ fails if the goal $\leftarrow l$ succeeds.

Note the sense that if the goal $\leftarrow l$ may succeed, then we may be aware of l .

Contradiction removal

Compared with the method of [1] involving the coherence principle, we have a different method of removing contradictory succeeding derivations of both the goal $\leftarrow a$ and the goal $\leftarrow \neg a$. If both goals may succeed, contradictory awareness of a and $\neg a$ is made, which should be escaped.

Now assume a propositional ELP

$$Q = \{r \leftarrow q, \neg q; q \leftarrow \sim p; p \leftarrow \sim q; \neg q \leftarrow\}.$$

For the goal $\leftarrow r$, we have the derivation by SLD resolution: A goal $\leftarrow r$ is reduced to a goal $\leftarrow q, \neg q$, as illustrated below.

$$\begin{array}{c} \leftarrow r \\ | \\ \leftarrow q, \neg q \end{array} \quad (\text{with a clause } r \leftarrow q, \neg q)$$

The goal $\leftarrow \neg q$ can succeed with the clause $\neg q \leftarrow$, where it is reduced to the goal \square as shown below.

$$\begin{array}{c} \leftarrow \neg q \\ | \\ \square \end{array} \quad (\text{with a clause } \neg q \leftarrow)$$

It follows that the goal $\leftarrow q, \neg q$ is reduced to the goal $\leftarrow q$, while the goal $\leftarrow q$ cannot succeed, for contradiction removal, after the success of the goal $\leftarrow \neg q$.

Alternatively, if we have a derivation of the goal $\leftarrow q$ to \square , whose details is omitted, the goal $\leftarrow q, \neg q$ is reduced to the goal $\leftarrow \neg q$, which is to be suspended. Here we can see that the goal $\leftarrow r$ cannot succeed. Finally the following requirements are implemented.

- (i) At most one of goals $\leftarrow q$ and $\leftarrow \neg q$ is permitted to succeed. (By means of some memory for the succeeding derivation to eliminate contradictions, the contradiction-free derivation is automated.)
- (ii) If the goal $\leftarrow q$ succeeds, then the goal $\leftarrow \neg q$ can be regarded as failing.

3 Reasoning Procedure

For reasoning to be apart from paraconsistency, we have two sets to be kept, which are to be transformed through succeeding and failing derivations:

- (i) the set of literals to remove contradictory succeeding derivations
- (ii) the set of negation-as-failure literals

The former set (i) is expressed by Σ, Σ_1, \dots , and the second one (ii) is by Δ, Δ_1, \dots . Given an ELP P , the predicate $suc_P(G; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ is derived, when a goal G succeeds with the assumed sets Σ_1 and Δ_1 , to acquire the sets Σ_2 and Δ_2 . The predicate $fail_P(G; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ is derived, when a goal G fails with the assumed sets Σ_1 and Δ_1 , to acquire the sets Σ_2 and Δ_2 . Following the derivations as above, we have the rules. Intuitively the succeeding derivation expresses some awareness, while the failing derivation detects unawareness, in relation to negation-as-failure rules.

We assume an ELP P and have the relational representations of succeeding and failing derivations, where the relations suc_P and $fail_P$ are defined simultaneously by recursion to be the least set satisfying the following closure. Because the relations demonstrate the implementation as procedural methods for the given ELP, they can be regarded as providing behaviours

such that the abstraction of the representation is more general than those in [19]. The subscript P for the ELP P may be omitted if it is clear in the context.

- (0) $suc_P(\square; \Sigma; \Delta; \Sigma; \Delta)$ for any Σ and Δ .
- (1) $suc_P(\leftarrow L_1, \dots, L_{i-1}, M_1, \dots, M_m, L_{i+1}, \dots, L_n; \Sigma_1 \cup \{l\}; \Delta_1; \Sigma_2; \Delta_2)$
for $\neg l \notin \Sigma_1$ and $(l \leftarrow M_1, \dots, M_m) \in P \Rightarrow$
 $suc_P(\leftarrow L_1, \dots, L_{i-1}, l, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.
- (2) $suc_P(\leftarrow L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ and $\sim l \in \Delta_1 \Rightarrow$
 $suc_P(\leftarrow L_1, \dots, L_{i-1}, \sim l, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.
- (3) $suc_P(\leftarrow L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_n; \Sigma'_2; \Delta'_2; \Sigma_2; \Delta_2)$ and
 $fail_P(\leftarrow l; \Sigma_1; \Delta_1 \cup \{\sim l\}; \Sigma'_2; \Delta'_2)$ for $\sim l \notin \Delta_1 \Rightarrow$
 $suc_P(\leftarrow L_1, \dots, L_{i-1}, \sim l, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.
- (4) There is no clause in P , which contains l in the head \Rightarrow
 $fail_P(\leftarrow L_1, \dots, L_{i-1}, l, L_{i+1}, \dots, L_n; \Sigma; \Delta; \Sigma; \Delta)$ for any Σ and Δ .
- (5) For any clause $l \leftarrow M_1^j, \dots, M_{n_j}^j \in P$ ($1 \leq j \leq k$) of all the clauses
which contain l in the head,
 $fail_P(\leftarrow L_1, \dots, L_{i-1}, M_1^j, \dots, M_{n_j}^j, L_{i+1}, \dots, L_n; \Sigma_j; \Delta_j; \Sigma_{j+1};$
 $\Delta_{j+1}) \Rightarrow fail_P(\leftarrow L_1, \dots, L_{i-1}, l, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_{k+1}; \Delta_{k+1})$.
- (6) $suc_P(\leftarrow \neg l; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2) \Rightarrow fail_P(\leftarrow l; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.
- (7) $fail_P(\leftarrow L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ and $\sim l \in \Delta_1 \Rightarrow$
 $fail_P(\leftarrow L_1, \dots, L_{i-1}, \sim l, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.
- (8) $suc_P(\leftarrow l; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ for $\sim l \notin \Delta_1 \Rightarrow$
 $fail_P(\leftarrow L_1, \dots, L_{i-1}, \sim l, L_{i+1}, \dots, L_n; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.

We can see the characteristics of the relational representations:

- (a) If we have the relation $suc_P(G; \emptyset; \emptyset; \Sigma; \Delta)$ for a goal G , the whole interpreter for the logic program can detect corresponding derivations. The literals of the goal G in the relation suc_P are regarded as made aware of, while if the goal G is included in the relation $fail_P$ then the literals of the goal is concerned with unawareness.
- (b) In the relation $suc_P(G; \emptyset; \emptyset; \Sigma; \Delta)$, the set Δ contains the extraction of negation-as-failure literals.

- (c) By the set Σ of the relation $suc_P(G; \emptyset; \emptyset; \Sigma; \Delta)$, we can trace the literals used for SLD resolution.

We see an illustration.

Example. Take a simple ELP $P = \{p \leftarrow, \neg p \leftarrow\}$.

- (i) Clearly we have the empty clause \square from a goal $\leftarrow p$ with a given clause $p \leftarrow$ such that

$$suc_P(\leftarrow p; \emptyset; \emptyset; \{p\}; \emptyset).$$

- (ii) Similarly we have $suc_P(\leftarrow \neg p; \emptyset; \emptyset; \{\neg p\}; \emptyset)$. However, we cannot have

$$suc_P(\leftarrow \neg p; \{p\}; \emptyset; \{\neg p\}; \emptyset),$$

after that we have got $suc_P(\leftarrow p; \emptyset; \emptyset; \{p\}; \emptyset)$, as in the case of (i).

- (iii) After we have got $suc_P(\leftarrow p; \emptyset; \emptyset; \{p\}; \emptyset)$ which is concerned with awareness of the literal p , the inference rule gives

$$fail_P(\leftarrow \neg p; \emptyset; \emptyset; \{p\}; \emptyset),$$

which detects unawareness of the literal $\neg p$.

In the following sense, the relation suc_P , which may involve the effect of the relation $fail_P$, is sound. This is paraphrased to the sense of consistency that if a goal $\leftarrow l$ succeeds, then $\sim l$ cannot be included in the set of negation-s-failure literals. It also contains an interpretation that awareness reasoned by the relation suc_P is consistent with unawareness detected by the relation $fail_P$. The proof is made. Here we see its outline.

Definition 3.1 For a set Σ , we define the set $\tilde{\Sigma}$ to be $\{\sim \neg l \mid l \in \Sigma\}$.

Theorem 3.2 Assume that $suc_P(\leftarrow l; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ such that $\Sigma_1 = \Delta_1 = \emptyset$. Then $\sim l \notin \Delta_2 \cup \tilde{\Sigma}_2$.

Proof (Outline) (1) By the definition of the relation suc_P , $\neg l \notin \Sigma_2$. It follows that $\sim l \notin \tilde{\Sigma}_2$.

(2) On the contrary to the assumption that $\sim l \notin \Delta_2$, suppose that $\sim l \in \Delta_2$. Then, by the construction of the set Δ_2 , it follows that

$$fail_P(\leftarrow l; \Sigma'_1; \Delta'_1; \Sigma'_2; \Delta'_2)$$

for some sets $\Sigma'_1, \Delta'_1, \Sigma'_2, \Delta'_2$ such that $l \in \Delta'_1 \subseteq \Delta'_2 \subseteq \Delta_2$. There are the following cases to support this relation $fail_P$.

- (i) In case that $suc_P(\leftarrow \neg l; \Sigma'_1; \Delta'_1; \Sigma'_2; \Delta'_2)$, this contradicts the first assumption that $suc_P(\leftarrow l; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$.
- (ii) In case that there is no clause whose head is l for the relation $fail_P$, this contradicts the first assumption that $suc_P(\leftarrow l; \Sigma_1; \Delta_1; \Sigma_2; \Delta_2)$ such that there is some clause whose head is l .
- (iii) In case that $fail_P(\leftarrow \sim m_1, \dots, \sim m_n; \Sigma''_1; \Delta''_1; \Sigma''_2; \Delta''_2)$ for some sets $\Sigma''_1, \Delta''_1, \Sigma''_2, \Delta''_2$, which may be caused for some negative goal

$$\leftarrow \sim m_1, \dots, \sim m_n$$

derivable from the goal $\leftarrow l$, it is concluded that $n \neq 0$. Otherwise, the negative goal $\leftarrow \sim m_1, \dots, \sim m_n$ is \square and it contradicts the relation $fail_P$. For the negative goal

$$\leftarrow \sim m_1, \dots, \sim m_n,$$

there is some literal $m_i \notin \Delta''_1$ ($1 \leq i \leq n$) such that $suc_P(\leftarrow m_i; \Sigma''_1; \Delta''_1; \Sigma''_2; \Delta''_2)$, because of the relation $fail_P$. On the assumption that $\sim m_i \in \Delta_2$, we repeat the same discussion. We finally reach the case that:

$$fail_P(\leftarrow \sim m_1^f, \dots, \sim m_n^f; \Sigma_1^f; \Delta_1^f; \Sigma_2^f; \Delta_2^f)$$

for some sets $\Sigma_1^f, \Delta_1^f, \Sigma_2^f, \Delta_2^f$, but there is no literal $\sim m_i^f$ such that $\sim m_i^f \notin \Delta_1^f$. This causes the case that $\leftarrow \sim m_1^f, \dots, \sim m_n^f = \square$, which contradicts that $fail_P(\leftarrow \sim m_1^f, \dots, \sim m_n^f; \Sigma_1^f; \Delta_1^f; \Sigma_2^f; \Delta_2^f)$. This concludes the proof.

Q.E.D.

4 Literal-Constrained Term

We here have the form: a constrained literal followed by a (function) term, where

- (a) the literal is derivable from some proof system like the logic programming system such that the literal may be interpreted as awareness (of an agent), and
- (b) the function term is held (by an agent) under the awareness of the literal,

such that the form may be regarded as denoting a behaviour of an agent.

Syntax

An extended term from the original is shown below, where a logic-constraint may be made by a literal. If we prefer to the ELP, which derives literals, then the derivable literals may be constraints.

Definition 4.1 On the assumption of a proof system (say, Γ), a literal-constrained term (*term*, for short) is recursively defined as follows:

- (i) If x is a variable, then x is a term.
- (ii) If M, N are terms, then $(M N)$ is a term.
- (iii) If x is a variable and M a term, then $(\lambda x.M)$ is a term.
- (iv) If p is a literal and M a term, then $p < M >$ is a term.

Semantics

By the term $p < M >$ with some system Γ (which may possibly be the ELP in the previous section), we mean that:

- If p is derivable from Γ , then the term M is supported.
- Unless p is derivable from Γ , then the term M is not supported.

When the ELP may be taken as a proof system Γ , the contradiction removal procedure is significant, because $p < M >$ and $\neg p < M >$ cannot be coherent.

Illustration

Assume the following functional program.

```
Even(x) = if x = 0 then true  
          else if x = 1 then false  
          else Odd(x - 1)
```

```
Odd(x) = if x = 0 then false
```

else if $x = 1$ **then** *true*
else $Even(x - 1)$

As a standard way, by means of a *fixed point operator*:

Let $Y = \lambda f.(\lambda x.f(x x))(\lambda x.f(x x))$,

Let $evenfn = \lambda g.\lambda n.(if\ n = 0\ then\ true$
 $else\ if\ n = 1\ then\ false$
 $else\ g\ (-\ n\ 1))$

Let $oddfn = \lambda f.\lambda m.(if\ m = 0\ then\ false$
 $else\ if\ m = 1\ then\ true$
 $else\ f\ (-\ m\ 1))$

Let $even = Y\ (evenfn\ oddfn)$

Let $odd = Y\ (oddfn\ evenfn)$

The terms “*evenfn*” and “*oddfn*” may be included in the expressions such as:

$p < evenfn >$ and $q < oddfn >$,

where the literals may denote awareness, which acknowledges the application of $p < evenfn >$ to $q < oddfn >$ for the usual term $(evenfn\ oddfn)$ to *even*, and vice versa for the term $(oddfn\ evenfn)$ to *odd*.

Condition and conversion

By the expression $p \longrightarrow q$, we mean that if p is derivable from Γ , then q is derivable from Γ .

In addition to the standard α , β , and η conversions, the following two conversions are to be presented:

$$(\gamma 1) \frac{(p < M > \ q < N >)}{q < (M\ N) >} \ p \longrightarrow q$$

$$(\gamma 2) \frac{(\lambda x.p < M >)}{p < \lambda x.M >}$$

Assume the program.

$f(x) = \mathbf{if } x = 0 \mathbf{ then } 1 \mathbf{ else } x \times f(x - 1)$

Let $factorial = Y factorialfn$,

Let $factorialfn = \lambda f. \lambda x. (((iszero x) 1) (times x (f (- x 1))))$,

where the λ -term $((B X_1) X_2)$ can denote the conditional sentence **if** B **then** X_1 **else** X_2 . If we have the fixed point operator $p < Y >$, the if-part $q < (iszero 0) 1 >$, and the else-part

$r < Times x (f (- x 1)) >$

such that $p \rightarrow q$ and $q \rightarrow r$, then

$r < Y factorialfn > = r < factorial >$.

The relation \Rightarrow^* denotes a reflexive and transitive closure based on α , β , η , $\gamma 1$ and $\gamma 2$ conversions.

Church-Rosser theorem

Because

- (a) the calculus for terms constructed by using only (i), (ii) and (iii) conceives the *Church-Rosser Theorem*, and
- (b) α , β and η conversions are commutative with $\gamma 1$ and $\gamma 2$ conversion applications,

we may see that the term constructed by Definition 4.1 is transformed to a *normal form* (which any conversion except α cannot be applied to) uniquely up to α conversion, with respect to the relation \Rightarrow^* , if the term has one. It is stated as:

Theorem 4.2 *If the term M has a normal form, it is unique up to α conversion.*

Proof (Outline) Applications of $\gamma 1$ and $\gamma 2$ may be sound with respect to the transformation to the normal form by the following senses:

Assume two terms M and N whose normal forms are M_{normal} and N_{normal} , respectively. For any terms M_1 and N_1 such that

$$\begin{aligned} M &\Rightarrow^* M_1 \Rightarrow^* M_{normal}, \text{ and} \\ N &\Rightarrow^* N_1 \Rightarrow^* N_{normal}, \end{aligned}$$

we have:

$$\frac{\frac{p < M > \Rightarrow^* p < M_1 > \quad q < N > \Rightarrow^* q < N_1 >}{(p < M_1 > \quad q < N_1 >) \quad p \longrightarrow q}}{q < (M_1 \ N_1) >}}{q < M_{normal} \ N_{normal} >}$$

with respect to $\gamma 1$ conversion, and

$$\frac{\lambda x.p < M > \quad p < M > \Rightarrow^* p < M_1 >}{\frac{\lambda x.p < M_1 >}{p < \lambda x.M_1 >}}}{p < \lambda x.M_{normal} >}$$

with respect to $\gamma 2$ conversion such that we intuitively see the induction for the proof. Q.E.D.

If we adopt the ELP P as a formal system (Γ as above), whether

$$p \longrightarrow q$$

can be determined by reasoning that on condition that we have $suc_P(\leftarrow p; \emptyset; \emptyset; \Sigma; \Delta)$, we then have $suc_P(\leftarrow q; \Sigma; \Delta; \Sigma'; \Delta')$ for $\Delta \subseteq \Delta'$ and $\Sigma \subseteq \Sigma'$.

5 Concluding Remarks

The problem of treatments for the logic-constrained function is related to the backgrounds: (i) Logic and database views are fundamental to analyze knowledge structure ([14, 17]), to understand dynamic structure with reference to knowledge ([15, 16]). (ii) Process algebra deals with sequence structure of communications (evaluations) ([10, 13]) even for distributed systems ([5]). (iii) The logic programming system (in computational logic) contains the notion of negatives such that both classical negation (in the literal regarding awareness) and negation as failure (regarding unawareness) are combined for more powerful representations ([1]).

For a literal-constrained term, the literal should be derivable from an indicated proof system. If derivability is concerned with awareness, the term (as a function application) is regarded as originating from awareness with constraint. In this paper, the logic programming system with a contradiction removal procedure is presented, not allowed to be paraconsistent.

As regards awareness in terms of derivability, we can also have the formal systems: hybrid logic (with modality and nomination) (as in [3]), and action logic (as in [8]).

With reference to actions like those of [15, 16], analyses of whether or not we can apply them to the literal-constrained term are needed.

As the proof system itself, we summarize some points on specific expressiveness of the ELP. (1) This paper presents an abstract representation of reasoning for its application to the reasoning of contradiction removal regarding derivability vs. awareness, independent of abduction reasoning as in [4, 6]. (2) The backgrounds of soundness of succeeding and failing derivations may be closely related to model theory, following [1, 19]. (3) The notion of exceptions for each literal to be constrained by is implementable in the derivations we present, while we may apply *weak negation* to it as in [20]. (4) There is a problem of whether or not a non-grounded version of the literals (for awareness constraints) may be built in the proof system to derive literals for constraints. A non-grounded version of negation as failure is relevant to the discussions as in [17, 18].

References

- [1] Alferes, J.J., Damásio, C.V. and Pereira, L.M., A logic programming system for nonmonotonic reasoning, *J. of Automated Reasoning*, 14, pp.93-147, 1995.
- [2] Besnard, P., *An Introduction to Default Logic*, Springer-Verlag, 1989.
- [3] Brauner, T., Natural deduction for hybrid logic, *JLC*, 14, 3, pp.329-353, 2004.
- [4] Brogi, A., Lamma, P., Mancarella, P. and Mello, P., A unifying view for logic programming with non-monotonic reasoning, *Theoretical Computer Science*, 184, 1-2, pp.1-59, 1997.
- [5] Bruns, G., *Distributed Systems Analysis with CCS*, Prentice-Hall, 1996.
- [6] Dung, P.M., An argumentation-theoretic foundation for logic programming, *J. of Logic Programming*, 22, pp.151-177, 1995.
- [7] Gelfond, M. and Lifschitz, V., The stable model semantics for logic programs, *Proc. of 5th ICLP*, pp.1070-1080, 1988.

- [8] Giordano,L., Martelli,A. and Schwind,C., Ramification and causality in a modal action logic, *JLC*, 10, 5, pp.625–662, 2000.
- [9] Gordon,M.J.C., *Programming Language Theory and its Implementation*, Prentice Hall, 1988.
- [10] Hoare,C.A.R., *Communicating Sequential Processes*, Prentice-Hall, 1985.
- [11] Kucera,A. and Esparza,J., A logical viewpoint on process-algebra, *J. of Logic and Computation*, 13, 6, pp.863–880, 2003.
- [12] Lloyd,J.W., *Foundations of Logic Programming*, 2nd, Extended Edition, Springer-Verlag, 1993.
- [13] Milner,R., *Communication and Concurrency*, Prentice-Hall, 1989.
- [14] Minker,J. (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers, Inc., 1987.
- [15] Mosses,P.M., *Action Semantics*, Cambridge University, 1992.
- [16] Reiter,R., *Knowledge in Action*, The MIT Press, 2001.
- [17] Shepherdson,J.C., Negation in Logic Programming, in: Minker,J. (ed.), *Foundations of Deductive Databases and Logic Programming*, pp.19-88, 1987.
- [18] Yamasaki,S. and Kurose,Y., Soundness of abductive proof procedure with respect to constraint for non-ground abducibles, *Theoretical Computer Science*, 206, pp.257-281, 1998.
- [19] Yamasaki,S. and Kurose,Y., A sound and complete procedure for a general logic program in non-floundering derivations with respect to the 3-valued stable model semantics, *Theoretical Computer Science*, 266, pp.489–512, 2001.
- [20] Yamasaki,S., Logic programming with default, weak and strict negations, *Theory and Practice of Logic Programming*, 6, pp.737-749, 2006.

The 4rd International Workshop on Multi-Agent Systems and Simulation (MAS&S): Engineering Complex Systems through Agent-Based Modeling and Simulation, MALLOW-MAS&S'10

(Introductory Essay of the Workshop)

Carole Bernon*, Alfredo Garro[†], and Jorge J. Gomez-Sanz[‡]

*IRIT, Université Paul Sabatier (France)

118, Route de Narbonne, 31062 TOULOUSE Cedex 09, France

Email: carole.bernon@irit.fr

[†]Department of Electronics, Informatics and Systems (DEIS)

Università della Calabria

Via P. Bucci cubo 41C, 87036 Arcavacata di Rende (CS), Italy

Email: alfredo.garro@unical.it

[‡]Dep. Ingeniería del Software e Inteligencia Artificial

Universidad Complutense Madrid (Spain)

Ciudad Universitaria s/n, 28040 Madrid, Spain

Email: jjgomez@fdi.ucm.es

Abstract

Multi-agent systems (MASs) provide powerful models for representing both real-world systems and applications with an appropriate degree of complexity and dynamics. Several research and industrial experiences have already shown that the use of MASs offers advantages in a wide range of application domains (e.g. financial, economic, social, logistic, chemical, engineering). When MASs represent software applications to be effectively delivered, they need to be validated and evaluated before their deployment and execution, thus methodologies that support validation and evaluation through simulation of the MAS under development are highly required. In other emerging areas (e.g. ACE, ACF), MASs are designed for representing systems at different levels of complexity through the use of autonomous, goal-driven and interacting entities organized into societies which exhibit emergent properties. The agent-based model of a system can then be executed to simulate the behavior of the complete system so that knowledge of the behaviors of the entities (micro-level) produce an understanding of the overall outcome at the system-level (macro-level). In both cases (MASs as software applications and MASs as models for the analysis of complex systems), simulation plays a crucial role which needs to be further investigated.

I. INTRODUCTION

This is the fourth edition of MAS&S. The first edition was jointly held with EUROSIS ISC 2006 (Industrial Simulation Conference), June 5-7, 2006, Palermo, Italy [1]. The second edition happened in EUROSIS ESM 2007 (European Simulation and Modelling Conference), October 22-24, 2007, St. Julian's, Malta [2]. The third edition took place as part of MALLOW, the second edition of Multi-Agent Logics, Languages, and Organisations (Federated Workshops), 7-11 September Torino, Italy.

The best papers of the first edition have also been selected and their extended and revised version published in International Journal of Agent Oriented Software Engineering [3]. Similarly, best papers from second to third editions were selected and extended for a special issue of the Simulation Modelling Practice and Theory Journal, which is in press.

MAS&S was conceived for stimulating discussion among researchers and practitioners working on ABS and AOSE, to enable the identification and the definition of methodologies and techniques for integrating them.

Carole Bernon
Alfredo Garro
Jorge J. Gomez-Sanz
August 3, 2010

II. WORKSHOP COMMITTEES

A. *Workshop Organizers*

Carole Bernon	IRIT - Université Paul Sabatier, France
Alfredo Garro	Università della Calabria, Italy
Jorge J. Gomez-Sanz	Universidad Complutense de Madrid, Spain

B. *Programme Committee*

Jean-Paul Arcangeli	Université Paul Sabatier, France
Juan Antonio Botía Blaya	Universidad de Murcia, Spain
Paul Davidson	Blekinge Institute of Technology, Sweden
Paolo Giorgini	Università di Trento, Italy
Samer Hassan	Universidad Complutense de Madrid, Spain
Vincent Hilaire	Université de Belfort-Montbéliard, France
Franziska Klügl	Örebro Universitet, Sweden
Adolfo López-Paredes	University of Valladolid, Spain
Muaz Niazi	Foundation University, Pakistan
Michael J. North	Argonne National Laboratory, USA
Andrea Omicini	Università di Bologna, Italy
Paolo Petta	OFAI, Austria
Gauthier Picard	ENSM, Saint-Etienne, France
Sébastien Picault	LIFL, Lille, France
Luca Sabatucci	ITC-irst, FBK, Italy
Valeria Seidita	Università degli Studi di Palermo, Italy
Pietro Terna	Università di Torino, Italy
Erwan Tranvouez	LSIS, France
Giuseppe Vizzari	Università di Milano Bicocca, Italy

C. *Steering Committee*

Massimo Cossentino	ICAR/CNR, Italy
Giancarlo Fortino	University of Calabria, Italy
Juan Pavón	Universidad Complutense Madrid, Spain
Marie-Pierre Gleizes	IRIT - Université Paul Sabatier, France
Wilma Russo	University of Calabria, Italy

III. LIST OF PAPERS

- The Impact of Market Preferences on the Evolution of Market Price and Product Quality by *Hongliang Liu, Enda Howley, and Jim Duggan*

- Learning Virtual Agents for Decision-Making in Business Simulators
by *Javier Garcia, Fernando Borrajo, and Fernando Fernandez*
- Looking for the Self-fulfilling Prophecy Effect in a Double Auction Artificial Stock Market
by *Albert Meco, Javier Arroyo, Juan Pavón, and Javier Pajares*
- BDI Agents with Fuzzy Perception for Simulating Decision Making in Environments with Imperfect Information
by *Giovani Farias, Graçaliz Dimuro, and Antonio Carlos Costa*
- When Will I See you Again: Modelling the Influence of Social Networks on Social Activities
by *Nicole Ronald, Virginia Dignum, and Catholijn Jonker*
- Human Behaviours Simulation in Ubiquitous Computing Environments
by *Teresa García-Valverde, Francisco Campuzano, Emilio Serrano, and Juan A. Botía*
- A Survey on Coordination Methodologies for Simulated Robotic Soccer Teams
by *Fernando Almeida, Nuno Lau, and Luis Paulo Reis*
- ELDAMeth: A Methodology for Simulation-based Prototyping of Distributed Agent Systems
by *Giancarlo Fortino and Wilma Russo*
- Design and Simulation of a Wave-like Self-Organization Strategy for Resource-Flow Systems
by *Jan Sudeikat, Jan-Philipp Steghöfer, Hella Seebach, Wolfgang Reif, Wolfgang Renz, Thomas Preisler, and Peter Salchow*
- Generating Inspiration for Multi-Agent Simulation Design by Q-Learning
by *Robert Junges and Franziska Klügl*

IV. SPONSORING INSTITUTIONS

Alfredo Garro has partially been funded by the Department of Electronics, Informatics and Systems (DEIS), University of Calabria, Italy.

Jorge J. Gomez-Sanz has partially been funded by the the project Agent-based Modelling and Simulation of Complex Social Systems (SiCoSSys), supported by Spanish Council for Science and Innovation, with grant TIN2008-06464-C03-01, and by the Programa de Creació n y Consolidació n de Grupos de Investigació n UCM-Banco Santander for the group number 921354 (GRASIA group).

V. ACKNOWLEDGMENT

The workshop organizers would like to thank the MALLOW organizers (Olivier Boissier, Amal El Fallah Seghrouchni, Salima Hassas, and Nicolas Maudet,) for their endless support in making possible the organization of the 4rd edition of MAS&S in MALLOW and the chair sponsorships for financial funding.

REFERENCES

- [1] Alessandro Genco, Antonio Gentile and Salvatore Sorce, editors. *Proceedings of the Industrial Simulation Conference (ISC 2006)*, June 5-7, 2006, University of Palermo, Palermo, Italy, 535 pages, ISBN 90-77381-26-0.
- [2] Jaroslav Sklenar, Cyrille Bertelle and Giancarlo Fortino. *Proceedings of the European Simulation and Modeling Conference (ESM 2007)*, October 22-24, 2007, University of Malta, St Julians, Malta, 615 pages, ISBN 978-90-77381-36-6.
- [3] Massimo Cossentino, Professor Giancarlo Fortino and Professor Wilma Russo. *Special Issue on Multi-Agent Systems and Simulation*, International Journal of Agent-Oriented Software Engineering (IJAOSSE), 2(2), 2008, ISSN 1746-1383.

The impact of market preferences on the evolution of market price and product quality

Hongliang Liu, Enda Howley and Jim Duggan

Abstract—A significant challenge for firms in an open-competition marketplace is to balance the conflicting attributes of price and quality. Higher quality levels tend to lead to increased product costs, which, depending on market preferences, can trigger an increase in consumer demand. This paper presents a multi-agent model that allows for an exploration of how price and quality evolve as a result of direct market competition between firms. A new competition model, based on price and quality, is defined. Agents compete by determining their price and quality levels with a view to maximizing their profit. Our goal is to examine a range of market configurations and study how agent strategies evolve over time. We focus on those factors which contribute to each agent's survival in this evolutionary setting. We use game theoretic simulation as a basis to examine various agent strategies. A genetic algorithm is used to characterize a changing environment which evolves over time to reflect the emergence of fitter strategy attributes. Individuals can evolve their own market preferences over subsequent generations and adapt to their preferred market strategy. Agent strategies evolve rapidly to reflect the bias of their individual market. The price and quality relationship of a given market is a primary driver of the evolution of agent strategies in that market. Significantly, our results show the emergence of strategies that prefer low price and high quality sensitive markets. This is despite the penalties which are incurred by the higher costs of increased quality. These results have potentially interesting applications to real-world market dynamics, particularly as companies strive to position their products optimally on different markets.

Index Terms—Price and Quality Competition, Agent Computational Economics, Agent-based Simulation, Genetic Algorithm

I. I

Consumers from different markets exhibit wide preference differences due to natural variation in tastes and income disparities. These are mainly reflected by consumers' accepted price and quality levels. For example, consumers in rural areas may prefer lower price products while consumers in urban areas maybe willing to pay higher prices for higher quality products. These consumer preferences can indirectly establish trends in production. From the view of the firms, higher quality usually requires the use of more expensive components, and less standardized production process, and so on. As a result, higher quality levels tend to lead to increased product costs. Nevertheless, higher quality, depending on market preferences, can trigger an increase in consumer demand, and probably gain market share [1]. Therefore, there are trade-offs between quality and cost for firms. In terms of price, it is also a decision

challenge. Firms can charge a higher price for their product in order to get a higher unit profit. However, higher price levels usually lead to a reduction in customer demand. Therefore, ensuring a good balance between the conflicting attributes of price and quality is a significant challenge for firms in an open-competition marketplace.

In order to better understand this problem, a number of game theoretic models have been proposed [2] [3] [4]. This existing research has focused on the strategic or rational behavior of competition between two firms. However, what will happen when there are more than two firms and their decisions are affected by the effect of bounded rationality? Another common feature of the current research is that researchers limit their analysis on one market in these models. However, in the real world, firms usually compete with each other in different marketplaces. In order to address this issue, we propose a new multi-agent competition model, based on price and quality. In this model, we consider many firm agents competing in a number of markets. Markets are defined by their own unique properties. Price and quality sensitivities are used to represent these properties, and reflect a consumers' preferred product. Variations in these values effect the demand of the products in the market. Different markets may have different price and quality sensitivities. Each market demand is determined by the average price and quality levels of firm agents in that market. Thus, each firm agent faces decision challenges including their product price and quality levels, and their preferred markets. Furthermore, the effectiveness of one agent's strategy depends on the strategies of others. In this paper, we model firm agents as individuals in a genetic algorithm which has been widely used as learning mechanism for economic agents [5] [6]. The genetic algorithm is also used to characterize a competitive market environment where the agents compete with each other for the market share. The firm agents can make price and quality decisions and evolve their own market preferences. However, they have a limited knowledge of their environment and their performance is largely determined by the actions of their peers. These features of our model are significantly different with models in the existing research.

In this paper, we aim to examine a range of market configurations and study how firm agent strategies evolve over time. We investigate how firm agents strategically position their products over time and what are the impacts of alternative market preferences on the evolution. We have conducted a series of experiments on a range of market configurations. Our results show the impacts of market preferences on the evolution of market price and product quality. The firm agent strategies evolve rapidly to reflect the bias of their individual

Hongliang Liu, Enda Howley and Jim Duggan are with the Department of Information Technology, National University of Ireland, Galway (email: h.liu1@nuigalway.ie, enda.howley@nuigalway.ie and jim.duggan@nuigalway.ie)

market. The price and quality relationship of a given market is a primary driver of the evolution of agent strategies in that market. Significantly, our results show the emergence of strategies that prefer markets which have low price and also high quality sensitive markets. This is despite the penalties which are incurred by the higher costs of increased quality. These results have potentially interesting applications to real-world market dynamics, particularly as companies strive to position their products optimally on many markets.

The sections of this paper are structured as follows. In Section II, we will review much of the related work relevant to price and quality competition. In Section III, we will outline our model design. Section IV will provide a detailed examination of our experimental results. Finally, in Section V we will outline our conclusions and some future work.

II. B R

The study of price and quality competition has attracted many researchers' attention. There are two main streams in the current research. One is a formal study of rational behaviors among strategically interacting agents using game theory. While the alternative approach is to use agent-based modeling and simulation to examine market economies. This is also known as agent-based computational economics (ABCE) which is the computational study of economics modeled as evolving systems of autonomous interacting agents [6].

A. Game Theory Models

Since the seminal work of Hotelling [2], a rich and diverse literature on price and quality competition has emerged. Harold Hotelling analyzes a model of spatial competition which demonstrates the relationship between location and pricing behavior of firms. In this model, Hotelling assumes that potential consumers are evenly distributed in a linear geographic location such as a straight street. Consumers have no preferences to the firms and only buy products from these that provides better value in terms of price and transportation cost. Both firms have the same constant marginal costs and compete on the store location and price. From this spatial competition model, Hotelling argues that the equilibrium strategy for each firm is to choose a location at the center of the market which is commonly referred to as "Principle of Minimum Differentiation" or "Hotelling's law". This argument means that for any location of one firm, the other firm has an incentive to move toward its opponent in order to expand the territory under its exclusive control. In this model, a customer's location can also be interpreted as a customer's preference for quality, therefore, many papers on price and quality competition are inspired by this work. For example, Moorthy considers the quality choice in a duopoly, assuming the existence of a quadratic cost function for quality which is different with the Hotelling's location model [7]. Banker et al. examine a price and quality competition also under a duopoly setting, where consumers' demand is a linear function of price and quality levels and the cost of quality is also a quadratic form [3]. Moorthy and Banker et al. analyze the impact of quality on competitive advantages. Vörös designs a price and quality

model using decreasing and increasing exponential demand functions for price and quality, respectively, and analyzes the influences of the quality inflating which means that the same quality performance is worth less tomorrow than today [8]. Recently, Matsubayashi et al. explore the impact of different customers' loyalty to each firm on the outcome of price and quality competition [4].

B. Agent based simulations

Agent based simulations have been successfully applied many problems such as telecommunications and market strategies [9]. In many economic applications, genetic algorithms (GAs) have been widely used to represent the learning processes of agents [10] [11]. GAs were developed by Holland in 1975 as a way of studying adaptation, optimization and learning [12]. GAs are inspired by evolutionary biology such as selection, crossover (also called recombination) and mutation. A basic GA manipulates a population of chromosomes that encode candidate solutions to a problem. Each chromosome or individual in a GA is assigned a measure of performance, called its fitness. In a game context, a chromosome can be interpreted as a strategy, and the GAs processes are models of learning. In GAs, the reproduction operator can be interpreted as learning by imitation, the crossover operator can be interpreted as learning through communication, and the mutation operator is interpreted as learning by experiment [13].

GAs have been used to examine some well known game theory models such as Prisoner's Dilemma [10], Cournot competition and Bertrand competition[14]. However, almost all the existing research has employed classical game theory to examine the price and quality competition as we have examined earlier. Only recently, Tay et al. have used a genetic algorithm to test Hunt's General Theory of competition [15]. They consider an oligopolistic market with a number of sellers who are competing on price and a product attribute which reflects a consumer's ideal preferences. The sellers' demand function is a linear function of price and the product attribute which differs from our demand function for markets. Furthermore, we are interested in different research topics. They aim to use a GA as an alternative simulation method to test a competition theory. Our purpose of this paper is to investigate the impact of market preferences on the evolution of agents' strategies.

In summary, there is a body of literature in economics on price and quality competition. However, these models rely on very strong assumptions such as rational behaviour of two firms and one market. The research from ABCE has not been addressed this perspective on price and quantity competition. In this paper, we propose a multi-agent model and aim to address these issues.

III. P Q C M

In this section, we propose our game theoretic model. We consider many firm agents competing with each other over a number of competitive markets. Different markets may have different preferences over price and quality which are reflected through market demands in the markets. Firms in the same

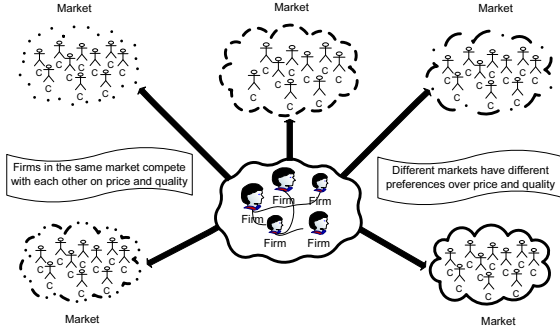


Fig. 1. Price and Quality Competition Model

marketplace compete with each other on price and quality for higher profits. As for firms, a relative lower price level or a higher quality level may attract more consumers. This depends not only on other firm agents' strategies but also on the preferences of the markets. Furthermore, the lower price or higher quality strategies also reduce unit profit level as higher quality levels incur higher unit cost levels. Therefore, in our model, each firm agent faces decision challenges including price levels, quality levels and their preferred markets as shown in Figure 1. In the following, we first present our market properties, then the firm agents and their decision-making process. Finally, our simulator design is outlined.

A. Market Properties

We consider m markets. Each market demand is dependent on the average price and quality levels (p, q) of all firm agents in the market. The market demand will increase as the price level goes down given any quality level, and on the contrary, it increases as the quality of the product improves for any price level. In order to reflect these relationships in real markets, we use Equation (1) to model market k 's demand $D_k(p, q)$.

$$D_k(p, q) = \mathcal{A}_k e^{-\alpha_k p} (1 - b e^{-\beta_k q}) \quad (1)$$

where \mathcal{A}_k is the potential maximum demand, $b \in (0, 1]$, $\alpha_k \in [0, 1]$, and $\beta_k \in [0, 1]$ are parameters. Note that the demand function is monotonically decreasing over price p and increasing over quality q since $\partial D(p, q)/\partial p < 0$ and $\partial D(p, q)/\partial q > 0$. The combination of the parameters (α, β) corresponds to a set of consumers' price and quality sensitivities for a given market.

- (α) This represents the consumers' price sensitivity as the higher α the demand goes down faster given the same price change. The higher α means higher consumers' price sensitivity.
- (β) This represents the consumers' quality sensitivity as the higher β the demand changes faster given the same quality change. Similarly, the higher β value reflects higher consumers' quality sensitivity.

B. Firm Agents

In these m markets there are f firm agents in total. Each firm agent faces decision challenges including their product price and quality levels, and their preferred markets. Let $\eta_i = (p_{i,t}, q_{i,t}, k_{i,t})$ denote the agent i 's decision strategies at time step t where $p_{i,t}, q_{i,t}, k_{i,t}$ are the price level, the quality level and the market $k_{i,t}$ ($k_{i,t} \in [1, m]$). The firm agents from the same marketplace k compete with each other for a higher market share and profit over time.

The firm agent i 's market share $(s_{i,k,t})$ in market k at time step t depends not only on its own price and quality levels but also on the other agents' strategies. We propose a new mechanism as follows.

$$s_{i,t} = \frac{D_k(p_{i,t}, q_{i,t})}{\sum_{j=1}^w D_k(p_{j,t}, q_{j,t})} \quad (2)$$

where w is the number of firm agents in the market k at time t . This mechanism is different with the mechanisms used in the existing price and quality competition models [7] [16] [3] [4]. In the existing models, researchers only consider two firms competing with each other and one firm's demand is a linear function of both firms strategies.

The firm agents from the same market compete in determining their price and quality levels to maximise their profits. The profit $(\pi_{i,t})$ for agent i in market k at time step t is given as follows:

$$\pi_{i,t} = (p_{i,t} - C(q_i)) s_{i,t} D_k(p, q) \quad (3)$$

where $p = \sum_{i=1}^w p_{i,t}$, $q = \sum_{i=1}^w q_{i,t}$, $D_k(p, q)$ is the demand of market k , $s_{i,t} D_k(p, q)$ is the firm agent i 's demand, and $C(q_i)$ the agent i 's quality cost.

Higher quality levels are usually accompanied by higher costs in most businesses. In our model, we use a quadratic cost function: $C(q) = \epsilon q^2$. The ϵ is a positive parameter. This type of cost function reflects the nonlinear impact of quality levels on costs and is often used in the marketing literature [7] [3] [4].

1) *Decision-making process:* From the discussion above, we note that the firm agents face decision challenges on their product price and quality levels, and their preferred markets $(p_{i,t}, q_{i,t}, k_{i,t})$. In this paper, the GA is not only used to characterize a competitive market environment, where the firm agents interact and compete with each other over time, but also model firm agents' decision-making process. In our GA, each firm agent is represented through an agent chromosome. This chromosome holds a number of genes which represents how that particular agent behaves.

$$\text{Chromosome} = (G_P, G_Q, G_M) \quad (4)$$

The G_P gene represents the agent's price decision strategy. The G_Q gene represents the agent's quality decision strategy. Finally, the G_M gene represents the preferred market's ID and is used to determine which market the agent participates in.

Furthermore, we use the profit function as the fitness function in our GA (See Equation 3). We do not distinguish between profit and fitness and will alternatively use both words in the following context.

TABLE I
P

Variable	Range/value	Description	Variable	Range/value	Description
T	200	Simulation length	β	[0, 1]	market preference over quality
f	60	Firm agent number		0.05	Selection rate (GA)
m	5	Market number		0.8	Crossover rate (GA)
\mathcal{A}_k	6000	Potential maximum demand		0.05	Mutation rate (GA)
b	0.9	Weight parameter	G_P	[0, 5]	Price gene
ϵ	1.0	Quality cost parameter	G_Q	[0, 1]	Quality gene
α	[0, 1]	market preference over price	G_M	{1, 2, 3, 4, 5}	Market ID gene

In our GA, we use an elitism mechanism to implement our selection operator. We select the best agents directly into the following generation which is controlled by the selection rate. This means, in each generation, a small number of agents do not change their strategies as their current strategies perform well. The rest of individuals or firm agents, have a certain probability to learn new strategies through our crossover operator and mutation operator. A single point crossover operator is implemented. For our mutation operator, the degree of change of each strategy gene is $0.1 * (max - min)$ where max and min is a gene's range.

C. Simulator Design

In order to examine the impact of market preferences on the evolution of market price and product quality, the GA is used to facilitate evolution and a competitive dynamic market environment. Our competitive market consists a number of markets and many firm agents interacting with each other. We assume that the firm agents can freely participant in any market, however, one firm agent can only participant in one market at each period. The firm agents in the same market compete with each other. In other words, firm agents compete locally in a market of their peers, where they have no knowledge about their peers, or the individual market preferences.

Initially these firm agent genes are generated using a uniform distribution for the first generation. Over subsequent generations new agent chromosomes are generated using our genetic algorithm. For each generation, we firstly calculate each market's demand, and then each firm agents market share, and profit (fitness) according to Equation 1, 2 and 3. Finally, the selection operator, crossover operator and mutation operator are applied. Through these operators, a number of the least fit individuals are removed and replaced with other new strategies which may perform better or worse than those replaced.

IV. EXPERIMENTAL RESULTS

In this section, we will present a series of experimental results from our simulations. Table I shows the parameter settings for the markets, firm agents and our GA. By varying the different parameters in our model we investigate the impact of market preferences on the evolution of market price and product quality. We examine two different market configurations: homogeneous and heterogeneous market settings. In the homogeneous model, all markets have the same price and quality sensitivities while in the heterogeneous model, the

markets have different price and quality sensitivities. In the following sections, we will firstly examine the results from homogeneous markets and then the results from heterogeneous markets.

A. Competition in homogeneous markets

All 5 markets have the same setting in homogeneous markets. Each market has two parameters α and β which reflect the market preferences. A high α reflects that a market is highly sensitive to price, while a high β reflects that a market places a premium on quality. The results in Figure 2 are from 50 runs of our simulator for each combination of α and β . Figures 2(a), 2(b), 2(c), and 2(d) depict the average price, quality, profit and demand quantity for the whole agent population at generation 200, respectively.

There are a number of features involving these experiments. Firstly, we observe that the agents' average price evolves to a lower level as the α value increases. In other words, the agents lower their price levels as the market becomes more sensitive to the price levels. Secondly, the agents' average quality level evolves to a higher level as the β value increases. This reflects that the agents increase the product quality levels as the markets pay more attention to quality. Therefore, we can conclude that agent strategies evolve to reflect the bias of their market. These emergent phenomena stem from firm agents' competition provided by our GA. As the markets are more sensitive to price or quality, the firm agents with lower price and higher quality products have a competitive advantage. These firms are considered the most fit agents in our GA. The lower price and higher quality genes are then promoted in the following generations. Finally, the market price and quality evolve to a lower level and a higher level respectively. Furthermore, these strategies subsequently affect the average profit as shown in Figure 2(c). Specifically, the average profit decreases as the markets are more sensitive to price. Higher market price sensitivities lead to intense competition, resulting in a decrease in profits. Conversely, we observe that the average profit increases as the markets are more sensitive to quality despite the higher costs of increased quality for firms. This is because that higher quality levels of products in the markets result in a higher market demand as shown in Figure 2(d), and subsequently an increased profit. Therefore, higher quality has a positive impact on agents' profit in our model. This feature of our model is consistent with the existing research results [1].

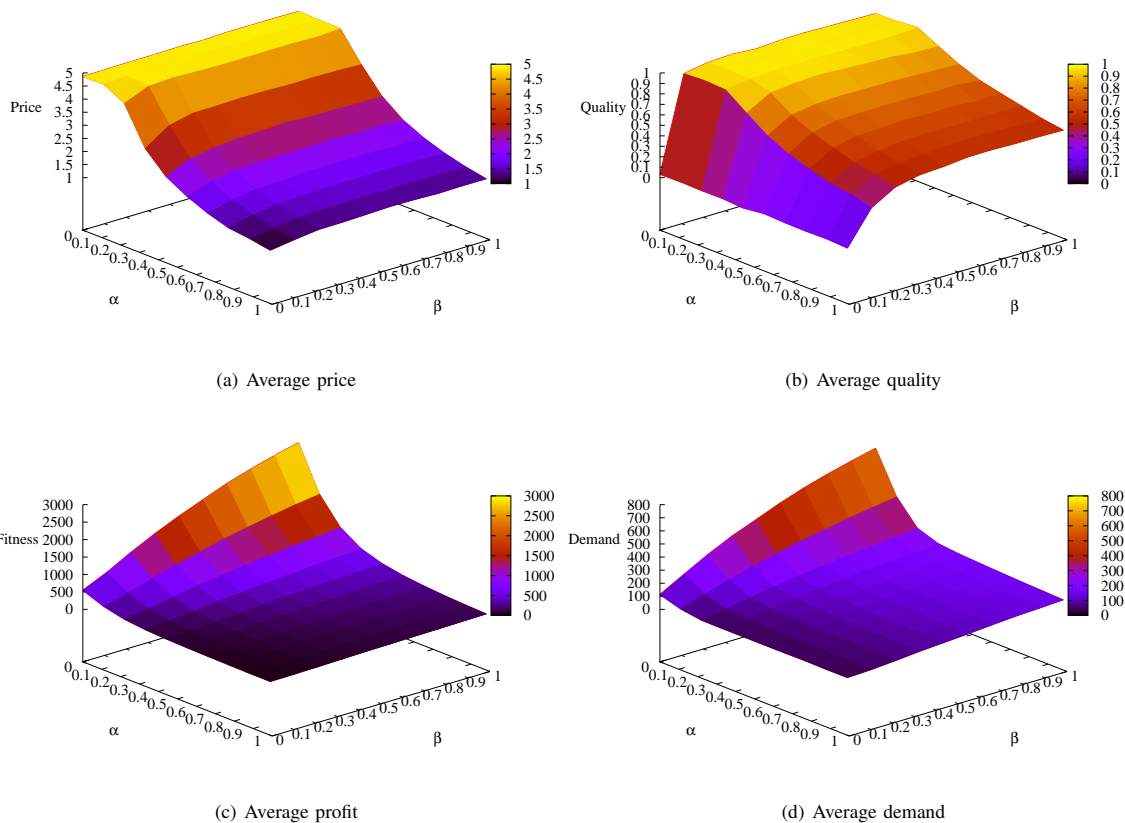


Fig. 2. Agent behaviors for values of α and β in homogeneous markets

B. Competition in heterogeneous markets

In this section, we examine a scenario where agents compete on price and quality in heterogeneous markets. In heterogeneous markets, each market has different price and quality sensitivities. Our purpose is to investigate how agents' strategies evolve over time in heterogeneous markets. The 5 different markets are set as (Market 1: $\alpha = 0.1$ and $\beta = 0.8$), (Market 2: $\alpha = 0.1$ and $\beta = 0.1$), (Market 3: $\alpha = 0.4$ and $\beta = 0.4$), (Market 4: $\alpha = 0.8$ and $\beta = 0.8$) and (Market 5: $\alpha = 0.8$ and $\beta = 0.1$). Each market represents different degrees of price and quality sensitivities. Our markets have the following features. Markets 1, 2 and 3 have lower price sensitivities while Markets 4 and 5 have higher price sensitivities. Markets 2, 3 and 5 have lower quality sensitivities, while Markets 1 and 4 have higher quality sensitivities.

Figure 3 shows the average data from 50 runs. Figures 3(a), 3(b), 3(c), 3(d) and 3(e) depict how the firm agents' average price, quality, profit, each market demand and the firm agent numbers evolve over time. From these figures, we notice that the markets' preferences on price are significant factors on the evolution of market price levels. As Figure 3(a) shows, the price levels evolve to higher levels in Markets 1, 2 and 3 (lower price sensitivities), while in Markets 4 and 5 (higher price sensitivities), the agent price levels evolve to lower levels. This also stems from firm agents' competition provided by our GA

which we have discussed in the homogeneous markets. More interestingly, the effect of quality preferences in heterogeneous markets is different with that in homogeneous markets. For example, the quality levels in Market 4 do not evolve to a higher level although Market 4 is a higher quality sensitive market as shown in Figure 3(b). Conversely, in Market 2, the quality levels evolve to a higher level although this market has very low quality sensitivities. This derives from the features of these markets. Market 4 is very sensitive to price and subsequently, firm agents from this market have to reduce their product price levels. This drives their profits down and consequently, they have lower incentive to produce higher quality products although consumers in this market prefer higher quality products. For Market 2, we can apply similar analysis. Finally, we can observe the emergence strategies of the firm agents that many firm agents enter into Market 1 which is a lower price sensitive and higher quality sensitive market as Figure 3(e) shows. Although higher quality levels lead to a production cost, it results in a higher market demand. In Market 1, agents have to produce higher quality products which will incur higher quality cost, but also could stimulate consumer demand. In fact, due to the relationship of price and quality preferences, Market 1 becomes the biggest one among the 5 markets (see Figure 3(d)). Furthermore, we find that many agents rush into Market 1 which increases the degree

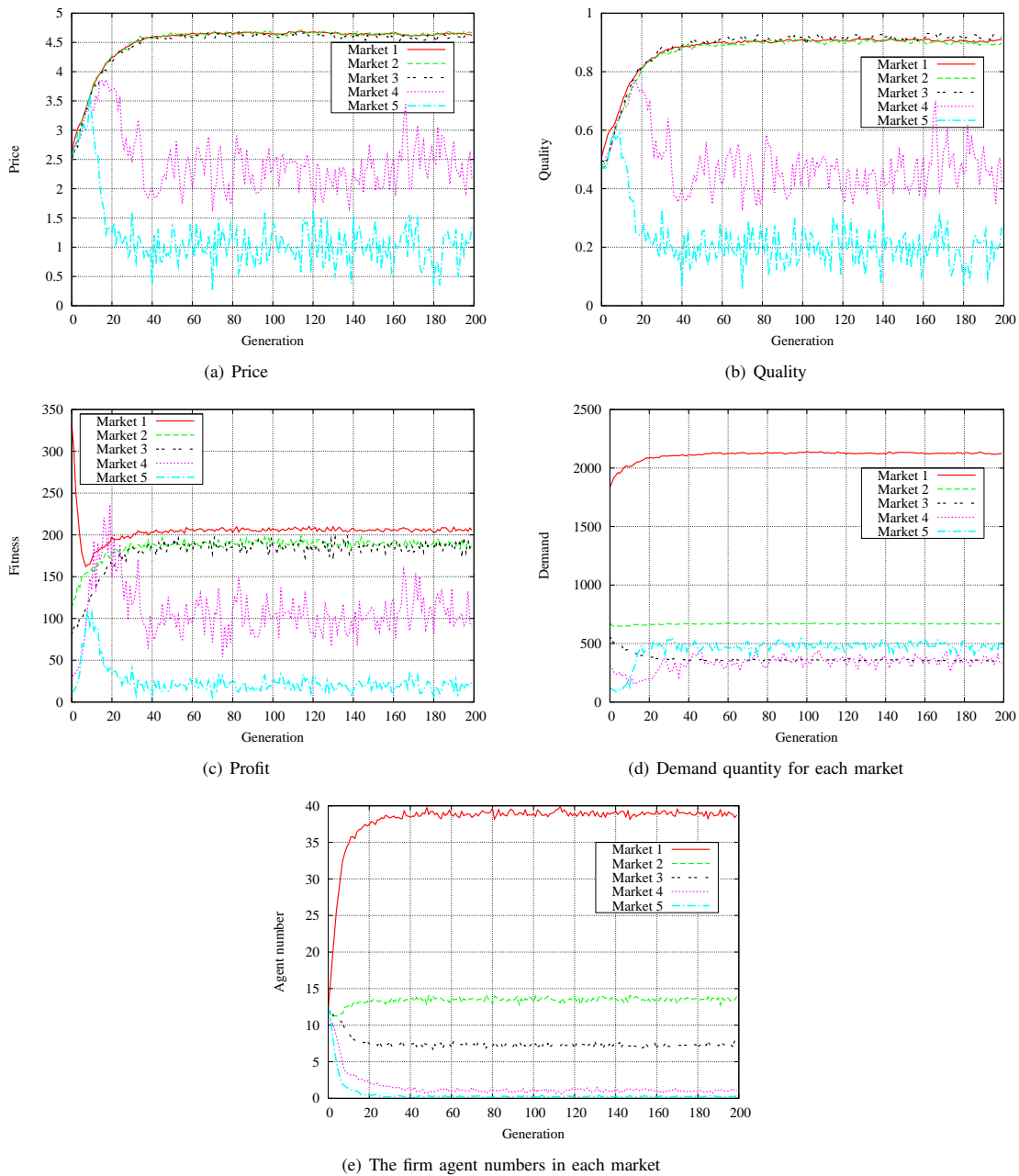


Fig. 3. Heterogeneous markets ((Market 1: $\alpha = 0.1, \beta = 0.8$), (Market 2: $\alpha = 0.1, \beta = 0.1$), (Market 3: $\alpha = 0.4, \beta = 0.4$), (Market 4: $\alpha = 0.8, \beta = 0.8$) and (Market 5: $\alpha = 0.8, \beta = 0.1$))

of competition. Subsequently, this drives the average profit down at the beginning as Figure 3(c) shows. However, the average profit in Market 1 goes up a little due to their learning on Market 1's preferences. Furthermore, we observe that the distribution of agents in the markets is related to the average agent profits of the markets. This reflects the agents's rational choices on market position.

Furthermore, we compare the agent numbers in each market from homogeneous markets and heterogeneous markets. Table II shows the agent's distribution in both market settings. This

data is recorded from 50 runs of our simulator over 200 generations. From this table, we can find that the number of agents is almost evenly distributed in homogeneous markets since there are no differences in markets. The distribution of agents in heterogeneous markets reflects the bias of agents' preferences which has been analysed above.

V. C S F W

The research outlined in this paper have investigated the evolution of the price and quality competition under a range

TABLE II

A

Market Name	Homogeneous markets		Heterogeneous markets	
	Agent Number	Standard Deviation	Agent Number	Standard Deviation
Market 1	11.9	1.1	36.3	3.0
Market 2	12.3	1.2	14.1	2.7
Market 3	11.8	0.9	6.9	1.8
Market 4	11.6	1.1	2.1	1.5
Market 5	12.4	0.9	0.6	1.1

of market configurations. This research holds particular significance for those interested in price and quality competition. The contributions of this paper include the following several aspects.

We have proposed a new multi-agent price and quality competition model. This model differs from existing models in a number of ways. Firstly, we consider many firms competing with each other in a number of markets simultaneously while existing models only consider one or two firms competing in one market such as Hotelling's Model [2], and Banker et al's model [3]. Furthermore, different markets may have different properties, such as the demand size, price and quality sensitivities. Secondly, we design a new mechanism to determine each firm agent's demand quantity. This mechanism indirectly reflects each firm agent's market share is not only determined by their own strategies but also affected by other agents' decisions. This model could be easily extended to include many extra features, such as advertisement effects of firms.

We have investigated the impact of market preferences on the evolution of price and quality under a range of market configurations. We find that the price and quality relationship in a given market is a primary driver of the evolution of firm agent strategies in that market. Firm agents' price strategies evolve rapidly to reflect the preferences of the markets in both homogeneous and heterogeneous markets. We can also observe the similar features for the agents' quality strategies in the homogeneous markets. However, in heterogeneous markets, the agents' average quality levels evolve to higher quality level even in the lower quality sensitive markets despite the penalties incurred by higher cost of increased quality due to the positive impact of quality. Furthermore, we notice an emergent phenomenon in the heterogeneous markets that the firm agents prefer low price and high quality sensitive markets. Based on these results, we can conclude that market preferences have significant effects on the agents' rational decisions. These results have potentially interesting applications to real-world market dynamics and help make strategic decisions on market competition and market entry.

However, there are still a number of factors that influence this study. Firstly, the assumption that all the firms have the same quality cost function is not realistic in the real world. Secondly, the firm agents in our model have no capacity constraints. In future, we would like to improve our model and explore its applications to the real-world market dynamics.

A

The authors would like to gratefully acknowledge the continued support of Science Foundation Ireland.

R

- [1] L. W. Phillips, D. R. Chang, and R. D. Buzzell, "Product quality, cost position and business performance: A test of some key hypotheses," *The Journal of Marketing*, vol. 47, no. 2, pp. 26–43, 1983. [Online]. Available: <http://www.jstor.org/stable/1251491>
- [2] H. Hotelling, "Stability in competition," *The Economic Journal*, vol. 39, no. 153, pp. 41–57, 1929. [Online]. Available: <http://www.jstor.org/stable/2224214>
- [3] R. D. Banker, I. Khosla, and K. K. Sinha, "Quality and competition," *Manage. Sci.*, vol. 44, no. 9, pp. 1179–1192, 1998.
- [4] N. Matsubayashi and Y. Yamada, "A note on price and quality competition between asymmetric firms," *European Journal of Operational Research*, vol. 187, no. 2, pp. 571 – 581, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/B6VCT-4NJ0TH8-5/2/e8c7e1d12dfec6c8ed064d4f3f34e606>
- [5] J. H. Holland and J. H. Miller, "Artificial adaptive agents in economic theory," *American Economic Review*, vol. 81, no. 2, pp. 365–71, May 1991. [Online]. Available: <http://ideas.repec.org/a/aea/aecrev/v81/y1991i2p365-71.html>
- [6] L. Tesfatsion, "Agent-based computational economics: Growing economies from the bottom up," *Artif. Life*, vol. 8, no. 1, pp. 55–82, 2002.
- [7] K. S. Moorthy, "Product and Price Competition in a Duopoly," *Market Science*, vol. 7, no. 2, pp. 141–168, 1988. [Online]. Available: <http://mktsci.journal.informs.org/cgi/content/abstract/7/2/141>
- [8] J. Vörös, "Product balancing under conditions of quality inflation, cost pressures and growth strategies," *European Journal of Operational Research*, vol. 141, no. 1, pp. 153 – 166, 2002. [Online]. Available: <http://www.sciencedirect.com/science/article/B6VCT-45WYVSK-C/2/00d166996cca8bfb329bd190096a44a7>
- [9] M. J. Jennings, Nicholas R.; Wooldridge, *Agent Technology Foundations, Applications, and Markets*, 2002.
- [10] R. E. Marks, "Breeding hybrid strategies: Optimal behaviour for oligopolists," *Journal of Evolutionary Economics*, vol. 2, no. 1, pp. 17–38, March 1992. [Online]. Available: <http://ideas.repec.org/a/spr/joevec/v2y1992i1p17-38.html>
- [11] H. Dawid, *Adaptive Learning by Genetic Algorithms: Analytical Results and Applications to Economic Models*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.
- [12] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
- [13] T. Riechmann, "Genetic algorithm learning and evolutionary games," *Journal of Economic Dynamics and Control*, vol. 25, no. 6-7, pp. 1019–1037, June 2001. [Online]. Available: <http://ideas.repec.org/a/eee/dyncon/v25y2001i6-7p1019-1037.html>
- [14] T. C. Price, "Using co-evolutionary programming to simulate strategic behaviour in markets," *Journal of Evolutionary Economics*, pp. 219–254, Jun. 1997. [Online]. Available: <http://ideas.repec.org/p/cla/levarc/588.html>
- [15] N. S. Tay and R. F. Lusch, "A preliminary test of hunt's general theory of competition: using artificial adaptive agents to study complex and ill-defined environments," *Journal of Business Research*, vol. 58, no. 9, pp. 1155 – 1168, 2005. [Online]. Available: <http://www.sciencedirect.com/science/article/B6V7S-4D34NPF-2/2/a024f676d852c9ed088dda30bd37173c>
- [16] M. Polo, "Hotelling duopoly with uninformed consumers," *The Journal of Industrial Economics*, vol. 39, no. 6, pp. 701–715, 1991. [Online]. Available: <http://www.jstor.org/stable/2098672>

Learning Virtual Agents for Decision-Making in Business Simulators

Javier García and Fernando Fernández
Universidad Carlos III de Madrid
Avenida de la Universidad, 30, 28911
Leganés, Madrid, Spain
Email: fjgpolo,ffernand@inf.uc3m.es

Fernando Borrajo
Universidad Autónoma de Madrid
Ctra. de Colmenar Viejo, km. 14, 28049, Madrid, Spain
Email: fernando.borrajo@uam.es

Abstract—In this paper we describe SIMBA, a simulator for business administration, as a Multi-Agent platform for the design, implementation and evaluation of virtual agents. SIMBA creates a complex competitive environment in which intelligent agents play the role of business decision makers. An important issue of SIMBA architecture is that humans can interact with virtual agents. Decision making in SIMBA is a challenge, since it requires handling large and continuous state and action spaces. In this paper, we propose to tackle this problem using Reinforcement Learning (RL) and K-Nearest Neighbors (KNN) approaches. RL requires the use of generalization techniques to be applied in large state and action spaces. We present different combinations in the choice of the generalization method based on Vector Quantization (VQ) and CMAC. We demonstrate that learning agents are very competitive, and they can outperform human expert decision strategies from business literature.

I. INTRODUCTION

Business simulators are a promising tool for research. The main characteristic of SIMBA (*SIMulator for Business Administration*) [2] is that it emulates business reality. It can be used from a competitive point of view, since different companies compete among themselves to improve their results. In this paper, SIMBA is considered as a multi-agent framework where the different agents manage their companies in different ways. SIMBA can include several autonomous agents to play the role of competing teams and, based on the research on decision making patterns of human teams, further research is made to improve the complexity and effectiveness of such intelligent agents.

Decision making in SIMBA requires handling more than 100 continuous state variables, and more than 10 continuous decision variables, which makes the problem hard even for business administration experts. The motivation of this paper is the design, implementation and evaluation of virtual agents in SIMBA using different machine learning (ML) approaches. The goal is that the developed agents can outperform human-like behavior when competing against hand-coded and random virtual agents, but also against expert humans players.

Human players have experimented the consequences of their decisions in competition with the developed virtual agents. But, given that the agents try to “win” in all cases, they make the game too hard for novice players. So “pedagogical” objectives for human players competing with our virtual

agents, are not directly included in the goal of this paper. Designing virtual agents whose behavior challenges human players adequately is a key issue in computer games development [23]. Games are boring when they are too easy and frustrating when they are too hard [8]. Difficulty of the game is critically important for its “pedagogical” worth. The game difficulty must be such that it is “just barely too difficult” for the subject. If the game is too easy or too hard, “pedagogical” worth appears to be less efficient. So most games allow human players adjust basic difficulty (easy, medium, hard).

However, developing agents that can outperform human-like behavior, under narrow circumstances, can do pretty well [15] (ex: *chess* and *Deep Blue* or *Othello* and *Logistello*). *Deep Blue* defeated World Chess Champion Garry Kasparov in an exhibition match. Campbell and Hsu describe the architecture and implementation of their chess machine in the paper [7]. A few months after this chess success, *Othello* became the new game to fall to computers when Michael Buro’s program *Logistello* defeated the World Othello Champion Takeshi Murakami. In the paper [3], Buro discusses the learning algorithms used in his program. Thus, the goal of this paper is the development of virtual “business” agents that can be able to beat hand-coded and random virtual agents, but also human business experts.

To do so, we use two different learning approaches. The first one is Instance Based Learning (IBL). In this paper we propose the Adaptive KNN algorithm, a variation of KNN, where experience tuples are stored and selected automatically to generate new behaviors.

However, decision making for business administration is an episodic task where decisions are sequentially taken. Therefore we also propose to use Reinforcement Learning (RL). The RL agents developed need to apply generalization techniques to perform the learning process, given that both the state and action spaces are continuous. In this paper, we propose two different generalization methods in order to tackle the large state and action spaces. The first one, Extended Vector Quantization for Q-Learning, uses Vector Quantization (VQ) to discretize both the state and action spaces, extending previous works where VQ was used only to discretize the state space[6]. Some tasks have been solved by coarsely discretizing the action variables [14], but up to our knowledge, this is the first time that VQ is used to discretize the action space.

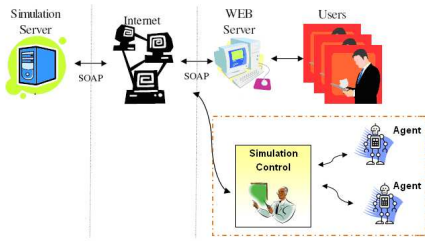


Fig. 1. SIMBA's Architecture

The second generalization approach, CMAC-VQQL, is based on the combination of VQ to discretize the action space and CMAC (*Cerebellar Model Articulation Controller*) [1], which is motivated by CMAC's demonstrated capability to generalize the state space.

Section II describes SIMBA. Section III introduces the learning approaches proposed, while Section IV shows how these approaches have been used to learn the virtual agents for decision making in SIMBA. Section V shows comparative results of the virtual agents, when competing among them but also when competing against expert human players. Section VI summarizes the related work. Last, Section VII concludes.

II. SIMBA

In this section, *SIMBA* simulator is described in detail.

A. *SIMBA's Architecture*

Figure 1 shows the architecture of the business simulator from a Multi-Agent perspective. The architecture designed enables multiple players to interact with the simulator, including both software agents and human players. The main components of the system are:

- **Simulation Server:** Once all decisions are taken for the current round, it computes the values of the variables in the marketplace for every player. Finally, it sends the results computed to each player. The player (software or human) uses these results to choose the best decisions in the next round of the simulation.
- **Simulation Control:** It manages the software agents and their decisions. It receives the decision taken by the software agents and sends them to the Simulation Server. The simulation server the results computed to the simulation control. The simulation control sends the results to the corresponding software agent.
- **Software Agents:** They represent an alternative to human players. In every step, the software agents receive the results computed for the Simulation Server. The software agents use this information to take the decisions for the next round of the simulation.

B. *Business Human Strategies*

Different business strategies appear in the business literature, and they all could be followed to manage the companies

in SIMBA, as will be shown in Section V. We describe some classical ones:

1. Incremental decisions. This type of business strategy is based on incremental decisions for all decision variables, which typically ranges from a 10% to a 20%. This business strategy is considered as a conservative behavior.

2. Risk decisions. It is based on strong changes in business decisions. It has strong impacts in market reactions, and is useful to detect gaps and market opportunities.

3. Reactive. An organization with this type of strategy attempts to locate and maintain a secure niche in a relatively stable product or service area [11].

4. Low cost strategy. With this strategy, managers try to gain a competitive advantage by focusing the energy of all the departments on driving the organization's costs down below the costs of its rivals [12].

5. Differentiation and specialization. A differentiation strategy is seen when a company offers a service or product that is perceived as unique or distinctive from its competitors [12].

Which strategy management is chosen in every moment depends on the organization's strengths and its competitor's weaknesses.

C. *Autonomous Decision Making in Simba*

The goal of this section is to describe how a *SIMBA* software agent can be implemented. To do this, we describe the state and action spaces, the transition function to transit between states and the variable to maximize.

State Space. The state computed in every round or simulation step is composed of 174 continuous variables. Table I shows some of the features that compose the state space.

Action Space. The players (software or humans) must approach the decisions on the different functional areas of their companies. Each market in the competition requires the use of 25 variables. This is an indicator of *SIMBAS's* capacity to approach the complexity of managerial decision-making. In our experiments, we consider a subspace of the total action space and we use only the ten variables shown in table I. This reduction was suggested by the experts, because the discarded variables are not very significant. All the actions that the agents can perform are constrained by the semantic of the business model. For instance, a company can not sell its product if it does not have stock.

Transition function. The different players participate in a simulation in a step by step round mode. Each simulation step is called a period, which is equivalent to three real months. When a round ends, the time machine is run. By doing this, the simulator integrates the previous periods situation, the teams' decisions, and the parameters of the general economic environment together with those of each geographic market, and orders the Simulators Server to generate output information for the new period.

Variable to maximize. The agents try to maximize the result of the exercise (profit). From a RL point of view, the objective is to maximize the total reward received. In this case, we

TABLE I
A SUBSET OF FEATURES OF THE STATE AND ACTION SPACES.

FEATURES of the State Space	FEATURES of the Action Space
Account value	Selling price
Human resources	Advertising expenses
Material cost	Network sales budget
Operating margin	Commercial information
Financial expenses	Training budget
Pre-tax income	Production scheduled
Tax	Material order
Training expenses	Research and Development budget
Bank overdraft	Loan
Economic productivity	Term loan
Advertising prediction	
Effort sales network	

define the immediate reward as the result of the exercise in a period or step. Therefore, there is no delayed reward and, like in other classical domains like Keepaway [17], immediate rewards received in every simulation step are relevant.

III. PROPOSED ALGORITHMS FOR LEARNING VIRTUAL AGENTS

In this section we describe the new learning algorithms proposed, based on KNN and RL.

A. Adaptive KNN

In this paper, we propose a variant of KNN called Adaptive KNN (Table II). In this variant, we can distinguish two phases. In the first one, a data set C is obtained during an interaction between the agent and the environment. This data set C is composed by tuples in the form $\langle s, a, r \rangle$ where $s \in S$, $a \in A$ and $r \in \mathfrak{R}$ is the immediate reward. In the second one, the set C obtained in the previous phase is improved during a new interaction between the agent and the environment. In each step of this second phase, the simulator returns the current state s where the agent is. The algorithm selects the K nearest neighbors to the state s in C . Among these K neighbors, it selects the tuple with the best reward obtained in the phase one. Then modify slightly the actions of this tuple and execute it. If the new reward obtained is better than the worst reward in K , it replaces the worst tuple in K with the new experience generated. Thus, the algorithm adapts the initial set C obtained in the phase one, to get increasingly better results in the second phase.

B. RL Approaches

Among many different RL algorithms, Q-learning has been widely used in the literature [20]. In Q-Learning, the update function is performed following equation 1, where α is a learning parameter, and γ is a discount factor that reduces the relevance of future decisions.

$$Q(s_t, a_t) \rightarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

Except in very small environments it is impossible to enumerate the state and action spaces. In this section we explain two new approaches for state and action space generalization problem.

TABLE II
ADAPTIVE KNN ALGORITHM

Adaptive KNN
1. Gather experience tuples <ol style="list-style-type: none"> 1.1. Generate the set C of experience tuples of the type $\langle s, a, r \rangle$ from an interaction of the agent in the environment, where $s \in S$, $a \in A$ and $r \in \mathfrak{R}$ is the immediate reward.
2. During a new interaction between the agent and the environment <ol style="list-style-type: none"> 2.1 Get state s from simulator 2.2 Select the K nearest neighbors of s in the set C <ol style="list-style-type: none"> 2.2.1 For each tuple $c_i \in C$, where $c_i = \langle s_i, a, r \rangle$, calculate $d(s, s_i)$ 2.2.2 Order $d(s, s_i)$ from lowest to highest 2.2.3 Select first K tuples, C_K 2.3. Select the tuple c_b with the best r, where $c_b = \langle s_b, a_b, r_b \rangle$ and $c_b \in C_K$ 2.4. Modify a_b from c_b, $a_m = a_b \pm \text{random} \Delta$ 2.5. Execute action a_m obtaining reward $r' \in \mathfrak{R}$ 2.6. Update set C using the new experience <ol style="list-style-type: none"> 2.6.1. Select the tuple $c_w \in C_K$ with the worst reward r_w 2.6.2. if $r' > r_w$ then replace the tuple $c_w = \langle s_w, a, r_w \rangle$ with the tuple $\langle s, a_m, r' \rangle$
3. Return C

TABLE III
EXTENDED VQQL ALGORITHM

Extended VQQL
1. Gather experience tuples <ol style="list-style-type: none"> 1.1. Generate the set C of experience tuples of the type $\langle s_1, a, s_2, r \rangle$ from an interaction of the agent in the environment, where $s_1, s_2 \in S$, $a \in A$ and $r \in \mathfrak{R}$ is the immediate reward.
2. Reduce the dimension of the state space <ol style="list-style-type: none"> 2.1. Let C_s the set of states in C 2.2. Apply a feature selection approach using C_s to reduce the number of features in the state space. The resulting feature selection process is defined as a projection $\Gamma : S \rightarrow S'$ 2.3. Set $C'_s = \Gamma(C_s)$
3. Discretize the state space <ol style="list-style-type: none"> 3.1. Use GLA to obtain a state space discretization, $D_{s'} = s'_1, s'_2, \dots, s'_n$, $s'_i \in S'$, from C'_s. 3.2. Let $VQ^{S'} : S' \rightarrow D_{s'}$ the function that given any state in S' returns the discretized value in $D_{s'}$.
4. Discretize the action space <ol style="list-style-type: none"> 4.1. Let C_a the set of actions in C 4.2. Use GLA to obtain an action space discretization, $D_a = a_1, a_2, \dots, a_m$, $a_i \in A$, from C_a 4.3. Let $VQ^A : A \rightarrow D_a$ the function that given any state in A returns the discretized value in D_a
5. Learn the Q-Table <ol style="list-style-type: none"> 5.1. Map the set C of experience tuples to a set C'. For each tuple $\langle s_1, a, s_2, r \rangle$ in C, introduce in C' the tuple $\langle VQ^{S'}(\Gamma(s_1)), VQ^A(a), VQ^{S'}(\Gamma(s_2)), r \rangle$ 5.2. Apply the Q-Learning update function defined in equation 1 to learn a Q table $Q : D_{s'} \times D_a \rightarrow \mathfrak{R}$, using the set of experience tuples C'
6. Return $Q, \Gamma, VQ^{S'}$, and VQ^A

1) *Extended VQQL for state and action space generalization:* Applying VQ techniques permits to find a more compact representation of the state and action space [6]. A vector quantizer Q of dimension K and a size N is a mapping from a vector (state or action) in the K -dimensional Euclidean space, R^k , into a finite set C containing N states, $Q : R^k \rightarrow C$ where $C = \{y_1, y_2, \dots, y_N\}$, $y_i \in R^k$. In this way, given C , and a state $x \in R^k$, $VQ(x)$ assigns x to the closest state from C , $VQ(x) = \arg \min_{y \in C} \{dist(x, y)\}$.

To design the vector quantizer we use the Generalized Lloyd Algorithm (GLA). The Extended VQQL algorithm is shown in Table III.

It uses VQ to generalize the state and action spaces. In Extended VQQL algorithm, two vector quantizers are designed for each agent. The first one is used to generalize the state space and the second one is used to generalize the action space. The vector quantizers are designed from the input data C obtained during an interaction between the agent and the environment. The data set C is composed by tuples in the form $\langle s_1, a, s_2, r \rangle$ where s_1 and s_2 is in the state space S , a is in the action space A and r is the immediate reward. In many problems, s is composed by a large number of features. In these cases, we suggest to apply feature selection to reduce the number of features in the state space. Feature selection is a technique of selecting a subset of relevant features for building a new subset. So feature selection is used to select the relevant features of S to obtain a subset S' . This feature selection process is defined as $\Gamma : S \rightarrow S'$. The set of states $s' \in S'$, C'_s , are used as input for the Generalized Lloyd Algorithm to obtain the first vector quantizer. The vector quantizer $VQ^{s'}$ is a mapping from a vector $s' \rightarrow S'$ into a vector $s' \in D_{s'}$, where $D_{s'}$ is the state space discretization $D_{s'} = \{s'_1, s'_2, \dots, s'_n\}$ for $s'_i \in S'$. The set of actions $a \in A$, C_a , are used as input for the GLA to obtain the second vector quantizer.

The vector quantizer VQ^A is a mapping from a vector $a \in A$ into a vector $a \in D_a$, where D_a is the action space discretization $D_a = \{a_1, a_2, \dots, a_m\}$ for $a_i \in A$. In the last part of the algorithm, the Q-table is learned from the obtained discretizations using the set C' of experience tuples. To obtain the set C' from C , each tuple in C is mapped to the new representation. Therefore, every state in C is firstly projected to the space S' and then discretized, i.e. $VQ^{S'}(\Gamma(S))$; every action $a \in A$ in C is also discretized $VQ^A(a)$.

2) *CMAC-VQQL for state and action space generalization*: CMAC is a form of coarse coding [20]. In CMAC the features are grouped into partitions of input state space. Each of such partition is called a *tiling* and each element of a partition is called a *tile*. Each *tile* is a binary feature. The tilings were overlaid, each offset from the others. In each tiling, the state is in one tile. The approximate value function, Q_a , is represented not as a table, but as a parameterized form with parameter vector $\vec{\theta}_t$. This means that the approximate value function Q_a depends totally on $\vec{\theta}_t$. In CMAC, each tile has associated a weight. The set of all these weights is what makes up the vector $\vec{\theta}$. The approximate value function, $Q_a(s)$ is calculated in the equation 2.

$$Q_a(s) = \vec{\theta}^T \vec{\phi} = \sum_{i=0}^n \theta^{(i)} \phi^{(i)} \quad (2)$$

The CMAC-VQQL algorithm, described in Table IV, combines two generalization techniques. It uses CMAC to generalize the state space and VQ to generalize the action space. In this case, a data set C is obtained during an interaction between the agent and the environment. This data set C is composed by tuples in the form $\langle s_1, a, s_2, r \rangle$ where s_1 and s_2 is in the state space S , a is in the action space A and r is the immediate reward. In the same way that previously, s is

TABLE IV
CMAC-VQQL ALGORITHM

CMAC-VQQL
1. Gather experience tuples
1.1. Generate the set C of experience tuples of the type $\langle s_1, a, s_2, r \rangle$ from an interaction of the agent in the environment, where $s_1, s_2 \in S$, $a \in A$ and $r \in \mathfrak{R}$ is the immediate reward.
2. Reduce the dimension of the state space
2.1. Let C_s the set of states in C
2.2. Apply a feature selection approach using C_s to reduce the number of features in the state space. The resulting feature selection process is defined as a projection $\Gamma : S \rightarrow S'$
2.3. Set $C'_s = \Gamma(C_s)$
3. Discretize the action space
3.1. Let C_a the set of actions in C
3.2. Use GLA to obtain an action space discretization, $D_a = \{a_1, a_2, \dots, a_m\}$, $a_i \in A$, from C_a
3.3. Let $VQ^A : A \rightarrow D_a$ the function that given any state in A returns the discretized value in D_a
4. Design CMAC
4.1. Design a CMAC function approximator from C'_s taking into account the obtained action space D_a .
5. Approximate the Q function
5.1. Map the set C of experience tuples to a set C' . For each tuple $\langle s_1, a, s_2, r \rangle \in C$, introduce in C' the tuple $\langle \Phi(\Gamma(s_1)), VQ^A(C_a), \Phi(\Gamma(s_2)), r \rangle$ where Φ is the binary vector of features
5.2. Update the vector weights θ for the action $VQ^A(C_a)$ using $\Phi(\Gamma(s_1))$, $\Phi(\Gamma(s_2))$ and r .
5.3. Apply the approximate value function defined in equation 2 to approximate the Q function for the action $VQ^A(C_a)$ using θ and $\Phi(\Gamma(C_s))$.
6. Return Q, Γ , θ , and VQ^A

composed by a large number of features. Feature selection is used to select a subset S' of the relevant features of S .

The set of actions $a \in A$, C_a , are used as input for the GLA to obtain the second vector quantizer. The vector quantizer VQ^A is a mapping from a vector $a \in A$ into a vector $a \in D_a$, where D_a is the action space discretization $D_a = \{a_1, a_2, \dots, a_m\}$ for $a_i \in A$. Later, the CMAC is built from C'_s taking into account the obtained action space D_a . For each state variable x'_i in $s' \in S'$ the tile width and the number of tiles per tiling are selected taking into account their ranges. In our work, a separate value function for each of the discrete actions is used. In CMAC, each tile has associated a weight. The set of these weights is what makes up the vector θ . In the last part, the Q function is approximated by the equation 2.

IV. VIRTUAL AGENTS IN SIMBA

In the following evaluation performed, we assume that 6 companies are controlled by agents of different types. These agents are: Random Agents, that assign to each decision variable a random value following an uniform distribution; Hand-Coded Agents, that modify their decision variables by increasing their values using the Consumer Price Index (CPI); RL Agents, using the Extended VQQL and CMAC-VQQL algorithms described in Section III-B; and Adaptive KNN Agents, using the algorithm described in section III-A.

3) *Executing the Extended VQQL Algorithm*: Executing the Extended VQQL algorithm to learn the VQ Agents requires performing the 5 steps of the algorithm:

Step 1: Gather experience tuples. To gather experience, we perform an exploration in the domain by using hand-coded agents. Specifically, we obtain the experiences generated by a hand-coded agent managing company 1 against five hand-coded agents managing companies 2, 3, 4, 5 and 6 respectively.

Step 2: Reduce the dimension of the state space. The goal of this step is to select, from among all features in the state space, those features most related to the reward (the result of the exercise). To perform this phase, we use the data-mining tool, WEKA [22] using the attribute selection method *CfsSubsetEval*. This method evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. The resulting description of the state space after the attribute selection process is shown in Table I.

Step 3: State space discretization. Now, we use the GLA to discretize the state space.

Step 4: Discretize the action space. Again, we use GLA to discretize the action space. The action space is composed of the features shown in Table I.

Step 5: Learn the Q table. Once both the state and action spaces are discretized, the Q function is learned using the mapped experience tuples and the Q-Learning update function. The Q table is generated, composed of n rows (where n is the number of discretized states) and m columns (where m is the number of discretized actions).

4) *Executing CMAC-VQQL Algorithm:* Executing the CMAC-VQQL algorithm to learn the CMAC Agents requires performing the 5 steps of the algorithm as described in Table IV. Steps 1 and 2 of CMAC-VQQL are the same as steps 1 and 2 of Extended VQQL (gather experience and the reduction of the dimension of the state space). Step 3 of CMAC-VQQL (action space discretization) is also the same as step 4 of Extended-VQQL. Step 4 is the design of the CMAC function approximator. In our experiments we use single-dimensional tilings. For each state variable, 32 tilings were overlaid, each offset from the others by $1/32$ of the tile width. For each state variable, we specified the width of the tiles based on the width of the generalization that we desired. In the experiments we use three different configurations. The size of the primary vector θ in Configuration #1 is 754272 ($x_{1tiles} + x_{2tiles} + x_{12tiles}$), in Configuration #2 is 1364320, in Configuration #3 is 2440704. In our work, we use a separate value function for each of the generalized actions. Last, step 5 of the algorithm, learning the Q approximations, can be performed.

A. Adaptive KNN in SIMBA

To apply the Adaptive KNN algorithm to create a *SIMBA* software agent, we use the same state space, action space, and transition and reward functions that for the RL agent. We also use the same experience tuples than for the RL agent, although in the learning process, the set is updated following step 6 of the algorithm (as described in Table II).

TABLE V
RESULTS FOR DIFFERENT CONFIGURATIONS OF EXTENDED VQQL (IN MILLIONS OF EUROS).

Decisions States	128		64		32	
	Mean	Std	Mean	Std	Mean	Std
128	4,3	1,73	5,43	4,2E-04	7,73	0,09
64	6,21	3,15	7,51	0,32	8,14	0,51
32	4,94	3,68	5,69	0,06	7,62	0,28

TABLE VI
RESULTS FOR DIFFERENT CONFIGURATIONS OF CMAC-VQQL (IN MILLIONS OF EUROS).

Decisions Configuration	64		32		8	
	Mean	Std	Mean	Std	Mean	Std
1	4,87	1,62	6,49	0,04	7,0	0,12
2	6,23	0,13	5,82	0,90	6,25	0,37
3	5,26	0,20	5,95	3,2E-04	6,24	0,96

V. RESULTS

In the experiments, the learning agent always manages the first company of the six involved in the simulations. Each experiment consists of 10 simulations or episodes with 20 rounds and we obtain the mean value and the standard deviation for the result of the exercise during the 20 periods. In this situation, a hand-coded agent that manages the company 1 against five hand-coded agents that manage companies 2, 3, 4, 5 and 6 respectively obtains a mean value of the result of the exercise of 2,901,002.13 euros. A random agent in the same situation obtains -2,787,382.78 euros.

In the experiments with human experts, simulations have 8 rounds.

A. RL and KNN Results

In the first set of experiments we use the Extended VQQL algorithm to learn an agent that manages company 1 and plays against five hand-coded agents that manage companies 2, 3, 4, 5 and 6 respectively. The results for different discretizations size of the state (rows) and action (columns) spaces are shown in Table V.

The best result is obtained when we use a vector quantizer of 64 centroids (or states) to generalize the state space and a vector quantizer of 32 centroids (or actions) to generalize the action space.

In the second set of experiments we use the CMAC-VQQL algorithm. The results for the different CMAC configurations described in section IV-4 (rows) combined with the different sizes of the action space obtained by VQ (columns) are shown in Table VI.

The best result is obtained when we use the Configuration #1 of CMAC to generalize the state space and a vector quantizer of 8 centroids to generalize the action space. This value is smaller than the obtained with Extended VQQL but, again, all the configurations obtain better results than the hand-coded agent.

In the next set of experiments we use the KNN algorithm to build an agent. The results for the different KNN configurations are shown in Table VII.

TABLE VII
RESULTS FOR DIFFERENT CONFIGURATIONS OF KNN (IN MILLIONS OF EUROS).

Learning	K	5		10		15	
		Mean	Std	Mean	Std	Mean	Std
Adaptive		6,44	3,99	9,81	0,21	9,89	0,32
No adaptive		7,86	1,15	5,20	1,11	7,47	4,36

The columns of Table VII show different results for different values of K (5, 10 and 15 respectively). The first row presents the results of the Adaptive KNN algorithm, as it was described in Table II. The second row shows the results of a classical KNN approach, without the adaptation of the training set, i.e. without executing the steps five and six of the Adaptive KNN algorithm. The best results are obtained with the adaptive version, for $K=10$ and $K=15$. In these cases, we obtain a mean value for the result of the exercise of 9,8 millions of euros, which is higher than the ones obtained with RL.

In previous experiments, the learning agent always learned to manage the first company of the six involved in the simulations. However, the behavior of each company depends on their initial states and of historical data (periods -1, -2, etc). Therefore, learning performance may vary from one company to other. To evaluate this issue, we repeat the learning process for the best learning configurations, for each of the six companies. Each experiment consists of 10 simulations with 20 rounds and we obtain the mean value and the standard deviation for the result of the exercise during the 20 periods. The results shown in Figure 2 demonstrate that the Extended VQQL agent and Adaptive KNN agent obtain similar results, and both obtain better results than the hand-coded agent.

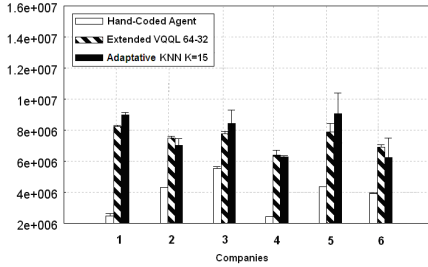


Fig. 2. Mean value and Standard deviation for the result of the exercise.

Now, we compare the behavior of the best RL agent with the behavior of the best Adaptive KNN agent obtained in previous experiments. In this experiment, all the companies have the same initial state and historical data, so the result is independent of the company managed. This experiment consists of 10 simulations with 20 rounds and we obtain the mean value and the standard deviation for the result of the exercise during the 20 periods. Figure 3 shows the mean value and the standard deviation for each kind of agent.

For the Adaptive KNN agent, the average value grows from the first period, and raises up to 16 millions of euros. However,

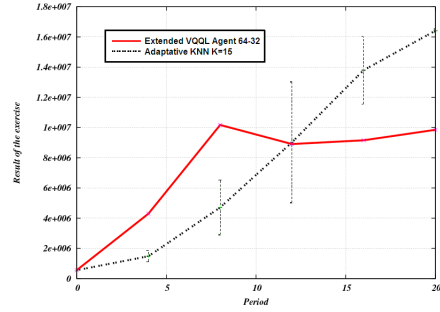


Fig. 3. RL Agents vs. Adaptive KNN Agents

TABLE VIII
RESULTS FOR INCREMENTAL DECISION STRATEGY (IN MILLIONS OF EUROS)

Agent	Simulation	10%	20%
Extended VQQL 64-32		7,27	7,28
Adaptive KNN K=15		1,58	1,58
Human Expert		0,56	-0,18

we see that standard deviation is very high, so the behavior of the agent managing different companies is very different. The result for the Extended VQQL agent have two behaviours well differentiated: before period 8, and after period 8. In the first part, the result of the exercise always grows up, and dominates the result of the Adaptive KNN agent. However, from period 8, the result of the exercise for the Extended VQQL agent stabilizes to a value of around 10 millions, and it is dominated by the other agent from period 10. Interestingly, we have revised all the simulations performed, and this behavior always appears. We believe that the RL agent is affected by the CPI and the evolution of the market and, with time, the actions obtained by the VQ algorithm becomes old-fashioned (note that 8 periods are equivalent to two years). Therefore, if we focus in the early periods, typically the RL agents behave better than the KNN ones.

B. RL and KNN Agents vs. Human Experts

In this section, we present experiments where software agents play against a human expert during 8 periods. The human expert actually is an associate full time professor in Strategic and Business Organization at Universidad Autónoma de Madrid (UAM), where he is Director of Master of Business Administration (Executive) and Director of Doctorate Program of Financial Economics.

In all the experiments, we use the best RL and Adaptive KNN agents obtained in the previous section. In the first experiment, the human expert uses the incremental decision strategy, described in section II. The results are shown in table VIII.

In this case, the Extended VQQL agent obtains the best results. Furthermore, given that only 8 episodes are run, the RL agent performs much better than the Adaptive KNN agent.

TABLE IX
AGENTS VS. HUMAN EXPERT (IN MILLIONS OF EUROS)

Agent	Simulation	1	2
Extended VQQL 64-32		5,36	6,09
Adaptive KNN K=15		3,47	2,53
Human Expert		-0,32	-1,30

The human expert obtains the worst results (independently of the increment used).

In the second experiment, two different simulations with 8 rounds each are performed. The human expert combines the use of the different business strategies described in section II. The results are shown in table IX.

In all the experiments, the software agents obtain better results than the human expert. From a qualitative point of view, the virtual agents usually compete in the same market scope. They are very effective and efficient, been almost impossible to beat them under the parameter setting used in these simulations. The best strategies usually make decisions in different market scopes, using high or low strategies (for instance, low cost or differentiation and specialization). It means that using more competitive strategies, the gap between the performance of the virtual agents and the human experts could be reduced.

VI. BACKGROUND

Business gaming usage has grown globally and has a long and varied history [4]. The first modern business simulation game can be dated back to 1932 in Europe and 1955 in North America. In 1932, Mary Birshstein, while teaching at the Leningrad Institute, got the idea to adapt the concept of war games to the business environment. In North America the first business simulator dates back to 1955, when RAND Corporation developed a simulation exercise that focused on the U.S. Air Force logistics system [9]. However, the first known use of a business simulator for pedagogical purposes in an university course, was at the University of Washington in a business policy course in 1957 [21].

From this point, the number of business simulation games in use grew rapidly. A 2004 e-mail survey of university business school professors in North America reported that 30.6% of 1,085 survey respondents were current business simulation users, while another 17.1% of the respondents were former business game users [5].

Over the years Artificial Intelligence (AI) and simulation have grown closer together. AI is used increasingly in complex simulation, and simulation is contributing to the development of AI [24]. The need for increased level of reality and fidelity in domain-specific games calls for the use of methods that bring realism and intelligence to actors and scenarios (also in business simulators). Intelligent software agents, called “autonomous” avatars or virtual players, are now being embodied in business games. Software agents can interact with each other and their environment producing new states, business information and events. In addition, these agents not only

provide information but also may affect the environment and direction of the simulation [19].

Machine learning techniques (decision trees, reinforcement learning, ...) have been used widely to develop software agents [18]. In [19] for example, the software agents uses decision trees to learn different behaviors. In this case, virtual players could take on the role of an executive or salesperson from a supplier firm, a union leader, or any other role relevant to the simulation exercise. In [10] the learning agents uses a typical genetic-based learning classifier system, XCS (*eXtended learning Classifier System*). In that work, RL techniques are used, allowing decision-making agents to learn from the reward obtained from executed actions and, in this way, to find an optimal behavior policy. In stochastic business games, the players take actions in order to maximize their benefits. While the game evolves, the players learn more about the best strategy to follow. With this, RL can be used to improve the behavior of the players in a stochastic business game [13]. However, in all these cases, the business simulator games used did not involve the huge state and action spaces that SIMBA involves.

In complex domains with large state and action spaces is necessary to apply generalization techniques such as VQ or CMAC. VQ has been used successfully in many other domains [6]. In addition, CMAC [17] are extensively used to generalize the state space, but the research on problems where the actions are chosen from a continuous space is much more limited. KNN has also been used in the scope of Business Intelligence. In [16], the authors investigates the relationship among corporate strategies, environmental forces, and the Balanced Scorecard (BSC) performance measures using KNN. In this case, the authors used all time the same initial set of experience and they did not try to adapt it, using the new experience generated during the game.

An important issue that make SIMBA different from other classical RL domains (like Keepaway [17]) is that it is not defined a priori as a cooperative or competitive domain. In SIMBA, the number of adversaries is very high, and the number of variables involved in the state and action space, too. In addition, in SIMBA software agents can play against humans. It is hard to find all these issues in other classical domains. So the learning process in SIMBA represent a real challenge.

VII. CONCLUSION

This paper introduces SIMBA as a business simulator which architecture enables different players, including both software agents and human players, to manage companies in different markets. The simulator generates a competitive environment, where the different agents try to maximize their companies’ profits. SIMBA represents a complex domain with large state and action spaces. Therefore, the learning approaches applied to generate the virtual agents must handle that handicap. We have demonstrated that the proposals presented, based on Lazy Learning and RL, achieve the goal of being very competitive

when compared with previous hand-coded strategies. Furthermore, we demonstrate that when competing with a human expert, which follows classical management strategies, the learning agents are able to outperform the behavior of the human.

In the case of RL, the choice of the generalization method have a strong effect on the results that we obtain. For this reason, the state and action space representation is chosen with great care, and we have proposed two new methods: Extended-VQQL and CMAC-VQQL. This is the first time that VQ is used to discretize the action space, and some preliminary results have shown that it is also useful in other domains, like autonomous helicopter control. The challenging results obtained by the learning approaches to generate virtual agents in SIMBA offers promising results for Autonomous Decision Making.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish MICINN project TIN2008-06701-C03-03 and by the Spanish TRACE project TRA2009-0080. The authors would like to thank the people from Simuladores Empresariales S.L.

REFERENCES

- [1] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, 97(3):220–227, 1975.
- [2] F. Borrajo, Y. Bueno, I. de Pablo, B. n. Santos, F. Fernández, J. García, and I. Sagredo. Simba: A simulator for business education and research. *Decision Support Systems*, June 2009.
- [3] M. Buro. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence*, 134:85–99, 2002.
- [4] A. J. Faria, D. Hutchinson, W. J. Wellington, and S. Gold. Developments in business gaming: A review of the past 40 years. *Simulation Gaming*, 40(4):464–487, August 2009.
- [5] A. J. Faria and W. J. Wellington. A survey of simulation game users, former-users, and never-users. *Simul. Gaming*, 35(2):178–207, 2004.
- [6] F. Fernández and D. Borrajo. Two steps reinforcement learning. *International Journal of Intelligent Systems*, 23(2):213–245, 2008.
- [7] F.-H. Hsu. *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press, Princeton, NJ, USA, 2002.
- [8] R. Hunicke and V. Chapman. Ai for dynamic difficulty adjustment in games. 2004.
- [9] J. R. Jackson. Learning from experience in business decision games. *California Management Review*, 1:23–29, 1959.
- [10] M. Kobayashi and T. Terano. Learning agents in a business simulator. In *Proceedings 2003. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2003.
- [11] R. E. Miles and C. C. Snow. *Organizational strategy, structure, and process*. McGraw-Hill, New York, 1978.
- [12] M. E. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, New York, 1 edition, June 1985.
- [13] K. K. Ravulapati and J. Rao. A reinforcement learning approach to stochastic business games. *IIE Transactions*, 36:373–385, 2004.
- [14] J. Santamaria, R. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6:163–217, 1998.
- [15] J. Schaeffer and H. J. van den Herik. Games, computers, and artificial intelligence. *Artif. Intell.*, 134(1-2):1–7, 2002.
- [16] M. H. Sohn, T. You, S.-L. Lee, and H. Lee. Corporate strategies, environmental forces, and performance measures: a weighting decision support system using the k-nearest neighbor technique. *Expert Syst. Appl.*, 25(3):279–292, 2003.
- [17] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.
- [18] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000.
- [19] G. J. Summers. Today’s business simulation industry. *Simul. Gaming*, 35(2):208–241, 2004.
- [20] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. Mit Pr, May 1998.
- [21] H. J. Watson. *Computer Simulation in Business*. John Wiley & Sons, Inc., New York, NY, USA, 1981.
- [22] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.
- [23] J. Yang, S. Min, C.-O. Wong, J. Kim, and K. Jung. Dynamic game level generation using on-line learning. In *Edutainment’07: Proceedings of the 2nd international conference on Technologies for e-learning and digital entertainment*, pages 916–924, Berlin, Heidelberg, 2007. Springer-Verlag.
- [24] L. Yilmaz, T. Ören, and N.-G. Aghaee. Intelligent agents, simulation, and gaming. *Simul. Gaming*, 37(3):339–349, 2006.

Looking for the self-fulfilling prophecy effect in a double auction artificial stock market

Abstract—This work proposes a double auction artificial stock market based on the Santa Fe market structure. Our market tries to shed light on some facts that usually arise in real stock markets, specially the creation of technical figures in price series. The origin of these figures is believed to be caused by the self-fulfilling prophecy effect, which will be investigated with the proposed market.

I. INTRODUCTION

The main purpose of the agent-based simulation of an artificial stock market (ASM) is to reproduce, in a controlled environment, some properties of real stock markets. In that way, ASMs are a suitable tool to analyze and understand market dynamics. See [1] for a comprehensive review of the topic.

Many market models has been developed in order to reproduced those properties, like Genoa Stock Market [2], [3], $\$$ -Game [4], or the Santa Fe Institute Stock Market (SFM), developed by LeBaron and his coauthors since the early nineties and analyzed in depth in [5]. Most of them are able to reproduce several well-known market properties -called stylized facts,- while each market has his own special microstructure. However, in the time series of prices that emerges in ASMs it is difficult to find some of the typical behaviors that appear in real-life price time series, for example, the bid-ask bouncing or the sideways movement within the support and resistance lines.

These behaviors, which are usually know as technical patterns, cannot be explained from the fundamental analysis perspective. Despite this fact, they are used by many investors and also by chief dealers, as is reported in [6]. The reality is that patterns such as trends, channels, resistances and supports can be spotted in stock charts. A possible explanation to these phenomena is the self-fulfilling prophecy effect [7]. As many people look for similar technical patterns in the stock markets and place their orders according to them, the patterns finally emerge as a result of this collective belief. This belief is reinforced when the stock price behave as expected, because technical traders feel confident with their chartist strategy and technical analysis is considered as a useful tool.

Technical analysts, also known as chartist investors, base their expectations in historical price patterns that are expected to appear again at some future point. They try to predict future extreme prices in order to buy assets when the value is under those limits, and sell them when the price is close to a bounce zone. Moreover, technical traders usually follow price trends. A common example is the use of the moving averages crosses to set their trading strategies. This method provides buy and

sell signals when a short run moving average crosses a long run moving average price.

Chartism is one of the two main investment approaches that can be usually found in stock markets [8]. The other one is fundamental trading, where investors base their investments upon future price expectations based on fundamental and economic factors, such as future dividend expectations, macroeconomic data and growth prospects. Nowadays, investors and chief dealers combine both technical and fundamental information [6]. Frankel and Froot in [9] showed that both approaches affected the US dollar exchange rate in the eighties. They associated the long-term expectations, which are stabilizing, with fundamentalists, and the shorter term forecasts, which seem to have a destabilizing nature, with the chartist expectations. As a result, many people used weighted averages of the chartist and fundamentalist forecasts in formulating their expectations for the value of the dollar at a given future date, with weights depending on how far the date is.

This article proposes an ASM based on the SFM structure that exhibits technical figures and that reproduce the self-fulfilling prophecy effect. The proposed ASM modifies some important features of the SFM with the aim of being more realistic and reproducing stylized facts. The most relevant changes are introduced in the next section.

II. DESCRIPTION OF THE DOUBLE AUCTION ARTIFICIAL STOCK MARKET PROPOSED

The SFM [10], [11] consists of a small number (typically 25) of artificially-intelligent agents that each period choose between investing in a stock and leaving their money invested in a fixed interest rate asset. The stock pays a stochastic dividend and has a price which fluctuates according to agent demand. The agents make their investment decisions by attempting to forecast the future return on the stock with the help of a set of forecast rules that are triggered when they match certain states of the market. Each rule map into a set of parameters that are used to yield a forecast for the future price and dividend using the rational expectations equilibrium theory. The forecast is converted to the share demand, according to the agent's demand function which follows risk aversion behavior. Agents learn through time because their predictive rules evolve by means of a genetic algorithm.

The SFM shows, amongst other features, the theoretically-predicted rational expectations behavior, with low overall trading volume, uncorrelated price series. However, it is difficult to find realistic market behavior such as high trading volume, time-varying volatility clustering (periods of swings followed

by periods of relative calm), bubbles and crashes and market patterns such as supports, resistances, channels, etc. One of the main reasons is that the auction mechanism is not realistic as there is an auctioneer that takes into the account the demand of shares and the fixed supply of shares (25 shares) to set the price that clears the market.

In our ASM, a continuous double auction system is implemented. In continuous double auction markets, agents place buy or sell orders at any time in an asynchronous manner. In this kind of auctions, a public order book lists the bids in descending order and the sell offers in ascending order. When a new order matches with the best waiting order of the opposite type then a trade is made, otherwise the new order remains waiting. Once the transaction is carried out, both orders are removed from the order book. This kind of auction is implemented in stock markets such as NYSE or AMEX. This kind of auction has been already implemented as an ASM [12], [13], [2], [14], where some aspects of market dynamics are successfully explored. As continuous double auction is commonly used in real stock markets, we believe that is the most suitable auction mechanism for an ASM that aims to replicate these markets.

In our market, agents are rationally bounded, which means that their rationality is limited by the information they have. They make price bids (offers to buy) and/or price asks (offers to sell) subject to a budget constraint and using the information they have about the state of the market. Our market allows agents to place both market-price and fixed-price orders. The ASM does not allow fixed-price orders. However, this kind of orders are used in real-life stock markets. In an artificial market that allows this orders it is possible to observe technical patterns. If a group of traders set fixed-price orders close to a certain price value, then supports or resistances lines may appear in the resulting price time series. Also, cascade effects could emerge behind certain price limits obtained using technical analysis.

In our ASM, agents tune the fixed-price orders using a system of forecasting rules similar to that used in the SFM, but based on support and resistance lines. In doing so, they will fix the price taking into account the support and resistance lines and the length of the trading horizon (it denotes if is a short-, mid- or long-term trade). The mechanism will be described below.

Our market follows the basic structure of the SFM model, but implementing a double auction market. Another noteworthy difference is the number of agents. Instead of the 25 agents used in the SFM, in our market there are 512 agents that will make possible to have enough trade operations in the market and a great variety of behaviors. It is important to remark that our aim is not related with the rational expectations equilibrium theory, but with the study of real-life phenomena such as resistance and support lines. These changes affect not only the auction mechanism but also the equations that determine the wealth and the classifier rules. More details will be given in the next subsections.

A. Agent's trading strategy

As in the SFM, our market has two assets: a risk free bond in infinite supply with constant interest rate ($r = 0.1$) and a risky asset. The price of the risky asset in t , p_t is endogenously determined by the market. The risky asset considered does not pay dividends, in contrast to the SFM's risky asset.

The trading strategy consists of buying (or selling, if the agent goes short) risky assets at the current price of the market and at the same time placing a fixed price stop-profit order. The price of the stop-profit order is determined by taking into account the agent's resistance line (or the agent's support line, if the agent goes short). Both orders are sent at time t_a . In addition, the agent also estimates the price of a stop-loss order using the support line (or the resistance line, if the agent goes short). This order will be placed as a market-price order only if the risky asset reaches the stop-loss price.

As any market-price operation has its corresponding stop-profit order, the total number of stocks M is always available in the market, providing the necessary liquidity to supply the possible demands of other investors. The system restrictions ensure the liquidity of the market and consequently market price orders are executed at the time when they are placed.

In t_a the stop-profit order is booked in its corresponding priority queue of awaiting orders, depending on if the trading agent goes long or short. The agent determines the stop-profit price using a future stock price that is forecasted with the help of a support line (or a resistance line if the investor is going short) that is drawn by the agent using three parameters determined by the activated forecast rule j (more details about the forecast rules are given in the next subsection). The parameters are: the number of local maximum (resp. minimum) points used to draw the resistance (resp. support) lines $a_{i,j}$, the length of the sliding window used to look for the local maxima and minima $b_{i,j}$, and the length of the trading horizon $c_{i,j}$. A support (resp. resistance) line is drawn joining at least two minimum (resp. maximum) price values. These parameters allow agents to operate to different time horizons.

If at time $c_{i,j}$ the price of the risky stock has not been matched the agent close its position and cancel the stop-profit order that was previously submitted. This is an interesting feature because, as is reported in [15] not all researchers in the experimental markets literature allow to cancel limit orders.

B. The classifier system

The behavior of the trader is determined by the classifier system they use to set their trade orders. The classifier system consists of a set of rules that are triggered when some market conditions are present. The classifier system implemented that follow our agents is based on the one used in the SFM, which is described in detail in [11].

The agents have to set of rules one for "going long" and other for "going short". The rules of both sets have the same structure, which consists of two parts. The left part of the rule is a string of 30 conditions, where each string position represents a state of the market. The possible values of each position are 1, 0 or $\#$. The 1 means that the state have to be

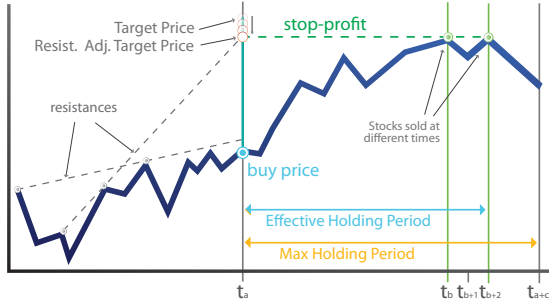


Fig. 1. Time line that illustrates an investment operation made by a trading agent

present, the zero that the state have not to be present while the $\#$ is a wildcard that matches either. The right part of the rule consists of three parameters ($a_{i,j}$, $b_{i,j}$ and $c_{i,j}$) that are used to draw the resistance and the support for the trade operation. If the agent is going long, it will use the resistance to set the stop-profit order and the support for the stop-loss order, while if the agent is going short it will do it the other way round. The idea is that each rule matches a state of the system, where the agent invest in the risky stock asset at market price. The position will be held until the asset reaches either the stop-profit or the stop loss prices. If the rule is matched at t_a , the agent expects to reach the stop profit price at $t_a + c_{i,j}$.

The parameters are initially set to random values distributed uniformly. As in the SFM the rules are not static. Each agent is allowed to learn by changing its set of rule. The learning process is implemented by means of a genetic algorithm where the poorly performing rules are substituted by new ones. Rules are selected for rejection and persistence based on a accuracy measure that takes into account both the errors in the price and in the forecast horizon.

C. The state of the agents

When a forecast rule j of agent i is activated in t_a , the agent will forecast the future stock returns for time horizon $t_a + c_{i,j}$, instead of $t_a + 1$ as in the SFM. Once the operation is done, the agent will hold its position until instant $t_a + c_{i,j}$ or until the price of the stock reaches a stop-profit or a stop-loss value at an undetermined time $t_b > t_a$, whatever comes first. In the second case, the operation may not be closed at that undetermined time, which will be denoted as t_b . It happens when there are not enough buy (resp. sell) orders to sell (resp. buy) all the stocks at the stop price in t_b . Figure 1 illustrates the process for the case where the agent goes long.

Once the transaction is completed (i.e. once the ordered stocks have been bought and then sold or viceversa for short positions), the investor's wealth has to be updated. As said before, the transaction is completed either when stocks reach the stop-profit price at t_b or when the holding time period is expired $t_a + c_j$. For the sake of simplicity, we will refer to these periods as t_e . The wealth equation must take into account that the sold of stocks can require more than one period. The

number of extra periods will be denoted as f . While agent traders can not execute several operations at the same time, and also can not modify their current trading strategy, wealth value is updated when a trading operation has concluded. Given that, the wealth of agent i in t_{e+f} is

$$W_{t_{e+f},i} = W_{t_a \rightarrow t_{e+f}}^{risky} + W_{t_a \rightarrow t_{e+f}}^{free} + W_{t_e \rightarrow t_{e+f}}^{free}. \quad (1)$$

The equations of these three terms are explained next.

The term $W_{t_a \rightarrow t_{e+f},i}^{risky}$ represents the changes from t_a to t_{e+f} in the wealth invested in the risky asset. Its equation is slightly different depending on the way stocks are sold. If they are sold because they reach the stop-profit price, then $t_e = t_b$ is the period when that price is reached and the wealth is

$$W_{t_a \rightarrow t_{e+f},i}^{risky} = p_{t_e} \sum_{l=0}^f x_{t_{e+l},i}^{out}, \quad (2)$$

where $x_{t_{e+l},i}^{out}$ is the number of stocks sold at time t_{e+l} by agent i and satisfying $\sum_{l=0}^f x_{t_{e+l},i}^{out} = x_{t_a,i}$.

On the other hand, if stocks are sold because the maximum holding time ends, which happens at $t_e = t_a + c_j$, then all the stocks are sold at that time (which means that $f = 0$). However, it may happen that not all the stocks are sold at the price p_{t_e} . This happens when the demand of the awaiting order does not cover the whole sell. If this is the case, other awaiting orders, possibly with a different price are required. The wealth $W_{t_a \rightarrow t_{e+f},i}^{risky}$ in this case is estimated as

$$W_{t_a \rightarrow t_{e+f},i}^{risky} = \sum_{l=1}^v x_{t_{e+l},i}^{out} p_{t_e,l}. \quad (3)$$

As all the stocks are sold at the same price, the price $p_{t_e,l}$ and the stocks number $x_{t_{e+l},i}^{out}$ both depend on a parameter $l = 1, \dots, v$ that represents the number of awaiting sell order matched.

The second term, $W_{t_a \rightarrow t_{e+f},i}^{free}$, represents the evolution of the capital invested in the free risk asset since t_a . It is given by

$$W_{t_a \rightarrow t_{e+f},i}^{free} = (1+r)^{t_{b+f}-t_a} (W_{t_a,i} - \sum_{l=1}^k p_{t_{a+l},i} x_{t_{a+l},i}^{in}), \quad (4)$$

where r is the constant interest rate of the free-risk asset and the pair of values $(p_{t_{a+l},i}, x_{t_{a+l},i}^{in})$ represents the awaiting sell order matched at time t with the market price order. It means that $x_{t_{a+l},i}^{in}$ stocks have been bought at price $p_{t_{a+l},i}$ with $\sum_{l=1}^k x_{t_{a+l},i}^{in} = x_{t_a,i}$.

Finally, the term $W_{t_e \rightarrow t_{e+f},i}^{free*}$ represents the evolution of the capital of the stocks sold between t_e and t_{e+f} , because during this period this capital is invested in the free risk asset. Thus, the term is only taken into account if $f > 0$. It is given by

$$W_{t_e \rightarrow t_{e+f},i}^{free*} = \sum_{l=1}^f (1+r)^{t_{b+f}-t_{b+l}} (p_{t_{b+l},i} x_{t_{b+l},i}^{out}). \quad (5)$$

In our market as in the SFM, investors use a simply constant relative risk aversion preference for stock demands [16]. When

a classifier detects an investment opportunity, the investor agent estimates the asset demands, trying to maximize the wealth utility function $U_{i,t+c_j} = -\exp(-\lambda W_{i,t+c_j})$, where the wealth equation has been described above.

III. VALIDATION OF THE MODEL

This paper represents the current state of a work-in-progress. An example of a candlestick time series with the trading volume generated by the ASM is shown in Figure 2. The model is being satisfactorily programmed using Nvidia CUDA technology, which allows to drastically reduce the simulation time. This parallel programming technology also allows to scale the number of agents without increasing the simulation time. Detailed information about trading strategies, agent behavior and evolution, the rule system and auction mechanism will be explained in the subsequent extended paper along with the simulation results and validation.

Regarding validation it will consist of analyzing two issues. First the usual econometric properties, i.e. the *stylized-facts*, that are present in real-life stock market time series such as fat tails and leptokurtic properties in return distributions, excess of kurtosis, no significant autocorrelation and volatility clusters. Second, it will be shown that technical patterns, familiar to professional technical analysts, do appear in the time series of the prices as a result of the self-fulfilling prophecy effect.

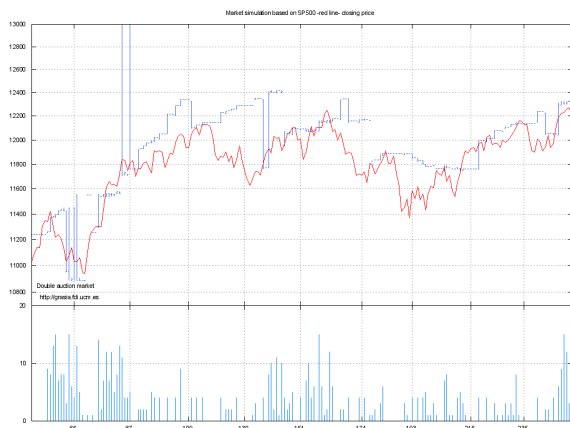


Fig. 2. Preliminary results: synthetic market (blue bars) following SP500 daily closing price (red line) as reference market.

ACKNOWLEDGMENT

The authors acknowledge support from the project *Agent-based Modelling and Simulation of Complex Social Systems (SiCoSSys)*, supported by Spanish Council for Science and Innovation, with grant TIN2008-06464-C03-01. We also thank the support from the *Programa de Creación y Consolidación de Grupos de Investigación UCM-BSCH, GR58/08*

REFERENCES

[1] B. LeBaron, *The Handbook of Computational Economics, Vol. 2: Agent-Based Computational Economics*, ser. Handbooks in Economics Series. Amsterdam: North-Holland, 2006, ch. Agent-based Computational Finance, pp. 1187–1234.

[2] M. Raberto, S. Cincotti, S. M. Focardi, and M. Marchesi, “Agent-based simulation of a financial market,” arXiv.org, Quantitative Finance Papers cond-mat/0103600, Mar. 2001. [Online]. Available: <http://ideas.repec.org/p/arx/papers/cond-mat-0103600.html>

[3] S. Cincotti, S. Focardi, L. Ponta, M. Raberto, and E. Scalas, “The waiting-time distribution of trading activity in a double auction artificial financial market,” vol. 567, pp. 239–247, 2006.

[4] J. Andersen and D. Sornette, “The d-game,” *European Physics Journal*, vol. 31, no. 1, pp. 141–145, 2003.

[5] N. Ehrentreich, *Agent-Based Modeling: the Santa Fe Institute Artificial Stock Market Model Revisited*. Springer-Verlag, 2008.

[6] L. Menkhoff, “Examining the use of technical currency analysis,” *International Journal of Finance and Economics*, vol. 2, pp. 307–318, 1997.

[7] M. P. Taylor and H. Allen, “The use of technical analysis in the foreign exchange market,” *Journal of International Money and Finance*, vol. 11, no. 3, pp. 304–314, June 1992.

[8] C. Hommes, “Heterogeneous agent models in economics and finance,” in *Handbook of Computational Economics*, 1st ed., L. Tesfatsion and K. L. Judd, Eds. Elsevier, 2006, vol. 2, ch. 23, pp. 1109–1186.

[9] J. Frankel and K. Froot, *Private Behaviour and Government Policy in Interdependent Economies*. New York: Oxford University Press, 1990, ch. Chartists, fundamentalists and the demand for dollars, pp. 73–126.

[10] B. LeBaron, W. B. Arthur, and R. Palmer, “Time series properties of an artificial stock market,” *Journal of Economic Dynamics and Control*, vol. 23, no. 9-10, pp. 1487–1516, September 1999.

[11] B. LeBaron, “Building the santa fe artificial stock market,” Brandeis University, Working paper, June 2002.

[12] N. T. Chan, B. LeBaron, A. W. Lo, and T. Poggio, “Agent-based models of financial markets: A comparison with experimental markets,” MIT Sloan, Working Paper 4195-01, 2001.

[13] D. K. Gode and S. Sunder, “Double auction dynamics: structural effects of non-binding price controls,” *Journal of Economic Dynamics & Control*, vol. 28, no. 9, pp. 1707–1731, 2004.

[14] J. Derveeuw, B. Beaufile, P. Mathieu, and O. Brandouy, “Testing double auction as a component within a generic market model architecture,” vol. 599, pp. 47–61, 2007.

[15] S. Crowley and O. Sade, “Does the option to cancel an order in a double auction market matter?” *Economics Letters*, vol. 83, no. 1, pp. 89 – 97, 2004.

[16] S. J. Grossman and J. Stiglitz, “On the impossibility of informationally efficient markets,” *American Economic Review*, vol. 70, no. 3, pp. 393–408, 1980.

BDI Agents with Fuzzy Perception for Simulating Decision Making in Environments with Imperfect Information

Giovani P. Farias, Graçaliz P. Dimuro and Antônio C. Rocha Costa
PPGMC - C3, Universidade Federal do Rio Grande
96201-900 Rio Grande, RS, Brazil, Email: {giovani.farias, gracaliz, ac.rocha.costa}@gmail.com

Abstract—This work introduces a model of fuzzy perception for BDI agents, to support the simulation of decision making processes in environments with imperfect information. An application to a fuzzy prey-predator environment was developed, as an example, where the process of deciding which prey a predator should attack is based on its fuzzy perception of the strength of the prey, and on the comparison of the preys' strengths with its own strength. Different simulations were realized for the comparative evaluation of different types of predator agents, in contexts with and without competition between predators. The quantitative analysis of the simulations shows that the fuzzy predator agent presents the best scores. However, the important result is that the fuzzy predator seems to behave more adequately in the environment, in the sense that it presents an apparently more natural, coherent and realistic behavior.

I. INTRODUCTION

Fuzzy sets and Fuzzy Logic (FL) [1] may be viewed as an attempt to formalize/mechanize two kinds of human capabilities. The first one is the capability to reason and make rational decisions in an environment of *imperfect information* (i.e. of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility). And second, the capability to perform a wide variety of physical and mental tasks without any measurements and any computations [2].

Zadeh [3] pointed out the *Incompatibility Principle*, which states that “complexity and precision are incompatible properties”, arguing that the conventional numerical-based approaches are inadequate to model human-like complex processes. Therefore, “the closer one looks at a real-world problem, the fuzzier becomes its solution”.

In the context of Social Simulation (SS), Grüne-Yanoff [4] and Rossiter et al. [5] remarked that one often has to deal with “fuzzy” social concepts, which are difficult to formalize and observe in the real-world system. For that reason, FL has been used in SS for representing vagueness, uncertainty and subjectiveness in everyday life.

Among the agent models commonly used in agent-based simulation of decision processes in complex environments, there are the ones of an intentional nature, whose behaviors can be explained by attributing certain mental attitudes to the agents, such as knowledge, belief, desire, intention, obligation, commitment (see, e.g., [6], [7]).

A well-known intentional model is the BDI (Beliefs, Desires and Intentions) architecture, introduced by Rao and Georgeff [8]. This model is based on the representation of the agent's *beliefs* about the states of the world and a set of *desires*, which identify those states that the agent has as goals. By a process of deliberation, the agent formulates one or more *intentions* (the states which the agent is committed to bringing about). The agent then builds a plan to achieve those intentions (through some form of means-ends reasoning), and executes it. After that, the agent uses its perception about the environment (which may include itself) in order to have its beliefs updated.

Although Rao and Georgeff explicitly acknowledge that an agent's model of the world is incomplete, the BDI model does not take into account the influence of the imperfect information (in the sense discussed above) acquired from the world in beliefs, desires and intentions. In particular, it does not consider that the agent could have a “fuzzy” perception of the world. Then, in this paper, we experiment with a BDI agent with *fuzzy perception* operating in a task environment with imperfect information, namely, a fuzzy prey-predator system.

Prey-Predator systems are an important theme in the area of Population Dynamics, their modeling having achieved a classical status through the formulation of the so-called Lotka-Volterra equations [9]. The particular type of systems that we simulate was inspired by the Fuzzy Prey-Predator Model introduced by Peixoto et al. [10].

The paper is organized as follows. Section II discusses related work. Section III presents some concepts on the fuzzy inference system used in this work. The environment with imperfect information inspired by the Fuzzy Prey-Predator Model is introduced in Sect. IV, including our approach for the fuzzy perception module to be included in the BDI architecture of the predator agent. The results on simulations are discussed in Sect. V. Section VI is the Conclusion.

II. RELATED WORK

In the context of social simulation, FL has been playing an important role, and it is possible to find many interesting works using FL to deal with different problems that can not be solved with classical simulation models and tools. In this section, we briefly present some of those works, according to the different issues covered by them.

Hassan et al. [11] observed that simple agent models, as those normally used with existing tools, are neither sufficient nor adequate to deal with the uncertainty and subjectiveness that have to be considered in the analysis of *values* (e.g., *trust*) in human societies. In their agent-based social modeling and simulation, FL was used to naturally specify attributes of the agents representing individuals, the evolution of the agent minds, the inheritance, the relationship and similarity between individuals, etc. In the same direction, in [12], fuzzy filters were used for modeling *trust* in social modeling using multiagent systems.

In Ghassem-Aghaee and Ören [13], human *personality* facets and traits (according to the Big Five and OCEAN models) were specified as conditional rules in fuzzy agents, in order to perform human behavior simulation. With related objectives, Dimuro et al. [14] introduced an approach based on FL for the evaluation of the social exchange values generated in the simulation of social exchanges between *personality*-based agents, with the analysis of the fuzzy equilibrium equation.

Sabeur and Denis [15] presented an application of FL in the simulation of human behavior and social networks, representing *behavioral* elements, such as stress, motivation or fatigue, and *sociological* aspects. Hassan et al. [16] use a fuzzy system to model friendship dynamics with an agent-based model that could manage social *relationships*, together with demographics and evolutionary crossover.

Fort and Pérez [17] used FL to model the adaptive behaviour of the agents playing the Iterated Prisoner's Dilemma, governed by Pavlovian strategies, to analyze the evolution of *co-operation*. Sabater et al. [18] proposed a fuzzy representation of evaluations for the system Repage, which adopts a cognitive theory of *reputation*.

Concerning *fuzzy perception* in robots, Cuesta and Ollero [19] used it to improve robot's navigation, and Mobahi and Ansari [20] applied fuzzy perception to improve the credibility in robot's emotions.

Notice that the agent architectures proposed so far mostly deal with two-valued information. Casali et al. [21], however, incorporated a formal model to represent and reason under uncertainty, introducing a general model for *graded BDI agents*, and an architecture, based on multi-context systems, able to model these graded mental attitudes. In [22], the model was used to specify an architecture for a travel assistant agent that helps a tourist to choose holiday packages, and in [23] it was applied to build a recommender system for tourism.

Hybrid models can be found in the literature, introducing some kind of fuzziness to *BDI* architecture. Long and Esterline [24] introduced a BDI agent, which uses fuzzy inference engines, fuzzy controllers and classifiers, for the modeling of co-operative societies of artificial agents, outlining some social conditions necessary for agents to form joint intentions and actions. Lokuge and D. Alahakoon [25] introduced a BDI agent coupled with a neural network and an adaptive neuro fuzzy inference system for application in container terminal operations, allowing the improvement the decision making process in such a complex, dynamic environment. A BDI

agent with a fuzzy neural network was also used by Hai-bo et al. [26] for application in autonomous underwater vehicles. Shen et al. [27] have explored a hybrid BDI model based on deliberative and fuzzy reasoning, and in [28] the model was improved within the context of wireless sensor networks.

However, neither the nice formalization by Casali et al. nor the other analyzed works have considered the influence of fuzzy perception on the operation of a BDI agent and its decision making.

III. ON FUZZY INFERENCE SYSTEMS

Fuzzy set theory [1], [2], [3] is based on the idea that several elements in human thinking are not exact data, but can be approximated as classes of objects in which the transition from membership to nonmembership is gradual rather than abrupt, represented by membership grades in the interval $[0; 1]$. Since human reasoning sometimes does not follow the two-valued or multivalued logic, FL is a logic with fuzzy truths, fuzzy connectives, and fuzzy rules of inference.

Fuzzy inference systems are non-linear models that aim to describe the input-output relationship of a real system using a family of linguistic **If-then** constructions and the inference mechanisms of FL. Among the several methods available for fuzzy inference, we adopt in this work the *Kang-Takagi-Sugeno* (KTS) method [29], where each fuzzy rule represents a local model of the real system under consideration¹. The k_{th} rule of a KTS system with input vector $X = (x_1, \dots, x_N)$ and output z presents the general form:

$$\begin{aligned} &\text{If} \quad (x_1 \text{ is } A_{1,k}) \text{ and } \dots \text{ and } (x_N \text{ is } A_{N,k}) \\ &\text{then} \quad z = f_k(X), \end{aligned} \quad (1)$$

where the linguistic terms $A_{n,k}$ ($n = 1, \dots, N$) in the rule antecedents represent fuzzy sets with membership functions $\mu_{n,k}$, which are used to partition the domains of the input variables into overlapping regions. The functions f_k in the rule consequents are usually first-order polynomials of the form:

$$f_k(x_1, \dots, x_N) = b_{0,k} + b_{1,k}x_1 + \dots + b_{N,k}x_N. \quad (2)$$

For a given input $X = (x_1, \dots, x_N)$, the degree of fulfillment of the k_{th} rule evaluates the compatibility of the input X with the rule antecedent and determines the contribution of the rule's response $z = f_k(x_1, \dots, x_N)$ to the overall model's output. The degree of firing of the k_{th} rule is expressed as

$$w_k(x_1, \dots, x_N) = T_1(\mu_{A_{1,k}}(x_1), \dots, \mu_{A_{N,k}}(x_N)), \quad (3)$$

where T_1 is a t-norm (triangular norm). In this work, T_1 is the *Minimum* t-norm (called Gödel t-norm), and then Eq. 3 becomes :

$$w_k(x_1, \dots, x_N) = \min\{\mu_{A_{1,k}}(x_1), \dots, \mu_{A_{N,k}}(x_N)\}. \quad (4)$$

The overall output of a normalized first-order TSK fuzzy model with K rules is given by

$$z = \frac{\sum_{k=1}^K T_2(w_k(x_1, \dots, x_N), f_k(x_1, x_2, \dots, x_N))}{\sum_{k=1}^K w_k(x_1, \dots, x_N)}, \quad (5)$$

¹The adoption of the KTS method is due to its better performance in some applications, since it avoids the defuzzification step. See [29] for details.

where T_2 is also a t-norm. In this work, T_2 is the *Product* t-norm, so that Eq. 5 results in:

$$z = \frac{\sum_{k=1}^K w_k(x_1, \dots, x_N) \cdot f_k(x_1, x_2, \dots, x_N)}{\sum_{k=1}^K w_k(x_1, \dots, x_N)}. \quad (6)$$

IV. A FUZZY PREY-PREDATOR ENVIRONMENT

In [10], Peixoto et al. proposed a fuzzy rule-based system to elaborate a predator-prey model to study the interaction between aphids (preys) and ladybugs (predators) in citriculture. Due to the lack of available information about the phenomenon, instead of using the usual differential equations that characterize the classic deterministic models, they introduced a fuzzy approach for analyzing the problem.

In this paper, we informally build on the fuzzy prey-predator approach for an agent-based simulation in order to analyze the ability of a predator with fuzzy perception in surviving in an environment of imperfect information.²

In this environment, the age and the weight of a prey (and of a fuzzy predator itself) are vague information for the fuzzy predator. However, such information is crucial for a predator to evaluate the strength level of a certain prey in comparison with its own strength level, and, therefore, to estimate the probability of the success of its attack to such prey, which is given by:

$$Prob = 50 + \frac{RAP - RPP}{200}, \quad (7)$$

where RAP and RPP are the predator's and the prey's strength levels, respectively.

We assume that (i) predators and preys are initially randomly distributed in a grid; (ii) the food is always available for the different preys, and (iii) a predator loses weight for being looking around for preys and much more for each unsuccessful attack (on the contrary, it gains weight if its attack is successful). Then, the predator survival during the evolution of the time depends on its decision about attacking or not any prey it finds during its life. This decision is based on the imperfect information that the agent can perceive through its fuzzy perception mechanism, which uses a fuzzy inference system to determine the prey's strength level and its own.

The predator is a BDI agent with beliefs³ on the following parameters: age, weight and strength level. The age and weight the agent can perceive through its perception mechanism. The strength level can be estimated considering perceived ages and weights. The abilities of the predator are: random movement looking for preys, perception of preys's age and weight, estimation of prey's strength level in comparison with its own strength level at the current time, and decision on attacks to preys, which considers if the probability of success satisfies $Prob > 0.25$ (Eq. 7). The constraints of its life are:

²Notice that we did not study population dynamics, as it was done in [10], although this can be considered in future work.

³In this paper, we do not refer to the agent's desires or intentions, only to its beliefs, since this is the component of the BDI model that is connected to the fuzzy perception mechanism.

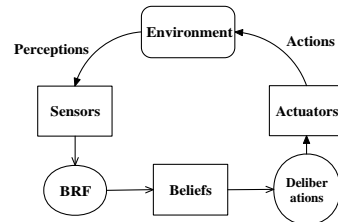


Fig. 1. Part of the BDI model with a fuzzy perception module.

- (i) at each movement it loses a fixed amount of weight (*weight loss rate*), and has its age incremented by a fixed value (*aging rate*);
- (ii) at each successful attack, it gains a fixed amount of weight (*attack reward*); otherwise, it loses a fixed amount of weight (*attack punishment*);
- (iii) there is a minimum weight that a predator can support; if it achieves a weight less than the minimum then it dies by *weakness*;
- (iv) there is a maximum age that a predator can achieve; after that it dies by ageing.

A. Characterizing the Fuzzy Predator (FP) Agent

The Fuzzy Predator (FP) has a perception mechanism directly connected to the BRF (Belief Revision Function) of its BDI architecture, partially depicted in Fig. 1. This means that the fuzzy perception mechanism receives as input data the prey's age and weight, as well as the predator's own age and weight, all of which are perceived through the predator's non precise sensors. Then, using the KTS inference system (Sect. III), the predator infers the prey's strength level, and also its own strength level, updating its beliefs with the inferred information, in order to let this information be used in the decision process.

The linguistic variables *age*, *weight* and *strength level* are modeled as fuzzy sets with trapezoidal membership functions (Fig. 2). The analysis of those linguistic variables allowed the construction of a knowledge base composed by the linguistic rules presented in part in Table I. Table II shows part of the rule base for the KTS inference system of the perception model of the FP agent, each one with 2 inputs $(age, weight) \in \mathbb{R}^2$ and the output $z \in \mathbb{R}$, where “*young*”, “*adult*”, “*old*”, “*very light*”, “*light*”, “*average*”, “*heavy*” e “*very heavy*” represent fuzzy subsets of \mathbb{R} .

Example 1: In order to see how the inference system of the fuzzy perception mechanism operates, let us consider the following crisp input data: $age = 16$ and $weight = 84$. Those values are fuzzified, considering the membership grades in relation to the fuzzy subsets that define those linguistic variables, given in Fig. 2. Then, the age value $age = 16$ is considered “*young*” with grade $\mu_{young}(16) = 0,4$ and “*adult*” with grade $\mu_{adult}(16) = 0,6$. The weight value $weight = 84$ is evaluated as “*heavy*” with grade $\mu_{heavy}(84) = 0,6$ and “*very heavy*” with grade $\mu_{very-heavy}(84) = 0,4$.

For each combination of those sets achieved by the input data, some of the rules of the knowledge base are activated. In this case, four rules are fired, namely, the rules R_4 , R_5 , R_9 and R_{10} of the Tables I and II. Using Eq. 4, it is possible to

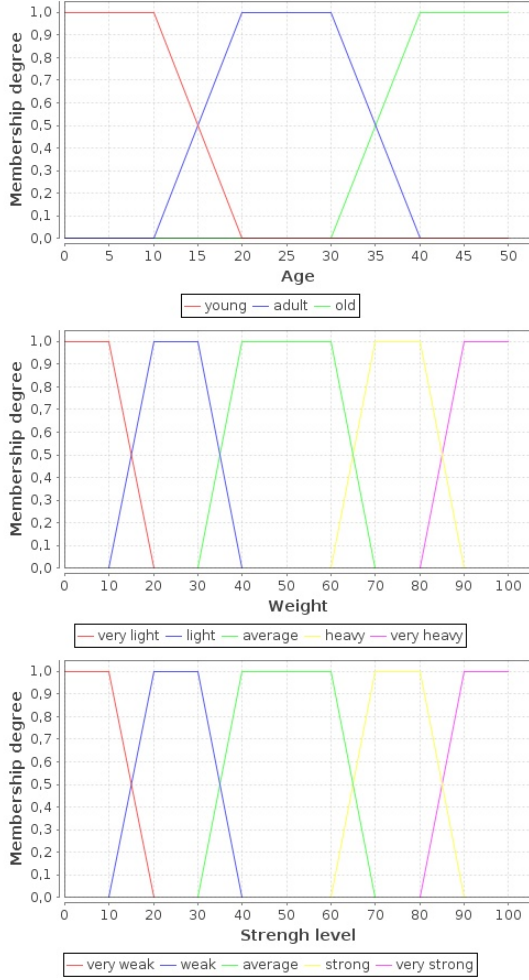


Fig. 2. Membership function for the considered linguistic variables.

find the degrees of firing of each one of those rules, as, e.g., $w_4 = \min\{\mu_{young}(16), \mu_{heavy}(84)\} = 0,4$. Then, one has that $w_5 = 0,4$, $w_9 = 0,6$ and $w_{10} = 0,4$. Using Eq. 6, we obtain the overall output of the process, where f_4 , f_5 , f_9 and f_{10} are calculated using Table II:

$$z = \frac{w_4 f_4(16, 84) + w_5 f_5(16, 84) + w_9 f_9(16, 84) + w_{10} f_{10}(16, 84)}{w_4 + w_5 + w_9 + w_{10}} = 74,$$

which represents the predator's strength level.

B. The Crisp Predator (CP)

For the comparative analysis of simulations, we implemented a Crisp Predator (CP), which is a BDI agent that does not consider that the information about itself and the one perceived from the environment are vague or incomplete. Its perception mechanism is inspired on the perception mechanism of the fuzzy predator, but, instead of using fuzzy subsets for the modeling of the input linguistic variables, we use classical sets with the usual characteristic functions into the set $\{0,1\}$. For each set of input data, only one rule of the

TABLE I
LINGUISTIC RULE BASE.

If	age	and	weight	then	strength level
R_1	young		very light		very weak
R_2	young		light		very weak
R_3	young		average		weak
R_4	young		heavy		average
R_5	young		very heavy		average
R_6	adult		very light		average
R_7	adult		light		average
R_8	adult		average		strong
R_9	adult		heavy		very strong
R_{10}	adult		very heavy		very strong
R_{11}	old		very light		very weak
R_{12}	old		light		very weak
R_{13}	old		average		weak
R_{15}	old		very heavy		average

knowledge base is activated. The characteristic functions of the sets related to the linguistic variable `age` are:

$$\mu_{young}(x) = \begin{cases} 1 & \text{if } x \leq 15; \\ 0 & \text{otherwise} \end{cases} \quad \mu_{adult}(x) = \begin{cases} 1 & \text{if } 15 < x < 35; \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{old}(x) = \begin{cases} 1 & \text{if } x \geq 35; \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

The characteristic functions of the sets related to the linguistic variable `weight` are:

$$\mu_{very-light}(x) = \begin{cases} 1 & \text{if } x \leq 15; \\ 0 & \text{otherwise} \end{cases} \quad \mu_{light}(x) = \begin{cases} 1 & \text{if } 15 < x \leq 35; \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{average}(x) = \begin{cases} 1 & \text{if } 35 < x \leq 65; \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$\mu_{heavy}(x) = \begin{cases} 1 & \text{if } 65 < x \leq 85 \\ 0 & \text{otherwise} \end{cases} \quad \mu_{very-heavy}(x) = \begin{cases} 1 & \text{if } x > 85 \\ 0 & \text{otherwise} \end{cases}$$

Example 2: Considering the same input data (`age`, `weight`) of Ex. 1 and the characteristic functions given in Equations 8 and 9, one has that `weight` = 84 and `age` = 16 are definitely evaluated as “heavy” ($\mu_{heavy}(84) = 1$) and “adult” ($\mu_{adult}(16) = 1$), respectively. In this case, only the rule R_9 of the rule base of Tables I and II is activated. Obviously, the firing degree of this rule is $w_9 = 1$. The general output, given by Eq. 6, results in the value of the straight level:

$$z = \frac{w_9 f_9(16, 84)}{w_9} = \frac{1 \cdot 88}{1} = 88$$

To enrich the possible comparisons, we have implemented a Greedy Predator (GP), which always attacks the preys it encounters, without considering any reasoning on strength levels and the probability of success of its attacks to preys.

V. ANALYSIS OF THE SIMULATION RESULTS

The simulations were realized to obtain a general view of the behaviors of the different predators⁴ in two kinds of the Fuzzy Prey-Predator Environment: competitive (Sect. V-A) and non-competitive environments (Sect. V-B). The implementation was done in the Jason platform [30].

⁴Since we are not analyzing population behavior, in the simulations we only consider either 2 or 3 predators, in order to be able to compare directly their surviving abilities.

TABLE II
RULE BASE FOR THE KTS INFERENCE SYSTEM.

If age and weight then strength level = $f_k(\text{age}, \text{weight})$	
R_1	young very light $f_1(x, y) = \frac{x+y}{2}$
R_2	young light $f_2(x, y) = \frac{x + (\frac{y}{2})}{2}$
R_3	young average $f_3(x, y) = \frac{(x+y) - 10}{2}$
R_4	young heavy $f_4(x, y) = (x-1) + \frac{y}{2}$
R_5	young very heavy $f_5(x, y) = x + \frac{y}{2}$
R_6	adult very light $f_6(x, y) = \frac{x+y}{2} + 25$
R_7	adult light $f_7(x, y) = \frac{x+y}{2} + 30$
R_8	adult average $f_8(x, y) = \frac{x + \frac{y}{2} + 100}{2}$
R_9	adult heavy $f_9(x, y) = \frac{x+y}{4} + 63$
R_{10}	adult very heavy $f_{10}(x, y) = \frac{\frac{x}{2} + y}{2} + 40$
R_{11}	old very light $f_{11}(x, y) = \frac{(50-x) + y}{2}$
R_{12}	old light $f_{12}(x, y) = \frac{(50-x) + \frac{y}{2}}{2}$
R_{13}	old average $f_{13}(x, y) = \frac{(50-x) + (y-10)}{2}$
R_{15}	old very heavy $f_{15}(x, y) = (50-x) + \frac{y}{2}$

The results were obtained from a total of 100 simulation runs. In each run, the time grows in discrete units (1 time unit = one predator movement). In the beginning of each run, the predators present the following initial parameters: $\text{age} = 1$ and $\text{weight} = 50$. Those parameters change at each time instant according to the following fixed rates⁵: the *weight loss rate* (-0,1 kg for each movement/time), the *aging rate* (-0.05 year for each movement/time), the *attach reward* (+2 kg for each successful attack), and the *attack punishment* (-1 kg for each non successful attack). The simulation run ends when all the predators have died, either for weakness (weight less than 1 kg) or for ageing (age equal to 50 years).

A. The Competitive Environment

The competitive environment consists of 2 kinds of predators (FP e CP) and different 250 preys. At each successful predator attack, the corresponding defeated prey dies. Considering that there is no prey reproduction, the prey population tends to decrease, increasing the probability of the predator not finding a prey as it moves in the environment, which may cause increasing weight losses. In this sense, both predators compete for the preys remaining in the environment.

⁵Variations of the initial parameters and rates are not considered here, since they affect only the agent's deliberations, not its perceptions.

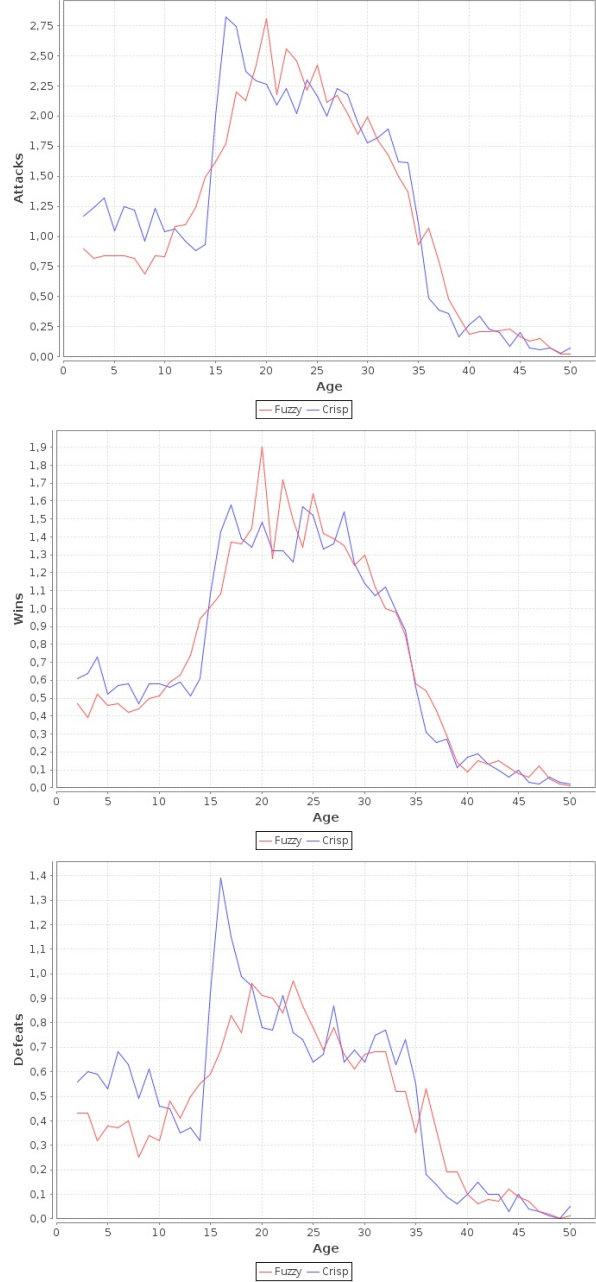


Fig. 3. The average of attacks (top), victories (middle) and defeats (bottom) at an age i , with $1 \leq i \leq 50$, in a competitive environment.

Figure 3 (top) shows the average number of predators' attacks at each year. Observe that the number of the CP's attacks surges around the age of 15. This is so because, before 15, the agent thinks that it is young (with too low strength level), but, suddenly, as it achieves 15 years old, it concludes that it is already an adult (with too high strength level). The increase in the number of the FP's attacks is more gradual,

showing more coherence in its decisions. On the other hand, one might have expected that the high number of attacks would have lasted until around the age of 35, since it is only after this age that the CP considers itself old. However, due the prey population decreasing, the number of the attacks of both predators also decreases, even before the age 35. Around the age 35, the decrease in the number of the attacks of the CP is much more abrupt than the smooth decreasing of the number of the attacks of the FP, as it passes from young to adult/old.

Figure 3 (middle) presents the average of predators' victories at each year. There is a significant increase in the number of victories when the CP is around 15, which is an expected result, since this is the period that, as it considers itself an adult by this age, it increases a lot the number of attacks until around the age of 35, when it considers itself old, as discussed in the previous paragraph. Also, due to the decrease in the prey population, and consequently, the decrease in the number of attacks, the number of victories also decreases, even before the age 35. Again, it is possible to observe that the graph corresponding to the FP increases and decreases smoothly, as the agent becomes old, whereas the one of the CP increases abruptly around 15 and decreases around 35, also drastically.

Analogous analysis can be done concerning the average of number of the predators' defeats at each year, which is shown in Fig. 3 (bottom).

B. The Non-competitive Environment

The non-competitive environment consists of 3 kinds of predators (FP, CP and GP), and 250 different preys. For each prey that dies in consequence of a predator attack, another prey with similar characteristics appears in the environment, at a random position. This means that the predators always have the same chance to find a prey to attack.

Figure 4 (top) shows the average number of predators' attacks at each year. For the same reasons discussed in Sect. V-A, the number of the CP's attacks surges around the age of 15. However, since the prey population is constant along the time, the high number of attacks of the CP lasts until around the age of 35, and then it follows drastically. The behavior of the FP is much more natural and coherent, since it presents a gradual increase in the number of attacks as it becomes an adult, and a also a smooth decrease as it becomes old. The high number of attacks of the GP during its life was as expected. During adulthood the numbers of attacks of all predators are similar.

Figure 4 (middle) presents the average number of predators' victories at each year. There is an abrupt increase in the number of victories when the CP is around 15, due to the high increase in the number of its attacks by this age. However, since the prey population does not decrease, the number of victories stays high until around the age 35. After that, it decreases radically. Again, it is possible to observe that the graph corresponding to the FP increases and decreases smoothly, as the agent becomes old. The higher number of victories of the GP is due to its attack strategy. During adulthood, the numbers of victories of the three kinds of

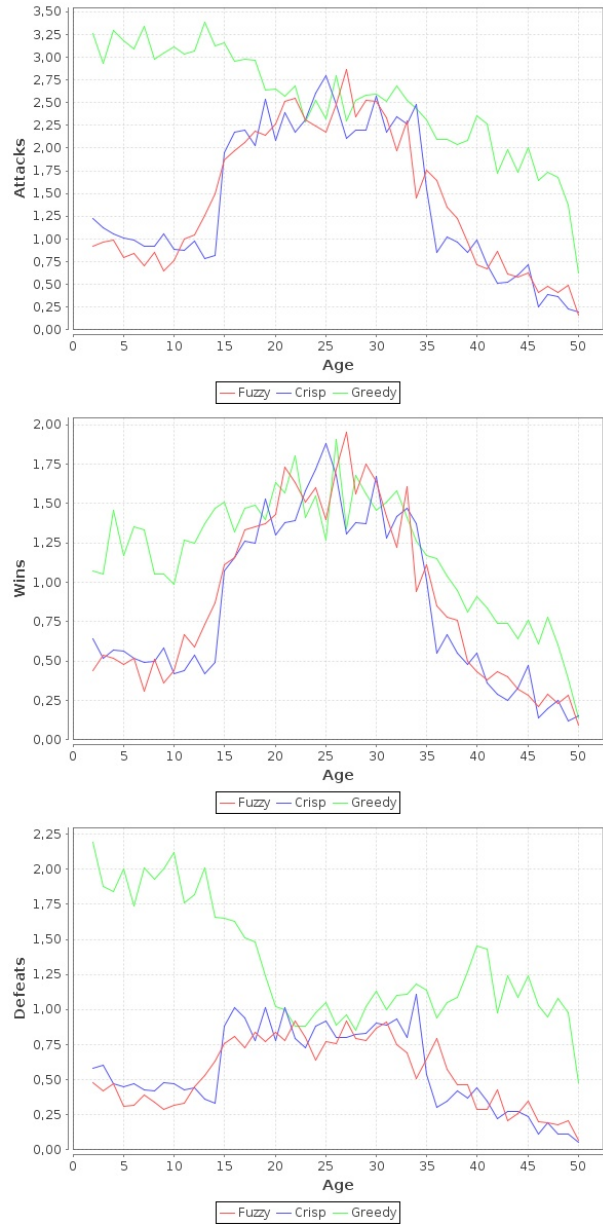


Fig. 4. The average of attacks (top), victories (middle) and defeats (bottom) at an age i , with $1 \leq i \leq 50$, in a non-competitive environment.

predators are similar. The highest numbers of victories, for Crisp and Fuzzy predators, appear between the ages of 20 and 33. Analogous analysis can be done concerning the average of number of predators' defeats at each year (Fig. 4 (bottom)).

Figure 5 (top) shows the average number of accumulated attacks during the predators' lives, until they reach a certain age i , with $1 \leq i \leq 50$. As expected, the GP had an average number of accumulated attacks much higher than the other two predators, which had a similar attack behavior.

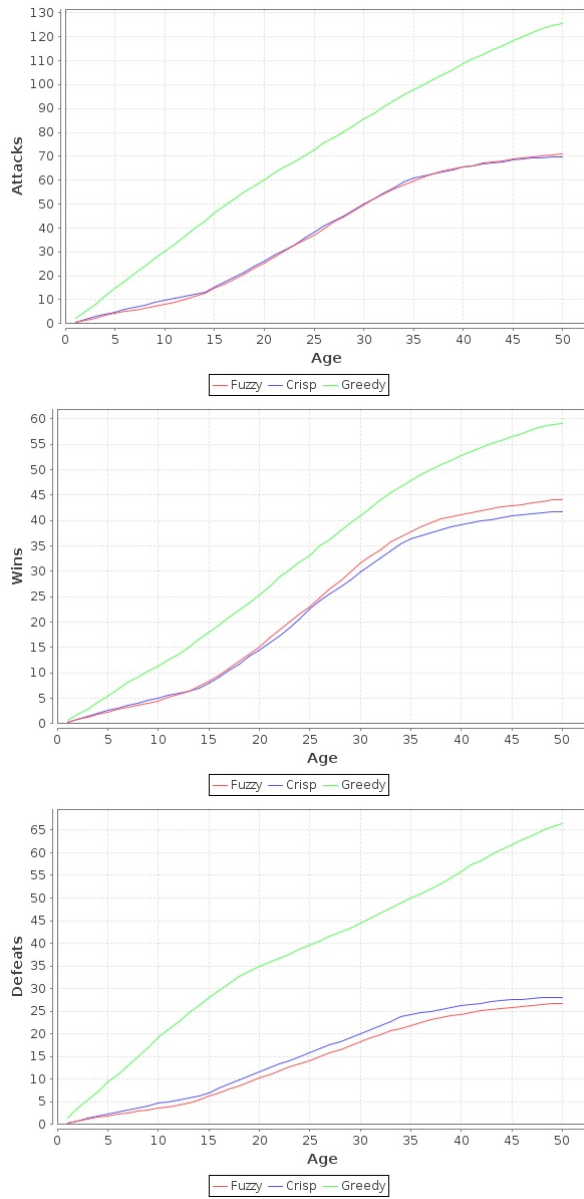


Fig. 5. Average accumulated number of attacks (top), victories (middle) and defeats (bottom) during the predator life until the age i , with $1 \leq i \leq 50$.

Figure 5 (middle) presents the average number of accumulated victories during the predators' lives, until they reach a specific age. As expected, the GP had an average number of accumulated victories much higher than the other two predators. However, this number for the FP is higher than that of the CP, as they become older.

Analogous analysis can be done concerning the average number of accumulated defeats during the predators' lives, until they reach a specific age, shown in Fig. 5 (bottom). Figure 6 (top) shows the average lifetime of the predators. As

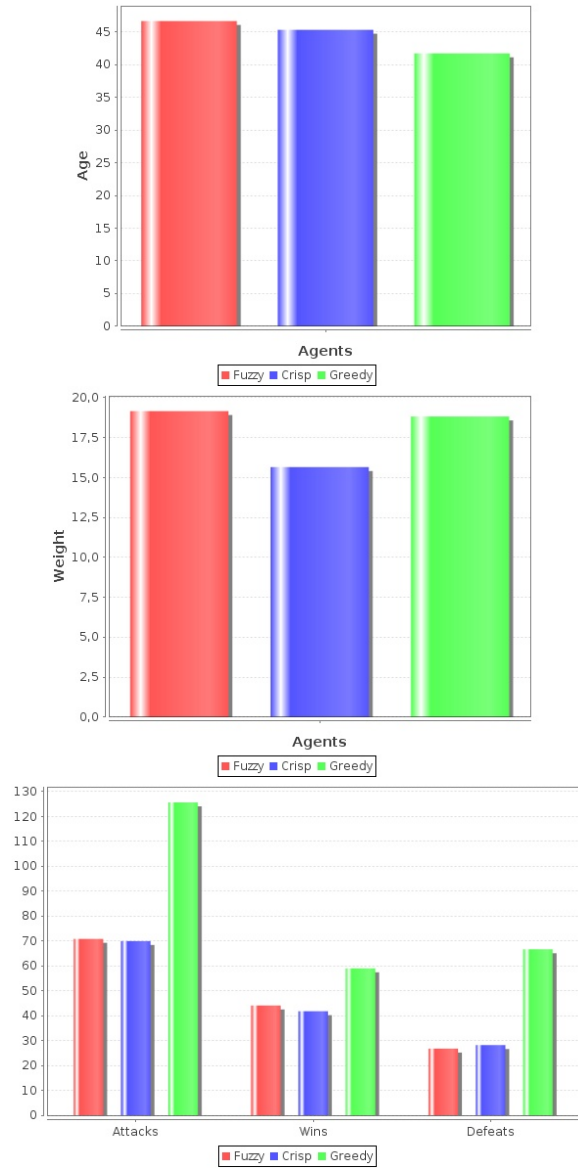


Fig. 6. Average lifetime (top), average weight at the end of the life (middle) and average of number of attacks/victories/defeats (bottom) of predators.

expected, the Greedy and Fuzzy predators present the lowest and the highest average lifetimes, respectively.

Figure 6 (middle) presents the average weight of predators at the end of their lives. The average weight of the FP at the end of its life is the highest one.

Figure 6 (bottom) shows the average number of attacks, victories and defeats of predators during its whole life. The GP is the one that presents the highest averages in all categories, and it is the one that has the average number of defeats higher than that of victories. Its average numbers of attacks and victories are higher than the ones of the CP, whereas the

average number of defeats is lower than that of the CP.

We conclude that the simulation of the fuzzy perception of the FP allowed for a more faithful simulation of naturally expected smoothness of the development of predation ability of predators in a Fuzzy Prey-Predator environment.

VI. CONCLUSION

This paper introduces a model of fuzzy perception for BDI agents in task environments with imperfect information, which was inspired by an analysis of a particular Fuzzy Prey-Predator model. The aim was to analyze the influence of fuzzy perception on the ability of BDI agents to simulate decision making processes in fuzzy environments. For that, we have defined a perception mechanism directly connected to the BDI agent's BRF. The perception mechanism uses a KTS inference system, which is application dependant.

The simulations allowed us to obtain a general view of the behaviors of the different predators (CP, FP, GP) in two kinds of the Fuzzy-Predator Environment: a competitive environment and a non-competitive environment.

Although the difference between the results obtained by the Fuzzy and the Crisp predator agents were not so significant in the quantitative analysis that we have performed, it seems that the Fuzzy predator agent showed a more adequate simulated behavior in the environment with imperfect information, presenting a more natural, coherent and realistic behavior than the other agents.

Finally, two issues on the obtained results are important to point out. Firstly, the BDI agent with fuzzy perception seems to be a good model to be used in agent-based simulations in environments with imperfect information. Secondly, a fuzzy perception module can be a good alternative solution in the design of a BDI agent that can not perceive the information of the environment with accuracy.

Future work will consider a fuzzy perception mechanism for a BDI agent that is more application independent. For that, we are considering the use of the Mamdani inference method [1] in the level of the agent plans, so that the fuzzification of the input data will be directly reflected in the agent's set of beliefs, then extending to account for fuzzy beliefs.

Acknowledgements. This work is part of a larger project (RS-SOC: Rede Estadual de Simulação Social), being run under the FAPERGS/CNPq/PRONEX (Proc. 10/0049-7) context, where fuzzy perception of social interactions are considered. It is also supported by CAPES and CNPq (Proc. 483257/09-5, 307185/07-9, 304580/07-4). We thank R. Bordini for valuable suggestions.

REFERENCES

- [1] L. A. Zadeh, "Fuzzy sets," *Inf. and Control*, vol. 8, pp. 338–353, 1965.
- [2] —, "Is there a need for fuzzy logic?" *Information Sciences*, vol. 178, no. 13, pp. 2751–2779, 2008.
- [3] —, "Outline of a new approach to the analysis of complex systems and decision processes," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-3, no. 1, pp. 28–44, 1973.
- [4] T. Grüne-Yanoff, "The explanatory potential of artificial societies," *Synthese*, vol. 169, no. 3, p. 3, 2006.
- [5] S. Rossiter, J. Noble, and K. R. W. Bell, "Social simulations: Improving interdisciplinary understanding of scientific positioning and validity," *JASSS*, vol. 13, no. 1, p. 10, 2010.
- [6] B. Subagdja, L. Sonenberg, and I. Rahwan, "Intentional learning agent architecture," *JAAMAS*, vol. 18, no. 3, pp. 417–470, 2009.
- [7] S. Sardina and L. Padgham, "A BDI agent programming language with failure handling, declarative goals, and planning," *JAAMAS*, 2010.
- [8] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a BDI-architecture," in *Proc. 2nd Intl. Conf. on Principles of Knowledge Representation and Reasoning*, R. Fikes and E. Sandewall, Eds. San Mateo: Morgan Kaufmann, 1991, pp. 473–484.
- [9] A. A. Berryman, "The origins and evolution of prey-predator theory," *Ecology*, vol. 73, no. 5, pp. 1520–1535, 1992.
- [10] M. S. Peixoto, L. C. Barros, and R. C. Bassanezi, "Predator-prey fuzzy model," *Ecological Modelling*, vol. 124, no. 1, pp. 39–44, 2008.
- [11] S. Hassan, L. Garmendia, and J. Pavón, "Agent-based social modeling and simulation with fuzzy sets," in *Innovations in Hybrid Intelligent Systems*, ser. Advances in Soft Computing, E. Corchado et al., Eds. Berlin: Springer, 2008, no. 44, pp. 40–47.
- [12] E. del Acebo and J. L. de la Rosa, "A fuzzy system based approach to social modeling in multi-agent systems," in *Proc. of Intl. Conf. on Autonomous Agents and Multiagent Systems*. ACM, 2002, pp. 463–464.
- [13] N. Ghasem-Aghaee and T. I. Ören, "Towards fuzzy agents with dynamic personality for human behavior simulation," in *Proc. of the 2003 Summer Computer Simulation Conference, Montreal, July 20-24, 2003*. San Diego: SCS, 2003, pp. 3–10.
- [14] G. P. Dimuro, A. V. Santos, G. P. Bedregal, and A. C. R. Costa, "Fuzzy evaluation of social exchanges between personality-based agents," in *New Trends In Artificial Intelligence*, L. S. Lopes et al., Eds. Aveiro: APIA, 2009, pp. 451–462.
- [15] E. Sabour and G. Denis, "Human behavior and social network simulation: Fuzzy sets/logic and agents-based approach," in *Proc. of the 2007 Spring Simulation Multi-Conference, Norfolk, 2007*. San Diego: SCS, 2007, pp. 102–109.
- [16] S. Hassan, M. Salgado, and J. Pavón, "Friends forever: Social relationships with a fuzzy agent-based model," in *Hybrid Artificial Intelligence Systems*, ser. LNCS. Berlin: Springer, 2008, no. 5271, pp. 523–532.
- [17] H. Fort and N. Pérez, "The fate of spatial dilemmas with different fuzzy measures of success," *JASSS*, vol. 8, no. 3, 2005.
- [18] J. Sabater-Mir, M. Paolucci, and R. Conte, "Repage: REputation and iMAGE among limited autonomous partners," *JASSS*, vol. 9, no. 2, pp. 539–555, 2009.
- [19] F. Cuesta and A. Ollero, "Intelligent control of mobile robots with fuzzy perception," in *Intelligent Mobile Robot Navigation*, ser. Springer Tracts in Advanced Robotics. Berlin: Springer, 2005, vol. 16, pp. 79–122.
- [20] H. Mobahi and S. Ansari, "Fuzzy perception, emotion and expression for interactive robots," in *Proc. of the IEEE Intl. Conf. on Systems, Man and Cybernetic (SMCC03)*, vol. 4, Washington, 2003, pp. 3918–3923.
- [21] A. Casali, L. Godo, and C. Sierra, "Graded BDI models for agent architectures," in *Computational Logic in Multiagent Systems*, ser. LNAI. Berlin: Springer, 2005, no. 3487, pp. 126–143.
- [22] —, "Modeling travel assistant agents: a graded BDI approach," in *Artificial Intelligence in Theory and Practice*, ser. IFIP. Berlin: Springer, 2006, vol. 217, pp. 415–424.
- [23] —, "g-BDI: A graded intensional agent model for practical reasoning," in *Modeling Decisions for Artificial Intelligence*, ser. LNCS. Berlin: Springer, 2009, no. 5861, pp. 5–20.
- [24] S. A. Long and A. C. Esterline, "Fuzzy BDI architecture for social agents," in *Proc. IEEE Southeastcon 2000*, N. R. Pal et al., Eds. Los Alamitos: IEEE, 2000, pp. 68–74.
- [25] P. Lokuge and D. Alahakoon, "Decisions based upon multiple values: the BVG agent architecture," in *Neural Information Processing*, ser. LNCS, N. R. Pal et al., Eds. Berlin: Springer, 2004, no. 3316, pp. 941–946.
- [26] L. Hai-bo, G. Guo-chang, S. Jing, and F. Yan, "AUV fuzzy neural BDI," *Marine Science and Application*, vol. 4, no. 3, pp. 37–41, 2007.
- [27] S. Shen, G. M. P. O'Hare, and R. Collier, "Decision-making of BDI agents, a fuzzy approach," in *Proc. 4th Intl. Conf. on Computer and Information Technology*. Washington: IEEE, 2004, pp. 1022 – 1027.
- [28] S. Shen, G. M. P. O'Hare, and M. J. O'Grady, "Fuzzy-set-based decision making through energy-aware and utility agents within wireless sensor networks," *Art. Intelligence Review*, vol. 27, no. 2-3, pp. 165–187, 2008.
- [29] Y. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.
- [30] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. New Jersey: Wiley, 2007.

When will I see you again: modelling the influence of social networks on social activities

Nicole Ronald
Design and Decision Support
Systems Group
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: n.a.ronald@tue.nl

Virginia Dignum
Faculty of Technology, Policy
and Management
Delft University of Technology
Delft, The Netherlands
Email: M.V.Dignum@tudelft.nl

Catholijn Jonker
Man-Machine Interaction Group
Delft University of Technology
Delft, The Netherlands
Email: C.M.Jonker@tudelft.nl

Abstract—Social activities account for a large amount of travel, yet due to their irregularity and the number of options regarding location, participants, and timing, they are difficult to model and predict. We assume that social activities are constrained by one’s social network, which consists of people you are close to, both socially and spatially. Therefore, a model of social activity behaviour should be sensitive to the network. In this paper, an agent-based model to describe social activities between two people over time is described and four different input networks (random, based on spatial distance, based on social distance, based on both distances) are experimented with. The results show that the overall social network has an effect on the number of activities generated in the entire system and also between pairs of friends.

I. INTRODUCTION

Every transport system may be described as a social system, composed of individuals who interact and influence the behaviour of each other. Multi-agent simulation is therefore becoming increasingly important in travel simulation, travel analysis, and travel forecasting, in particular due to its possibilities to model explicitly the individuals’ decision making processes. In fact, all travel is a result of individual decisions, as people try to manage his/her life in a satisfying way. As such, travel can be seen as result of individual goals (e.g. go to work to earn money, visit friends for pleasure) [1].

Our focus in this paper is on social face-to-face activities. People frequently interact face-to-face with each other. This could fulfill several needs: to gather information, to share an experience, to help one another, or for relaxation. Face-to-face interaction is sometimes also crucial for relationships to continue. Urry [2] notes that “[e]specially in order to sustain particular relationships with a friend or family or colleague that are ‘in the mind’, that person has intermittently to be seen, sensed, through physical copresence”.

In order to model these activities, the transport modelling field is experiencing a shift from understanding “where are people going” and “what activity are they doing” towards “who are they interacting with”. The generation and scheduling of social activities depends not only on the structure of the spatial network, which is covered by “where” and “what”, but requires that social networks, which mean “who” need to be incorporated as well.

In this project, we are interested in ascertaining the influence of social network typology on the number, frequency and type of social activities between network nodes. This is necessary because incorporating social networks into existing activity-travel models will add a lot of complexity and require more intensive data collections. Testing the sensitivity of potential models of activity behaviour to different networks is an important step in evaluating the usefulness of their incorporation.

The aim of this paper is to demonstrate the relevance of the social network structure, by investigating the performance of a simplified model with different input structures with respect to the number of activities generated for individuals, pairs of individuals, and for the entire population. We begin with a review of activity modelling and social network generation. A model with utility-based agents is described and the results are discussed. We conclude with recommendations for other applications and future work.

II. BACKGROUND AND RELATED WORK

A. Activity generation

Human activities are generated due to “physiological, psychological and economical needs” [3]. A distinction is commonly made between subsistence (e.g., work-related), maintenance (e.g., keeping the household running), and leisure activities.

Non-discretionary activities such as work and school can be partly explained by the traveller’s sociodemographic characteristics and generalised travel costs [4], as well as long-term decisions such as a decision to move to a particular town. Participation in, and scheduling of, other activities is not as easily predicted. Social and leisure activities are the reported purpose for a large number of trips, ranging from 25 to 40% for various countries [5].

In current state-of-the-art activity-travel models, social activities, if at all scheduled, are assigned to random locations and times [6] and do not take into account the constraints or preferences of friends. Being able to model these activities could lead to better prediction of activity schedules and forecasts of travel patterns and demand for urban facilities, in particular those relating to social and leisure activities.

A theory currently being explored for generating discretionary activities is based on needs. Activities both satisfy and generate needs and needs grow over time [7]. Maslow's hierarchy of needs has been proposed as a starting point [8], however it is difficult to collect data for model validation. A separate set of needs was proposed by Arentze and Timmermans [7] which could be identified through empirical research.

B. Social networks

Social networks are a representation of individuals and the relationships between them. The relationship between two individuals can be defined in a number of ways, for example how similar they are, how they are related to each other, whether they interact or how often they interact, or how information flows between them [9].

Networks can be represented in two ways: complete or personal. A complete network contains all of the relationships for all the individuals in the network, for example, all the friendship links between students in a class. Personal networks contain the relationships for a particular individual (known as the ego), however the attributes of the people they name (known as alters) are provided by the ego rather than the alter themselves. It is not guaranteed that the personal networks of egos in the sample will intersect.

As Newman [10] recognised, research has been slow in understanding the actual workings of networked systems and the focus has been on structural form and analysis. As a result, there are many methods for generating (e.g., the small world model [11] and the scale-free network [12]) and measurements for comparing static, complete (and not necessarily social) networks (e.g., [13]). However, it has been recognised that social networks have certain properties, in particular with respect to the similarity between people, their spatial proximity, the overall clustering coefficient (i.e., how tightly-knit the network is) and the variation in size of personal networks (e.g., how many friends do people have; also known as the degree). Hamill and Gilbert [14] presented a model known as social circles, where two people are connected depending on the distance between them. This distance could be social (e.g., based on whether two people are similar in terms of age, gender, occupation, religion, or shared values etc.) or spatial.

C. The effects of social networks on activities

The bulk of the research on the effects of social networks on activities is at the data analysis stage. Individuals are surveyed about their social network and asked to complete an activity diary for several days, listing who they interacted with and the nature of the activity.

As part of the Connected Lives study, Carrasco [15] collected data on individuals' personal networks and interactions, then used multi-level modelling to look for influences on frequencies of activities. The results showed that the number of components (i.e., subgroups), density (i.e., clustering), and degree of the personal network influences the frequency of social interactions, and are a better indication of frequency than the size of the network or isolates. Younger people tend

to have a higher frequency of activities, as well as egos and alters with similar ages.

The latter is an example of homophily, which is based on the idea that individuals interact with others who are similar to them [16]. Homophilies can be separated into two groups: those based on status, both ascribed (e.g., age, gender, etc.) and acquired (e.g., occupation, religion, etc.), and those based on values, such as attitudes and beliefs.

Given the data collected for activity-travel modelling purposes, at least two network generation algorithms have been developed. Illenberger et al. [17] presented a model based on spatial distance, while Arentze and Timmermans [18] developed an algorithm based on spatial and social distance. The latter can also be extended to include the influence of common friends, following the theory that if person 1 is friends with person 2 and person 3, then persons 2 and 3 have a good chance of also being friends.

Hackney and Marchal [19], building on previous work, developed a microsimulation which incorporated a social network on top of a daily activity scheduler. The individuals in the system exchange information with each other, either about locations or about friends. Currently their system does not include collaborative scheduling.

III. MODEL DESCRIPTION AND DESIGN

Joint social activities are defined by the different people involved, their relationships and interactions with each other, and their activities in and possible movement around the environment. The topology of interactions is not homogeneous and clusters may form. Therefore agent-based modelling appears to be appropriate for our model, due to the complex relationships and interactions between individuals and the individuals' situatedness in an urban environment [20].

The model consists of agents located in a spatial environment, where they have a home location. This environment is represented by a network of locations. Each agent has a list of other agents he/she is friends with and a list of locations that he/she knows. They also have sociodemographic attributes (e.g., age, gender, car ownership, work status etc.) and a schedule with a certain number of time periods. Each agent can undertake maximally one activity per time period.

Each pair of agents has a similarity measure, which follows from the notion of homophily. Pairs also keep track of when they last saw each other. Links are undirected, meaning that friendships are mutual.

The goals of the agents in the system are derived from the social needs of humans, which include interacting with, and gaining the respect and esteem of others. The agent goals are therefore:

- making and maintaining (long-term) relationships with other people;
- sharing experiences with other people, in the form of joint activity participation;
- sharing (giving and gaining) information with other people; and
- learning about their local environment.

In this paper we focus on the second goal of joint activity participation. Utility-based agents are used as this allows the agents to evaluate the outcomes of participating in different activities. This has advantages and disadvantages: utility functions are difficult to develop and tend to oversimplify the real-world processes [21], however as the aim is to create a model of a sample population for a city, i.e., thousands of agents, the agent model needs to be simple in order to be scalable.

A utility function (Equation 1) has been developed to take into account the required issues – type (a) and purpose (p) of the activity, location (l), day (d), the other person involved (j) –, essentially, what, where when and who. This is based on the needs-based theory discussed in Section II-A.

$$U_i(a, p, l, d, j) = V_i^{ap} + V_i^l + V_i^j + \epsilon \quad (1)$$

$$V_i^{ap} = f_t(\alpha_i^{ap}, d - t_{ap}) \quad (2)$$

$$V_i^l = f_t(1 - d_{il}, d - t_l) \quad (3)$$

$$V_i^j = f_t(s_{ij}, d - t_j) \quad (4)$$

$$f_t(x, t) = \left(\frac{2}{1 + e^{-xt}} \right) - 1 \quad (5)$$

$$s_{ij} = Q_g + Q_a \quad (6)$$

Activities can have a purpose, chosen from sharing experiences, sharing information, informal chatting, and support. The different purposes can be used to determine who is suitable for a given activity. Activities can also have a type, such as shopping, eating out, or sporting activities, which determines the location of the activity. In future, this will be also used to determine the duration of the activity.

The components of the utility function U_i consider when an individual last undertook an activity (Equation 2), visited a location (Equation 3), or saw someone (Equation 4). These values (t_l , t_{ap} , t_j) are combined with the date of the proposed activity d to find the last time the particular event happened. The utility increases over time (Equation 5), so that an activity/location/person that an individual hasn't seen/visited for a while is more attractive than one seen/visited the previous day.

The preferences for an activity with a particular purpose and type (α_i^{ap}) is also an input to the model. In this instance of the model, we consider preferences to be unidimensional as a simplification. It could be that preferences are dependent on the composition of the group, for example, in terms of gender, cultural background, size of the group etc.

The distance to the location (d_{il}) is also taken into account, based on the individual perception of the environment and travel time. For each pair of individuals i and j , a similarity measure was calculated (Equation 6), taking into account age (a) and gender (g). The values of d_{ij} and s_{ij} are scaled to $[0, 1]$.

In order to schedule activities, the agents need to negotiate with each other. This can be done using a negotiation protocol. Given that our aim is to understand the relation between social network and activities, we are more interested in the group formation than on the specific time and type of activity

undertaken. As such, we use the package deal method [22] that abstracts from negotiation issues (for example, the activity may determine the time and location or vice versa, or in which order they should be discussed).

We further assume that interactions and activities are undertaken between two agents, who are connected to each other in the social network. This means that the social and location networks do not change (as new connections are not being made), therefore the centrality calculations do not change.

Agent i , the host, makes a decision to start an interaction using an altered utility function, where the initial location l is set to the other agent's (j ; the participant) house:

$$U_s(a, p, l, d, j) = V_i^{ap} + V_i^j + V_i^{jl} \quad (7)$$

$$V_i^{jl} = f_t(1 - d_{il}, t_j) \quad (8)$$

If U_s exceeds i 's threshold, the host and participant exchange ideas for days and locations.

- 1) Host proposes an activity.
- 2) The respondent then creates a list of the possible day/time combinations (taking into account the host's time window) and sends them to the host.
- 3) The host collates the day/times and creates a list of the *intersection* of the suggestions.
- 4) The respondent determines what type of locations are appropriate from the patterns provided. They then look up which locations they know of that match those location types.
- 5) The host collates the locations and creates a list of the *union* of the suggestions.
- 6) The host then creates a list of possible activities, taking into account when agents are available and the locations they have suggested. The list is returned to the respondent.
- 7) The respondent evaluates this list using a utility function and returns the list with their preferences.
- 8) Using the Borda ranking method, the host determines the chosen option and notifies the respondent, who adds the activity to their schedule. The host also adds the activity to their schedule.

Negotiations can be unsuccessful if neither individual is available on the same day, neither can suggest any suitable locations, or one individual finds that the utility of all proposed activities does not exceed their threshold.

IV. NETWORK INPUTS

For all input networks, the agent population was constant, with the same personal properties (age, gender), thresholds and parameters, and home location. The average degree was kept roughly the same (~ 10), which is in line with analysis of friendship/social interaction networks [18].

Four different networks were generated. The first was a random graph based on Erdos-Renyi random graph [23], randomly generated by the NetworkX package for Python [24]. This network is shown in Figure 1.

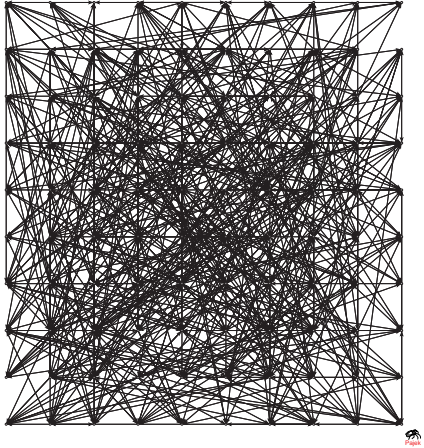


Fig. 1. The random network.

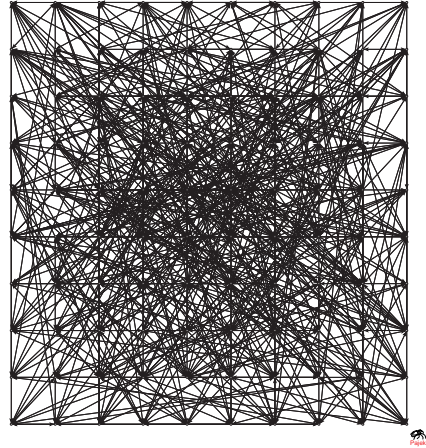


Fig. 3. The social circles network taking into account social distance.

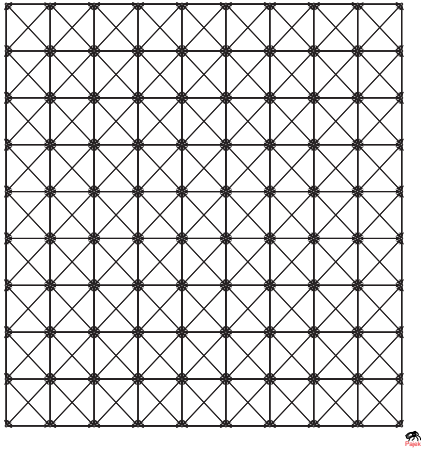


Fig. 2. The social circles network taking into account spatial distance.

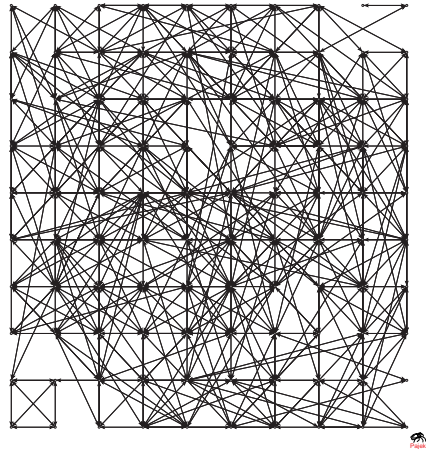


Fig. 4. The social circles network taking into account spatial and social distance.

The other networks were based on the social circles algorithm [14]. All individuals used the same distance size for simplicity, however this varied per network in order to meet the average degree requirement. The social distance was based on Equation 6.

The second network used only spatial distance as the distance measurement (Figure 2).

The third used only social distance as the distance measurement (Figure 3).

The fourth used both spatial and social distance as the distance measurement (Figure 4).

The different social networks have differing clustering coefficients and assortativity on degree (i.e., nodes are connected to other nodes with similar number of nodes [10]) and on node attributes such as age, gender, and activity threshold. These properties are shown in Table I.

V. AN ILLUSTRATIVE SCENARIO

In this scenario, the only locations present are home locations. This means, that for an activity between two agents, only two locations are possible. Activities were also scheduled for the current time period, however the protocol does allow for looking ahead. For the one activity type and purpose, $\alpha_{home, social}$ was set to 0.5. Each agent has an activity threshold randomly chosen from $[0.5, 1, 1.5, 2.0]$.

The agents all use the same utility function and negotiation protocol. Each agent also has an age level in the range $[1 - 4]$, which is consistent with the aggregation used in activity-travel surveys (e.g., [18]). The gender similarity is $Q_g = 1$ if two agents have the same gender, and $Q_g = 0$ otherwise. For age, following [18], $Q_a = 4 - n$, where n is the difference between the two age classes. The overall similarity or social distance s_{ij} is scaled to $[0, 1]$.

The error term takes into account the location ($N(0, 0.2)$),

Type	Degree	Cluster	Assort (degree)	Assort (threshold)	Assort (age)	Assort (gender)
Random	10.141	0.105	0.036	0.017	-0.021	-0.040
Spatial	10.141	0.509	0.531	0.009	0.069	-0.052
Social	12.040	1	1	0.0112	1	1
Soc/spa	10.505	0.491	0.264	0	0.862	0.565

TABLE I
THE PROPERTIES OF THE DIFFERENT SOCIAL NETWORKS.

each participant ($N(0, 0.1)$), and a personal short- (i.e., drawn every timestep, $N(0, 0.5)$) and long-term (i.e., drawn at the start of the simulation, $N(0, 0.2)$) error.

The model was run for 28 time periods as a warmup, and then for a further 28 time periods to collect data.

The aim of the experiment is to validate the following hypotheses:

- H1. The network structure will affect the number of activities.
- H2. The network properties will affect the number of activities.
- H3. At the node level, the distribution of activities will be different for different input networks and the node attributes (degree, clustering) will affect the number of activities.
- H4. At the relationship level, the distribution of activities will be different for different input networks and the dyad attributes (similarity, distance) will affect the number of activities.
- H5. The interaction protocol will be sensitive to different input networks in terms of the number of successfully and unsuccessfully negotiated activities.

VI. RESULTS AND DISCUSSION

All analysis was done in R, a statistical analysis package. ANOVA tests were used to measure the difference in means of output variables for different input networks, while Kolmogorov-Smirnov tests can indicate whether two distributions are similar. p indicates the significance of each test and r denotes the correlation coefficient. If p is less than 0.05, then this indicates that the result is statistically significant.

A. Hypothesis 1: The overall network structure

The effect of the overall network structure on the number of activities was measured using an ANOVA test. The result suggested a significant difference between the input network types ($p < 0.001$).

This means that hypothesis 1 can be accepted, as the network structure affects the number of activities.

B. Hypothesis 2: The network properties

The correlation between each network property (clustering coefficient, assortativity on degree) and the number of activities was not significant. This indicates that these aggregate measurements are not a good indication of the outcomes of the processes in the system and therefore hypothesis 2 cannot be accepted.

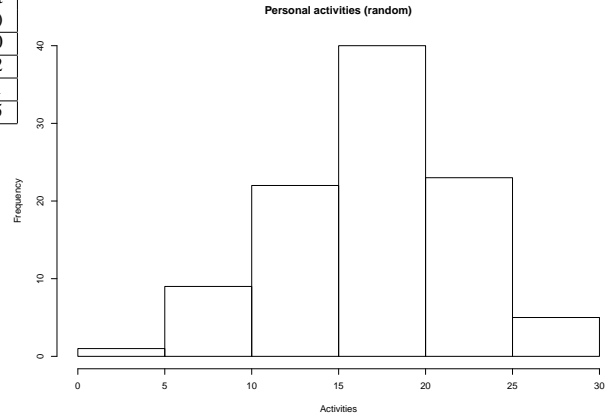


Fig. 5. The distribution of activities for the random network.

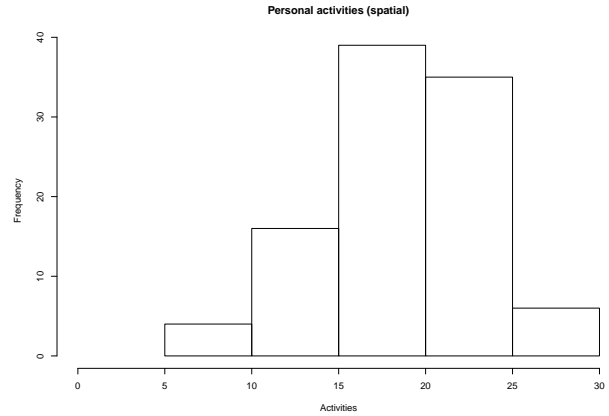


Fig. 6. The distribution of activities for the spatial network.

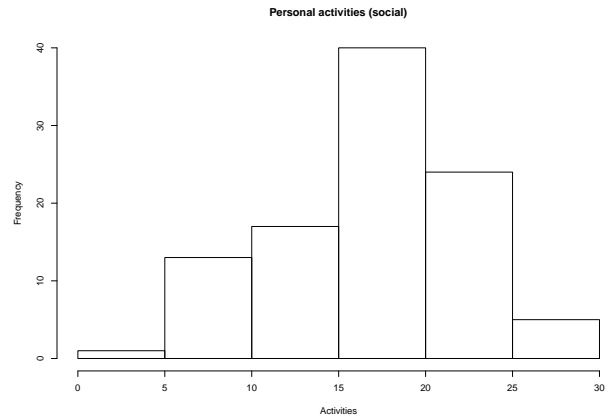


Fig. 7. The distribution of activities for the social network.

C. Hypothesis 3: At the node level

By averaging the number of activities across the ten runs for each person, the distribution of the activities can be measured.

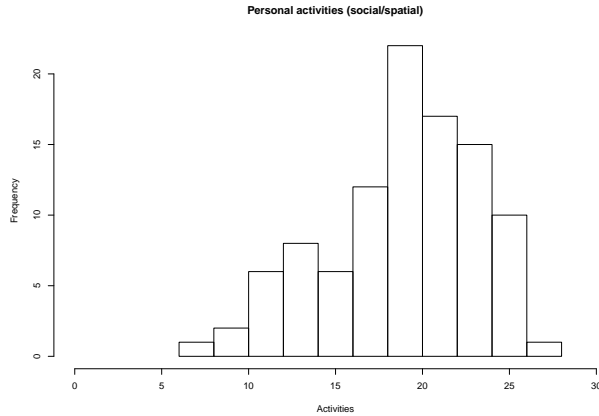


Fig. 8. The distribution of activities for the social/spatial network.

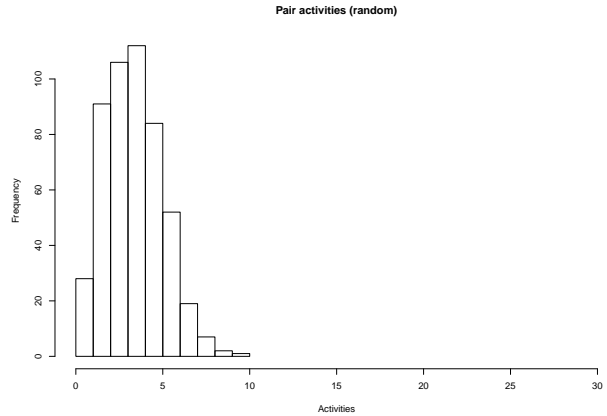


Fig. 9. The distribution of activities per pair for the random network.

Using a Kolmogorov-Smirnov test can indicate whether the distributions are similar or not.

The distributions at the node level are not significantly dissimilar, as shown in Figures 5, 6, 7, 8.

The correlation of the number of activities per person and their centrality or degree is significant ($p < 0.001$, $r = 0.216$). This could be because those with more friends have more opportunity to engage in activities. The threshold for activities is also significant ($p < 0.001$, $r = -0.328$), meaning that those with lower thresholds are participating in more activities as expected. The individual clustering coefficient is not significant, as activities are limited to only two agents. We would expect this to become significant if larger group sizes are modelled.

Although some individual properties are significant, as the overall distribution of activities is not dissimilar, hypothesis 3 cannot be accepted.

D. Hypothesis 4: At the relationship level

As with the personal level, the activities across runs for each pair were averaged. The distributions at pair level were significant (all $p < 0.01$), with the exception of the random network and the social/spatial distance network ($p = 0.70$). The distributions can be seen in Figures 9, 10, 11, 12.

There was a very weak correlation between the similarity of pairs and activities ($p < 0.05$, $r = 0.041$).

The correlation between distance between pairs and the number of activities was stronger ($p < 0.001$, $r = -0.347$), which shows that pairs who live closer to each other are engaging in more activities together.

These results indicate that the relationship level attributes of the network are more significant than the overall or the node attributes and therefore hypothesis 4 can be accepted.

E. Hypothesis 5: Performance of the protocol

We expect that the negotiation protocol is sensitive to the network. The protocol can fail at two points: if agents are not available at the same time, or there is no overlap in the

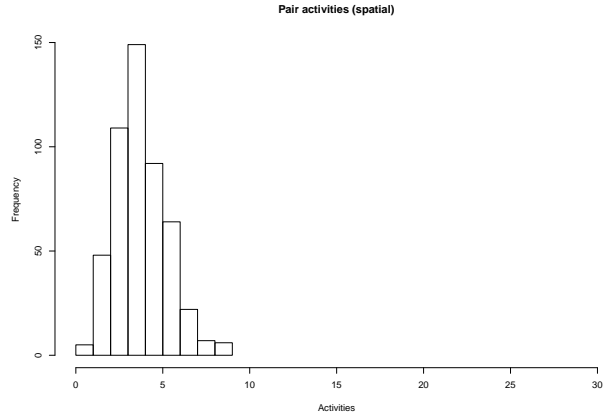


Fig. 10. The distribution of activities per pair for the spatial network.

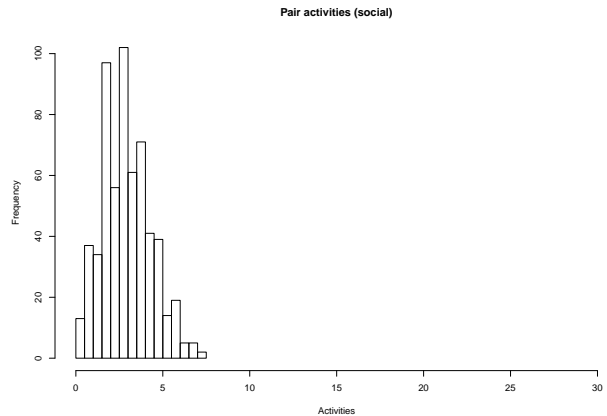


Fig. 11. The distribution of activities per pair for the social network.

preferred activities (e.g., both agents want to do completely different activities, or one does not like any of the options).

We have already shown that the successful activities differs

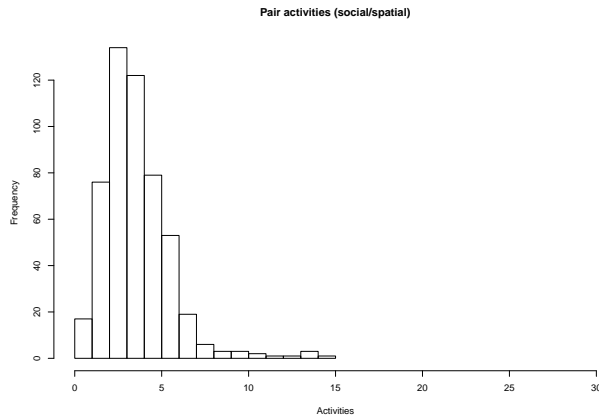


Fig. 12. The distribution of activities per pair for the social/spatial network.

for each network. The unsuccessful activities due to time ($p < 0.1$) and due to activity disagreement ($p < 0.01$) also differs for each network. Table II shows the average for each type.

Network	Successful	Unsuccessful (time)	Unsuccessful (activity)
Random	868.2	834.2	437.5
Spatial	967.5	876.5	178.7
Social	882.7	834.4	405.5
Soc/spa	951.3	868.8	200.7

TABLE II
THE NUMBER OF SUCCESSFUL AND UNSUCCESSFUL NEGOTIATIONS.

The networks with some sort of spatial component performed better; with these networks as a base, agents are less likely to decline an activity based on distance.

From these results, hypothesis 5 can be accepted.

F. Summary

The experiment shows that overall, the key factor is not the overall structure of the network, but the nature of the links between agents.

Whether spatial or social distance is given more weight in the utility function will also influence the outcomes. In this experiment, they were treated equally.

VII. CONCLUSION

Multi-agent simulation is a useful method for modelling the decision-making processes undertaken by individuals, in this case, regarding whether they participate in a social activity with other people or not. Current research assumes that social networks influence social activities, therefore testing the sensitivity of potential decision-making models to different networks is an important step in evaluating the usefulness of incorporating social networks in activity-travel models. This step could also be important for other domains where the social network is influential, e.g., social support networks or exchange networks [25].

We have described an agent-based simulation of social activities and discussed the results of experimentation with

several input networks, differing in structure and properties. We show that the relationship properties within the network are more significant than individual or overall network properties for this type of model. However, as the model is developed further, some personal or network properties could become important. For example, people can only maintain a certain number of friends, so the degree becomes important.

The model was simplified to one activity type/purpose and no network dynamics, so that the effects of the input network could be seen. Future work involves extending the model to include further details about activities (including different locations, activities with more than two participants, and taking into account time pressures/value of time), experimenting with agents using different utility functions and/or negotiation protocols, and exploring the effects of social distance/homophily in closer detail, in particular in the context of cultural characteristics.

The results of our research will be used by city planners to evaluate the effects on social activities and travel of both changes in population and their characteristics (e.g., increasing elderly population, an increase/decrease in car ownership) and changes in infrastructure (e.g., public transport routes, locations of new shopping facilities).

As research into the effects of social networks on travel behaviour is in its early stages, there are little data available and as a result most models are in early stages of development. Research into how these models can be validated is in progress [26]. However, this work can be seen as a step forward in the requirements for sensitivity testing of such models.

ACKNOWLEDGEMENTS

The first author would like to thank Theo Arentze and Harry Timmermans. The comments from the anonymous reviewers were also appreciated.

REFERENCES

- [1] M. Balmer, "Travel demand modeling for multi-agent transport simulations: algorithms and systems," Ph.D. dissertation, ETH Zürich, 2007.
- [2] J. Urry, "Connections," *Environment and Planning D: Society and Space*, vol. 22, pp. 27 – 37, 2004.
- [3] C.-H. Wen and F. S. Koppelman, "A conceptual and methodological framework for the generation of activity-travel patterns," *Transportation*, vol. 27, pp. 5–23, 2000.
- [4] J. Hackney and F. Marchal, "Model for coupling multi-agent social interactions and traffic simulation," in *Proceedings of Frontiers in Transportation 2007*, 2007.
- [5] K. Axhausen, "Social networks, mobility biographies and travel: The survey challenges," Institut für Verkehrsplanung und Transportsysteme, Tech. Rep. 343, 2006.
- [6] T. A. Arentze and H. J. Timmermans, "A learning-based transportation oriented simulation system," *Transportation Research B*, vol. 38, pp. 613–633, 2004.
- [7] T. Arentze and H. Timmermans, "Social networks, social interactions and activity-travel behavior: A framework for micro-simulation," in *TRB 2006 Annual Meeting*, 2006.
- [8] E. Miller, "An integrated framework for modelling short- and long-run household decision-making," in *Progress in Activity-Based Analysis*, H. Timmermans, Ed. Oxford, England: Elsevier, 2005, pp. 175–202.
- [9] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *Science*, vol. 323, pp. 892 – 895, 2009.
- [10] M. E. J. Newman, "The structure and function of networks," *Computer Physics Communications*, vol. 147, pp. 40 – 45, 2002.

- [11] D. J. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440 – 442, 1998.
- [12] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509 – 512, 1999.
- [13] R. A. Hanneman and M. Riddle, "Introduction to social network methods," 2005. [Online]. Available: <http://faculty.ucr.edu/~hanneman/nettext/>
- [14] L. Hamill and N. Gilbert, "Social circles: A simple structure for agent-based social network models," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 2, p. 3, 2009. [Online]. Available: <http://jasss.soc.surrey.ac.uk/12/2/3.html>
- [15] J. A. Carrasco, "Unravelling the social, urban, and time-space context of activity-travel behaviour: Results from a social network data collection experience," in *Proceedings of the 12th International Conference on Travel Behaviour Research*, 2009.
- [16] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, pp. 415 – 441, 2001. [Online]. Available: <http://www.jstor.org/stable/2678628>
- [17] J. Illenberger, G. Flötteröd, M. Kowald, and K. Nagel, "A model for spatially embedded social networks," in *Proceedings of the 12th International Conference on Travel Behaviour Research*, 2009.
- [18] T. Arentze, P. van den Berg, and H. Timmermans, "Modeling social networks in geographic space: approach and empirical application," in *Proceedings of the workshop on Frontiers in Transportation: Social networks and travel*, 2009.
- [19] J. Hackney and F. Marchal, "A model for coupling multi-agent social interactions and traffic simulation," in *TRB 2009 Annual Meeting*, 2009.
- [20] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation part 2: How to model with agents," in *Proceedings of the 2006 Winter Simulation Conference*, L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, Eds., 2006, pp. 73 – 83.
- [21] M. J. Wooldridge, *An introduction to multiagent systems*, 2nd ed. Chichester, England: John Wiley & Sons, 2009.
- [22] S. S. Fatima, M. Wooldridge, and N. R. Jennings, "Multi-issue negotiation with deadlines," *Journal of Artificial Intelligence*, vol. 27, pp. 381 – 417, 2006.
- [23] P. Erdős and A. Rényi, "On the evolution of random graphs," *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, vol. 5, pp. 17–61, 1960.
- [24] NetworkX Developers, "Networkx," 2010. [Online]. Available: <http://networkx.lanl.gov/>
- [25] K. S. Cook, "Exchange and power in networks of interorganizational relations," *The Sociological Quarterly*, vol. 18, no. 1, pp. 62 – 82, 1977.
- [26] N. Ronald, T. Arentze, and H. Timmermans, "Validation of complex agent-based models of social activities and travel behaviour," in *Proceedings of the 12th World Conference on Transport Research*, 2010.

Human behaviours simulation in ubiquitous computing environments

Teresa Garcia-Valverde, Francisco Campuzano, Emilio Serrano and Juan A. Botia
University of Murcia, Spain. E-mails: {mtgarcia,fjcampuzano,emilioserra,juanbot}@um.es

Abstract—Ambient Assisted Living (AAL) systems’ main goal is to augment live quality of elderly people, by using ICT based systems. In this paper, we are concerned with the artificial reproduction of a physical environment (i.e. a house) and an elder (i.e. the attended) living in such environment. An agent based social simulation system is used for such purpose. Such simulator will allow the integration of ubiquitous computing appliances, services and applications in such environment. A realistic reproduction of human behaviour in the simulator helps, in this context, in the validation of silent monitorisation, diagnosis and action based applications. Proofs are given in the paper which demonstrate the level of reality reached by comparing the artificial behaviour with real ones.

Index Terms—Ubiquitous computing, Ambient Assisted Living, behaviour simulation, user modelling.

I. INTRODUCTION

The main thesis presented in this paper is the following: agent based social simulation (ABSS) [1] may help in the engineering of Ambient Intelligence systems. ABSS is a simulation paradigm in which the focus is put on the definition of the separate components of the simulation in an isolated manner. In such simulation runs, the emergence of behaviours is the main subject under study. And the metaphor of agent is used for specification of single components and interactions among them. An ambient intelligence [2] system is a set of appliances, services and applications which silently surrounds and interact with the user in an intelligent manner. In such kind of systems, the user is the central entity of the model. Starting from the user, services and applications are built.

A main difficulty one may find in the development process of an AmI system is that of testing and validation. Testing is the process of executing a program with the intent of finding errors in the code [3]. Such errors must be debugged [3]. Some of the errors may be found by using a Unit approach with the system under test (SUT). Common errors which are found in this stage are related to common programming mistakes (e.g. values of variables out of range, shoddy checking of return values from methods and so on). Thus, robustness of the program is a must here. But a more elaborated test set may be defined in order to assess the functionality of the system (i.e. it behaves as expected). But, if the main issue in AmI systems is a smooth interaction with the user, an Unit based approach is no more valid here. It is clear that the user, or at least a model of the user, should be incorporated in the development process in order to measure to what extent, the SUT is behaving as expected when interacting with him. In this paper, an approach to test and validate AmI systems in a

stage prior to deployment is presented. The main idea behind this is that the user is modelled with a computational model and integrated into an ABSS model which incorporates as a simulated artifact, the environment, the hardware (i.e. mainly sensors and interfaces with the user) and integrates the real software (i.e. services and applications). The real software is precisely the SUT here.

The proposal is articulated by means of a methodological work. Such a methodology is a set of procedures which guides the developer in the definition, creation, testing and validation of the AmI system. It is based on a methodology previously described by Gilbert et al. [1]. It comprises the creation of the necessary ABSS models and how they should be employed to find errors in AmI services. The application of the methodology is exemplified in a real domain. The application domain is AAL (Ambient Assisted Living). An AAL system is an ICT based solution which is devoted to augment quality of live for elderly people. In this case, the interest is focused on a system called Necessity [4]. It is based on a sensor network deployed through the house and a central processing unit. Sensors include presence, pressure and open-door. The system is designed to work on single person environments (e.g. an elderly who lives alone and independently in his house). It is in charge of monitoring activity regime of the elderly 24x7 in a manner that when his activity pattern is anomalous, an emergency response is started. The rest of the paper will demonstrate that an artificial reproduction of a house, sensors and the attended, together integrated with the real software, helps in the fine tuning of the activity pattern management software.

The rest of the paper is structured as follows. Section II introduces the methodology used for the engineering of AmI services. Section III introduces the computational models employed to artificially reproduce the behaviour of the attended. Such behaviour is based on a probabilistic and hierarchical automata which governs the activity and location of the modeled elderly in each instant of time. In section IV, the validation approach is presented. It is based on the statistical contrast of artificial data traces obtained by simulating the automata just mentioned with similar traces coming from real users in similar context.

II. AVA, AN AGENT BASED METHODOLOGY FOR THE VALIDATION OF AMI SYSTEMS

This section explains an agent based methodology for the validation of AmI systems called AVA. This methodology

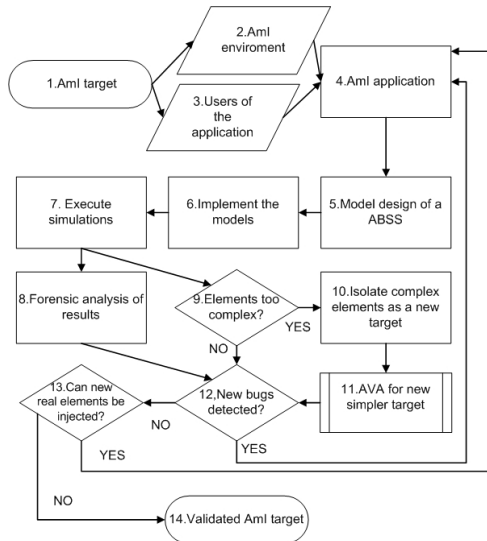


Fig. 1. AVA, An Agent based methodology for the Validation of AmI systems expressed as a flowchart

proposes the development of ABSS in order to validate AmI applications. AVA is an extension of the methodology promulgated by Gilbert et al. [1] for the development and use of general ABSS. The two main innovations with regarding the classical methodology by Giltber et al. are: (1) the existence of a step to generate simpler simulations and (2) the consideration of including real elements in the simulation to get more realistic results. This section will show the advantages of these innovations for the specific purpose of validating AmI applications. The AVA methodology is expressed as a flowchart in figure 1¹.

Gilbert et al. [1] defines the target of a social simulation as some “real world” phenomenon which the researcher is interested in. The AVA methodology is proposed to validate an AmI application including their interactions with the environment and users. Thus, AVA starts considering an AmI target (step 1) which includes an environment, users and an AmI application which may be finished or at an advanced stage of development. Typically, the use of ABSS to generate knowledge involves a necessary familiarization with the domain in the first step. This is required to generate models of the target in the following steps of the methodology [1]. The main elements to be studied in an AmI system are: the environment (step 2), users (step 3) and the AmI application (step 4). Note that while the environment and the user are inputs, the application system is a process. In principle, the environment and users are external and do not support changes. On the other hand, the AmI application can be modified. This application will be refined along the iterations of AVA to get a realistic validation. This

¹The flowchart uses standard elements of classic flowcharts [5] as flow of control (represented as arrows), processes (represented as rectangles), decisions (rhombus), input/output (parallelograms), start and end symbols (ovals) and predefined processes (rectangles with vertical lines at the sides).

paper discusses the performance of these steps for an AAL system for elderly people

The development of an AmI system model is performed in step 5. The model design associates the real system with a representation of this system (the model) [6]. Here, the AmI application must be modelled but also the users and the environment. These models are necessary to validate the AmI application because they interact with it. Moreover, a realistic validation of the application needs realistic models for users and environment. Therefore, models must describe reality before being as simple as possible [7]. Section III explains the construction of a user model for an specific AmI application.

Step 6 deals with the implementation of the AmI system in a simulation language. The implementation from the concepts of the model is not a trivial task [6]. A general programming language or a specific one of the available frameworks for the development of ABSS can be used for the construction of the simulation. The second option is much more convenient because several of the typical tasks in the construction of ABSS have been included in this kind of software packages [1]. Examples of these tasks are scheduling agents’ actions or building basic environments. The web of the Open Agent Based Modeling Consortium² nowadays lists 22 of these frameworks. Section III shows the use of a specific software package for the implementation of a realistic environment model: 3D Sweethome.

After building the simulation, this must be executed (step 7). Quick, cheap and numerous experiments can be performed thanks to the ABSS. These executions produce large amounts of data regarding the behaviour of users, environment and application models. Forensic analysis, step 8, is an offline analysis to be conducted on the data stored from the previous step. The analysis should consider whether the AmI application functionality is correct. Furthermore, this step must validate that the behaviour of users and environment models is consistent with the observed reality (steps 2 and 3). Without this validation, the theories generated from simulations have no relation to reality (as they are based on non-descriptive models). Therefore, the functionality of the AmI application model is linked to the users and environment models. Section IV deals with the validation of the users model for an AAL system

One of the innovative points in the AVA methodology is the use of simple simulations as a means to validate descriptive simulations. Step 9 checks if any elements of the simulation are too complex. In this case, complexity means that it is difficult to assess whether the behaviour of an element is the expected one. For example, some behaviours of users can be so complex that they need to be evaluated in isolation. An example of this type of behaviour would be the resolution of collisions on the motion of a large number of agents. This behaviour is a problem to be studied itself and its validation would be much more complicated with additional elements in

²OpenABM Consortium website: <http://www.openabm.org/>

the simulation (more users' behaviours, a realistic environment model, a realistic model of an AmI application, etc). In these cases, the methodology proposes to consider these complex elements as an object of study itself (step 10) and repeat the AVA methodology for them (step 11). The reuse of models and code in this new iteration will be direct because a more descriptive simulation is available as result of the previous steps. Once the complex element is validated in a simpler simulation, which is the final result of the methodology, the next step for the overall simulation (step 12) can be performed.

Step 12 checks if the developer has found errors in functionality. If that is the case, the AmI application of step 4 must be modified in order to correct these errors and the process repeated. Besides the primary objective (validating the AmI application), is typical to find bugs of previous steps at this point in the form of implementation failures or unrealistic models.

The final decision, step 13, checks if actual elements of the AmI system can be connected to the simulator. The AVA methodology proposes to inject or connect real elements in the simulation progressively in order to make more realistic validations³. This process is called "*reality injection*" and the basic idea is that real elements can coexist with simulated elements. After connecting real elements, the methodology must be repeated from step 4 to improve the application and the models. The result is an exhaustive validation which is as realistic as possible. The obvious question is why models and simulations are necessary if real elements (as real users) can be injected. The answer is that a model, by definition, is somewhat easier to study than the modelled reality. The purpose of including real elements in simulations is to improve the realism of the models. Then, in subsequent iterations, the real elements will not be included because the pure simulations allow faster and cheaper tests.

Finally, if models are descriptive enough, the bugs found in the functionality of the AmI application model will correspond to failures of functionality in the real AmI application. These failures should have been corrected in each iteration of the AVA methodology. Therefore, the result of the methodology (step 14) is that the AmI target is exhaustively validated.

III. REALISTIC BEHAVIOUR MODELLING

In this section, the particular models used, in the application of the AVA methodology in the AAL domain, are introduced. In section III.A, it is presented how the physical environment (i.e. the house and furniture) and sensors were defined. Section III.B refers to the production of realistic computational models for elders living in such environment and making sensors to react on their presence. Having such models (i.e. the house, sensors and persons) within a simulation, and its integration with the ubiquitous computing software, such software can be tested.

³Notice that this not involves necessarily a Participatory Multiagent Simulation. The real elements do not have to be humans playing the role of simulation components. These elements can be software applications, hardware or even parts of the environment.

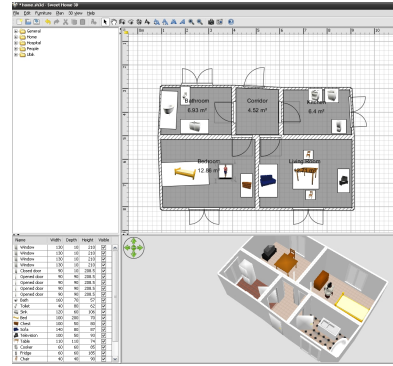


Fig. 2. A plane model in Ubik's editor and its 3D representation

A. Environment Modelling

For Multi-Agent Based Simulation (MABS), it is available Ubiksim⁴, a simulator developed by University of Murcia that works over MASON⁵. It has integrated an environment modeling tool based on SweetHome3D⁶, an application conveniently adapted for modelling attended people and their environment. It is possible to create houses over a 2D plane, also it offers a 3D navigable view (figure 2). This view is used for simulation's visualization at real time.

In the physical environment generated by using the Ubik editor, a simple house (see figure 2) with a kitchen, a bathroom, a bedroom and a living room is modelled. Presence sensors are included in every room of the house. And a sensor for open door (it is necessary for knowing when the elder leaves the home) is also included in the outdoor. When a simulation is run, the person moves in the house and stimulates sensors when he is detected by them. Such sensors, through the ubiquitous computing software, generate events. And these events generate log entries. Such simulated log entries are used afterwards to check if the virtual elder behaves in a realistic manner (see section IV).

Notice that log entries (both in the simulator and the real setting) are generated by the same monitoring service which continuously checks if the elder may be suffering some problem, by using a pattern recognition approach (more details on this may be found in [4]) on the events coming from sensors. In the first case, the person is virtual, in the second it is real. But the monitoring service is the same.

B. Behaviour Modelling

The target of the modelling activity is a typical aged person, who lives independently and alone in his own house. As he lives alone, the following situation may occur: he may suffer some health problem and stay immobilised in the floor for too much time before anybody comes and notices

⁴UbikSim: <http://ubiksim.sourceforge.net>, last access: 20 May 2010

⁵MASON Toolkit: <http://cs.gmu.edu/~eclab/projects/mason/>, last access: 20 May 2010

⁶Sweet Home 3D: <http://www.sweethome3d.eu/es/index.jsp>, last access: 20 May 2010

that something is wrong. But it is possible to develop an ubiquitous computing system which detects it and generates some emergency response process [4]. By following the AVA methodology, we may use a simulated elder within a simulated environment to test such system before it is deployed in a real environment for pilot testing. Such simulated elder should be necessarily simulated along the 24 hours of the day, repeatedly for a determined number of weeks. For this, it is assumed that the day is divided into time slots (i.e. morning, noon, afternoon and night). In each time slot, it is also assumed that the simulated person behaves specifically for such slot.

The behaviour of simulated people are modelled probabilistically. In this approach, behaviours are defined as situations the agent should play in each moment. Transitions between behaviours are probabilistic. The underlying model is a hierarchical automaton (i.e. in a higher level there is a number of complex behaviours that the agent may play and once it is in a concrete state, within the state there is another automaton with more simple behaviours). So, the modelling of each behaviour is treated separately and the modeller is abstracted of unnecessary details. So, in the lowest level (basic actions), each state is atomic. An agent never conducts two behaviours of the same level simultaneously.

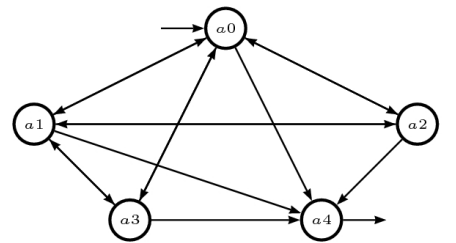
The behaviours used for modelling elders are of three types:

- Monotonous behaviours: the kind of behaviour the elder manifest always approximately in the same time slot, and on a daily basis (e.g. sleeping, having meal, medication and so on).
- Non monotonous behaviours: the kind of behaviour the elder usually manifest, not bounded to a concrete time slot, and repeated within a non constant period (e.g. going to the toilet, having a shower, cleaning the house and so on).
- Any time behaviours: such behaviours will sometimes interrupt others the elder is already doing, and will be generated regardless they were already generated in a temporal proximity (e.g. in his spare time).

A probabilistic automaton [8] is defined as the quintuple $(Q, V, P(0), F, M)$ where Q is a finite set of states, V is a finite set of input symbols, $P(0)$ is an initial state vector, F is a set of final states and M is a matrix that represents probabilities of transition for every state. In this definition, transition's probabilities depend on time. According to different daily time slots, the agent's behaviour acts on a different pattern. There are time slots for eating, sleeping and taking medication (Monotonous behaviours).

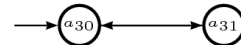
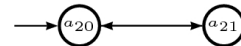
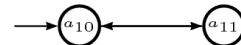
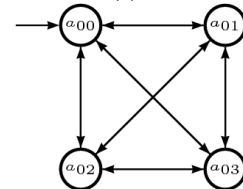
Notice that, when the elder is at any state, the necessity of changing to another state may arise. But this is not done immediately. Moreover, a number of different changes (i.e. transitions) may be pending simultaneously. Thus, a list of pending tasks (i.e. or events) is maintained. Such tasks are ordered by a static priority (e.g. going to the toilet goes before cleaning).

For generating transitions in real time, probability distribution functions are used according to the type of the behaviour and its features. These distributions are member



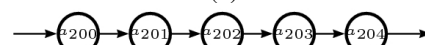
$$Q = \{a_0 = \text{NormalTime}, a_1 = \text{MedicationTime}, a_2 = \text{MealTime}, a_3 = \text{SleepTime}, a_4 = \text{Anomalous}\}$$

(a)



$$Q = \{a_{00} = \text{SpareTime}, a_{10} = \text{MedicationTime}, a_{20} = \text{MealTime}, a_{30} = \text{SleepTime}, a_{x1} = \text{ToiletTime}, a_{02} = \text{ShowerTime}, a_{03} = \text{CleanTime}\}$$

(b)



$$Q = \{a_{200} = \text{GoingToFridge}, a_{201} = \text{GoingToCooker}, a_{202} = \text{Cooking}, a_{203} = \text{GoingToTable}, a_{204} = \text{Eating}\}$$

(c)

Fig. 3. (a) Level 0 automaton, (b) Level 1 automata, (c) Level 2 automaton for state a_{20}

of the exponential family [9], [10]. Section IV defines all distribution functions used.

Notice that monotonous behaviours must be activated at specific time hours or within specific time intervals. For example, in the case of *MedicationTime*, a new necessity of changing to such state will be generated exactly at the time to take medicines. In the case of *MealTime* and *SleepTime*, the necessity is generated within a time interval. The distribution that models these transitions is bounded in this time slot. It must be assured that agent eats and sleeps every day, because of that, if not transition is generated into the time slot, a transition is generated at the end time.

To provide more realism, an automata hierarchy is introduced representing each state by lower level automata which define more specialized behaviours.

As shown in figure 3(a), in level 0 the initial state

is $a_0 = \text{NormalTime}$. For every one of the other states $\{a_1 = \text{MedicationTime}, a_2 = \text{MealTime}, a_3 = \text{SleepTime}\}$ exist a list of times generated by a probability function. When a time counter arrives to one of these times, a transition to state owner of the list is added to pending tasks list. When the action is finished, the automaton returns to initial state if there is not another pending task. The final state is $a_4 = \text{Anomalous}$, if it is reached, the execution will be stopped.

In level 1 (figure 3(b)) non monotonous behaviours are represented. There is an initial state in every refinement where the person does the main task of the upper level (eating, sleeping or taking medication). The state $a_{x1} = \text{ToiletTime}$ is considered in every refinement of level 0. However, the states $a_{02} = \text{ShowerTime}$ and $a_{03} = \text{CleanTime}$ only may be activated in normal state of level 0.

In the lowest level (level 2), the new automata define some specialized actions refining every state of level 1. These new states do not give relevant information, but the agent gains more realistic behaviours.

With the sequence of actions in figure 3(c) the person cooks before eating. It refines state a_{20} of level 1, the agent is not going to be static in the kitchen, because it must be going to different places and spends some time in every state.

In figure 4 a base implementation for agent's behaviour is presented. First of all, all possible automata's states are iterated for initializing lists which contain time instants (lines 13-24). These time instants are generated by a probability distribution function, and they represent when a transition to its associated state is going to be launched. After of that, the automaton begins to run. At every time instant, if actual time is equal than first item of a time list, a transition is added to pending tasks list (lines 57-60). When there is one state with higher priority than actual state in pending tasks list, a transition is also generated. Then, if actual state is unfinished, it is stored as a pending task and a new state is reached (lines 46-50). When this new state is finished, leaved state may be resumed.

The configuration parameters for the whole simulation involve:

- Temporal limit in every room before entering in anomalous state (t_{max})
- The probability distribution parameters according to the kind of behaviour
- Time slots of routinary temporal behaviours, like eating or sleeping ($ini_s, end_s, s \in \text{MealTime}, \text{SleepTime}$)

IV. VALIDATION OF THE APPROACH

From section II, it is clear that user modelling is a means for testing AmI services and applications without a real environment. This task is performed in the third step of the AVA methodology. Regarding to the validation, other important steps within AVA are the steps from 5 to 8. Model validation is needed in order to show that models are able to describe the users' behaviours. The rest of the section shows how the model validation has been approached. But basically, activity data from real users is compared (in statistical form) with the same type of data produced by the artificial models.

```

1 Let pt be a list of pending tasks ordered by priority
2 Let states be a list with all possible automata's states
3 Let times be a list of time instants when transitions
4 are going to be generated
5 Distribution Functions to model the occurrence
6 of monotonous behaviours:
7 Let mb be the function to model monotonous behaviours
8 Let nb be the function to model non monotonous behaviours
9 Let ab be the function to model anytime behaviours
10
11 //Instants of time initialization
12
13 for all s in states do
14   if isAnytime(s) then
15     times(s) <- ab()
16   else if isMonotonous(s) then
17     //Initial and final instants of bounded time slot
18     i <- ini(s)
19     e <- end(s)
20     times(s) <- mb(i, e)
21   else if isNonMonotonous(s) then
22     times(s) <- nb()
23   endif
24 endfor
25
26 actualTime <- 0
27 //actual state is defined by 3 numbers, one per level
28 level0 <- 0
29 level1 <- 0
30 level2 <- 0
31 actualState <- newState(level0, level1, level2)
32
33 //Once initialized the times, the automata begin to move
34
35 //If an anomalous state is reached, the execution stops
36 while (level0(actualState) <> 4)
37   //If actual task ends, initial state is activated
38   if timeLeft(actualState) = 0 then
39     actualState <- newState(0, 0, 0)
40   endif
41   //If next state has higher priority than actual state,
42   //actual state becomes a pending task and it is stored
43   //in pt
44   if (size(pt)) > 0
45     nextState <- first(pt)
46     if priority(nextState) > priority(actualState) then
47       add(pt, nextState)
48       actualState <- nextState
49       remove(pt, nextState)
50     endif
51   endif
52   for all s in states do
53     time <- first(times(s))
54     //If actual time matches first time instant in
55     //times, the task associated with this time instant
56     //is added to pending tasks
57     if actualTime = time then
58       remove(times(s), time)
59       add(pt, s, priority(s))
60     endif
61   endfor
62   actualTime <- actualTime + 1
63   //Decrement remaining time for finishing actual task
64   time <- timeLeft(actualState) - 1
65   setTimeLeft(actualState, time)
66 endwhile

```

Fig. 4. Realistic behaviour implementation

A. Data Preprocessing

Activity data from real users were obtained within a pilot project devoted to the validation of the Necessity system. Around 25 users, all elderly people living independently, were used in such Pilot project. In this paper, data coming from three users, under monitorisation during two months, were used. Data is in the form of a Necessity log. The logs offer the

possibility to represent where the user is, at any moment, at the house (including also if he leaves the house or he is seated or sleeping). Thus, three different data sets, with log entries corresponding to sensor events are available. These data sets need further processing.

Validating the artificial models is assuring that the right probability distribution function is used to reproduce the transition between the different states (i.e. behaviours). Thus, data series are obtained from log data as they were a random number series generated by the corresponding probability distribution. Such series are used afterwards in a goodness of fit test. For example, in case of the non monotonous behaviours and anytime behaviours, preprocessing the log data involves the extraction of the time series of the moments in which each event is produced. These time series only include values within the typical awake period of the corresponding user. Obviously, while the user are sleeping, his daytime routines change.

The data preprocessing for the monotonous behaviours is slightly different. This kind of behaviour is usually produced in a bounded time slots. For example, having dinner, having lunch or sleeping are behaviour which occur during specific time periods of the day. So, the preprocessing involves the extraction of the behaviours events inside the time slots. The time slots can be slightly different for each person, but it is possible to define an approximation of them which will be valid for all (see in section III the configuration parameters). Finally, in each slot time, the time intervals between each occurred event are measured. The extracted time series are composed of these time intervals.

B. Model Diagnosis

Validation is one of the most important issues in a simulation system. Validation consists in the determination that the simulated model is an acceptable representation of the real system, for the particular objectives of the model [11]. There are many techniques for validating simulations [11], [12], and specially, for validating agents based on simulated models [13].

The models which describe social processes, as the model proposed here, are generally hard to validate. In this approach, the behaviour is probabilistically modelled. However, some statistic tests should be done to assume that a probabilistic model is reasonable to explain the data. This process is called model diagnosis. And this section is devoted to make the diagnosis of the models presented above. The most serious problem that one usually faces in this kind of validation is the lack of real data [14]. However, in this work the data from the Necessity project is available and can be used.

From these preprocessed data, some histograms for different behaviours and people are shown in Fig. 5. The sample density is shown with a black line. The dashed line shows the probability density function of the theoretical distribution that models that behaviour.

Graphs 5 (a) and (b) show two monotonous behaviours, sleeping and having dinner (having lunch is similar). In these kinds of behaviours the event that raises the behaviour occurs

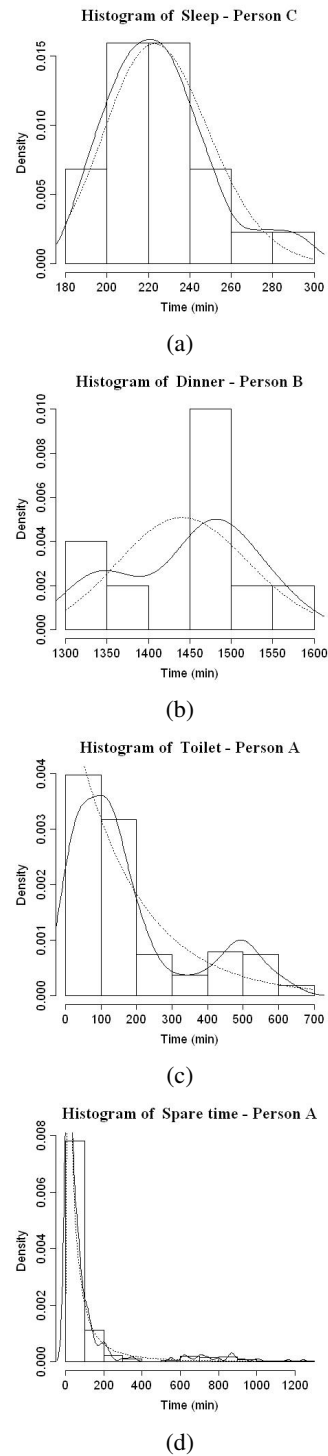


Fig. 5. (a) Minutes between 21:00 hours and the instant the attended C goes to bed, (b) Time between dinners for attended B, (c) Time between uses of the toilet for attended A, (d) Time between spare time for attended A

during a time interval. The interval is usually the same for each attended person, i.e., a person usually goes to sleep or to have dinner at the same hours. Because of this, the curve of the behaviour can be fitted to a gamma distribution. The gamma distribution is usually employed as a probability model for waiting times, in this case, the waiting time until the next event of the behaviour. So, a gamma distribution is suitable for modelling monotonous behaviours.

A kind of non monotonous behaviour, i.e. going to the toilet, is shown in Fig. 5 (c). In this case, the behaviour is fitted with an exponential distribution. Non monotonous behaviours are also waiting time models, but they are defined in wake periods. This is a special case of the gamma distribution which can be modelled with an exponential distribution.

Finally, the group of anytime behaviours are behaviours which will be often interrupted for the other behaviours. Because of this, size of intervals in an anytime behaviour are small due the interruptions. This causes the characteristic curved heavy-tailed distribution, specifically, the Pareto II distribution, also known as Lomax distribution.

Gamma, exponential and Lomax distributions are members of the exponential family of probability distributions. Actually, they are special cases of the beta prime distribution (also known as beta distribution of the second kind, Beta II). The Beta II distribution nests many important distributions as the gamma, the exponential or the Lomax distribution. The gamma and the Lomax distributions are special cases of Beta II. On the other hand, the exponential distribution is a special case of the gamma distribution $\Gamma(\alpha, \beta)$ when $\alpha = 1$, and a special case of the Lomax distribution with some restrictions [15], [16]. So, the Beta II distribution can be used here as a generalized distribution for all group of behaviours: monotonous, non monotonous and anytime behaviours. Then, each behaviour can be specified according to its features in order to obtain a better fitting.

Now, in the rest of the section, empirical evidences which support using gamma, exponential and Lomax for monotonous, non monotonous and anytime behaviours are given.

Notice that it is possible to estimate the distance between time series generated, as explained above, from real log data and time series generated by simulation of theoretical distributions. Such estimation is done by using some statistical test, like the Kolmogorov-Smirnov (K-S) test [17]. The K-S test is a nonparametric and distribution-free goodness-of-fit test. This means that they do not rely on parameter estimation or precise distributional assumptions [18]. The proposed model in this work does not assume any concrete distribution and does not require parameter estimation. This way, the K-S properties are suitable for the hypothesis test.

The K-S test and the chi-square test are the most commonly used and for large size sample both tests have the same power. However, the chi-square test requires a sufficient sample size in order to obtain a valid chi-square approximation [19], [20].

The K-S is a goodness-of-fit test to indicate whether it is reasonable or not to assume that a random sample comes from

a specific distribution. It is a form of hypothesis testing where the null hypothesis says that sample data follows the stated distribution. The hypothesis regarding the distributional form is rejected if the test statistic, D_n , is greater than the critical value obtained from a table, or, which is the same, if the p-value is lower than the significance level. The significance level is fixed in this work at 0.05, which it is the value usually referred in statistical literature.

Table I shows the p-values obtained from the K-S test for each validated behaviour with the adequate distribution. The null hypothesis is that the behaviour sample data come from the stated distribution and it is rejected if p-value is lower than the significance level.

Person	Behaviour				
	Sleep	Dinner	Eat	Toilet	Spare time
A	0.404	0.311	0.361	0.111	0.086
B	0.488	0.467	0.542	0.079	0.108
C	0.337	0.489	0.575	0.137	0.103

TABLE I
P-VALUES

From these results, none of the stated null hypothesis can be rejected. Therefore behaviour of the attended people could be fitted by the specified distribution, as it is described above.

V. RELATED WORKS

Various approaches have been proposed to create autonomous characters. For example, in [21] every character is provided with a small KBS (Knowledge-Based System). Such method is very flexible, but defining the knowledge base is a complex and time-consuming task. A reasoning system is also used in [22].

The approach to behavioural autonomy presented in section III is based in [23], in that approach the idea is to develop agents that act and choose in the way actual humans do. The agents are represented using parametrized decision algorithms, and choose and calibrate these algorithms so that the agents' behaviour matches real human behaviour observed in the same decision context. For this purpose, they use a parametrized learning automaton with a vector of actions associated that can be weighted to choose actions along the time the way humans would.

The decision of representing the automaton's transitions with probabilities instead of using a vector of strengths is based in [24], where the behaviour sequences are modelled through probabilistic automata (Probabilistic Finite-State Machine, PFSMs). Probabilistic personality influence implies that one cannot fully predict how a character will react to a stimulus.

In [25] and [26], the behavioural models described use a hierarchical structure of finite state automata similar as model described in section III. Each behaviour of a behaviour sequence is called a behaviour cell. At the top of the structure there is a behaviour entity with a finite state automaton composed of at least one behaviour cell. An elementary behaviour

is situated at the bottom of the hierarchical decomposition and encapsulates a specialized behaviour which directly controls one or more actions. The list of prioritized events is based in [27], where human agents have a pending task list. Priorities give more realism to human behaviours.

VI. CONCLUSION

In this work, a general behaviour model for different people is proposed. Adjusting the model to specific persons would imply using the suitable configuration parameters for the corresponding probability distributions governing transitions between states of the probabilistic automata. This process is part of a more general task which is testing AmI services and applications. For such purpose, the AVA methodology was presented. It has been applied for the validation of an AAL system called Necessity. More specifically, this paper is focused in producing models of humans (i.e. step 3 of AVA) and validation of the models (steps form 5 to 8).

Such an approach requires, as a means to validate the artificial behaviours, for all the artificial models, a method to check if gamma, exponential and Lomax distributions are well suited. Using the K-S test is possible to quantify the distance between the empirical distribution functions of two samples. Then the K-S test is used to validate the behaviours of the artificial models, by using real data of real elders. A high level of the p-value (higher than the significance level) means that the behaviours are drawn from the same distribution (the null hypothesis is not rejected). Notice that, in most cases, the obtained p-values are higher than the significance level and the null hypothesis is not rejected. But, it is also necessary to remark that a good fitting of the configuration parameters will always be required. And this is due to inevitable heterogeneity in behaviour of people (including elders). As a conclusion, it can be said that the proposed model is suitable for modeling probabilistically the behaviour of simulated people, as the work states.

Future works include a deep study of source data in order to generate a taxonomy of elders in terms of their behaviour within their houses, depending on mobility and habits. Such a taxonomy would be useful for an automatic parameter tuning of the models of elders. The user of the simulator, instead of configuring parameters by hand, would simply choose between a catalogue of elderly people patterns of behaviour.

REFERENCES

- [1] N. Gilbert and K. G. Troitzsch, *Simulation for the Social Scientist*. Open University Press, February 2005. [Online]. Available: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0335216013>
- [2] E. Aarts and J. L. Encarnação, "True visions: Tales on the realization of ambient intelligence," in *Into Ambient Intelligence, Chapter 1*. Springer Verlag, Berlin, Heidelberg, New York, 2005.
- [3] G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas, *The Art of Software Testing, Second Edition*. Wiley, June 2004. [Online]. Available: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0471469122>
- [4] J. A. Botía, A. Villa, J. T. Palma, D. Pérez, and E. Iborra, "Detecting domestic problems of elderly people: simple and unobtrusive sensors to generate the context of the attended," in *First International Workshop on Ambient Assisted Living, IWAAL*, Salamanca, Spain, 2009.
- [5] "Flowcharting techniques," *IBM GC20-8152-1 edition*, 1969.
- [6] A. Drogoul, D. Vanbergue, and T. Meurisse, "Multi-agent based simulation: Where are the agents?" in *Proceedings of the Third International Workshop on Multi-Agent-Based Simulation MABS 2002, Bologna, Italy*, ser. LNAI 2581, J. S. Sichman, F. Bousquet, and P. Davidsson, Eds. Berlin Heidelberg: Springer Verlag, July 2002, pp. 1–15.
- [7] B. Edmonds and S. Moss, "From kiss to kids - an 'anti-simplistic' modelling approach," in *MABS*, 2004, pp. 130–144.
- [8] M. Rabin, "Probabilistic automata*," *Information and control*, vol. 6, no. 3, pp. 230–245, 1963.
- [9] G. Darmon, "Sur les lois de probabilita estimation exhaustive," *CR Acad. Sci. Paris*, vol. 260, pp. 1265–1266, 1935.
- [10] B. Koopman, "On distributions admitting a sufficient statistic," *Transactions of the American Mathematical Society*, pp. 399–409, 1936.
- [11] A. Law, W. Kelton, and W. Kelton, *Simulation modeling and analysis*. McGraw-Hill New York, 1991.
- [12] K. Troitzsch, "Validating simulation models," *Networked Simulations and Simulated Networks*, pp. 265–270, 2004.
- [13] M. Richiardi, R. Leombruni, N. Saam, and M. Sonnessa, "A common protocol for agent-based social simulation," *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 1, p. 15, 2006.
- [14] F. Klügl, "A validation methodology for agent-based simulations," in *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 39–43.
- [15] J. B. McDonald, "Some generalized functions for the size distribution of income," *Econometrica*, vol. 52, no. 3, pp. 647–663, 1984.
- [16] J. McDonald and Y. Xu, "A generalization of the beta distribution with applications," *Journal of Econometrics*, vol. 66, no. 1, pp. 133–152, 1995.
- [17] H. Neave and P. Worthington, *Distribution-free tests*. Routledge London, 1989.
- [18] D. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. CRC Pr I Llc, 2004.
- [19] F. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.
- [20] F. David and N. Johnson, "The probability integral transformation when parameters are estimated from the sample," *Biometrika*, vol. 35, no. 1-2, p. 182, 1948.
- [21] G. Anastassakis, T. Panayiotopoulos, and T. Ritchings, "Virtual agent societies with the mVITAL intelligent agent system," in *Intelligent Virtual Agents*. Springer, 2001, pp. 112–125.
- [22] H. Noser and D. Thalmann, "Towards autonomous synthetic actors," *Synthetic Worlds. TL Kunii and A. Luciani, John Wiley and Sons, Ltd*, 1995.
- [23] W. Arthur, "On designing economic agents that behave like human agents," *Journal of Evolutionary Economics*, vol. 3, no. 1, pp. 1–22, 1993.
- [24] L. Chittaro and M. Serra, "Behavioral programming of autonomous characters based on probabilistic automata and personality," *Computer Animation and Virtual Worlds*, vol. 15, no. 34, pp. 319–326, 2004.
- [25] D. Thalmann, S. Musse, and M. Kallmann, "Virtual Humans' Behaviour: Individuals, Groups, and Crowds," *Proceedings of Digital Media Futures*, pp. 13–15, 1999.
- [26] P. Bécheiraz and D. Thalmann, "A behavioral animation system for autonomous actors personified by emotions," in *Proceedings of the 1998 Workshop on Embodied Conversational Characters*. Citeseer, 1998.
- [27] L. Temime, Y. Pannet, L. Kardas, L. Opatowski, D. Guillemot, and P. Bo "elle, "NOSOSIM: an agent-based model of pathogen circulation in a hospital ward," in *Proceedings of the 2009 Spring Simulation Multi-conference*. Society for Computer Simulation International, 2009, pp. 1–8.

A Survey on Coordination Methodologies for Simulated Robotic Soccer Teams

Fernando Almeida^{*†}, Nuno Lau^{†‡}, Luís Paulo Reis^{§¶}
falmeida@di.estv.ipv.pt, lau@det.ua.pt, lpreis@fe.up.pt

^{*}DI/IPV - Department of Informatics, Polytechnic Institute of Viseu, Viseu, Portugal

[†]DETI/UA - Electronics, Telecommunications and Informatics Department, University of Aveiro, Aveiro, Portugal

[‡]IEETA - Institute of Electronics and Telematics Engineering of Aveiro, Aveiro, Portugal

[§]DEI/FEUP - Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal

[¶]LIACC - Artificial Intelligence and Computer Science Laboratory, University of Porto, Porto, Portugal

Abstract—Multi-agent systems (MAS) are a research topic with ever-increasing importance. This is due to their inherently distributed organization that copes more naturally with real-life problems whose solution requires people to coordinate efforts.

One of its most prominent challenges consists on the creation of efficient coordination methodologies to enable the harmonious operation of teams of agents in adversarial environments. This challenge has been promoted by the Robot World Cup (RoboCup) international initiative every year since 1995.

RoboCup provides a pragmatic testbed based on standardized platforms for the systematic evaluation of developed MAS coordination techniques. This initiative encompasses a simulated robotic soccer league in which 11 against 11 simulated robots play a realistic soccer game that is particularly suited for researching coordination methodologies.

This paper presents a comprehensive overview of the most relevant coordination techniques proposed up till now in the simulated robotic soccer domain.

Index Terms—Coordination methodologies, MAS, simulated robotic soccer, RoboCup.

I. INTRODUCTION

The development of efficient methodologies (e.g. languages, models) for MAS coordination in adversarial environments is one of the most interesting scientific challenges promoted by the RoboCup [33] and is mainly supported by its soccer simulation leagues. The main goal of coordination mechanisms in these leagues is to adequately control a team of players and an optional coach to win matches against adversary teams.

Soccer is an inherently coordinated game in which team fitness directly relates to how well players can synchronize to perform tasks (e.g. passing). However, team coordination can be complex to achieve, mostly due to the multitude of variables (e.g. players and ball positions) players must consider to make the best decision at each instant. Moreover, measuring its success quantitatively is difficult as it doesn't necessarily relate to the final match score (e.g. a team might play better than the opposite but still lose), thus more data must be considered to perform an accurate assessment (e.g. ball possession).

The rest of the paper is organized as follows. Section II describes the RoboCup initiative and its physical soccer simulator. Section III presents a general definition of coordination and its related issues in the robotic soccer domain. Sections IV,

V, VII and VI provide a discussion of developed techniques for simulated robotic soccer organized in different perspectives. Section VIII addresses the lessons learned from the survey.

II. ROBOCUP: A TESTBED FOR COORDINATION

RoboCup was designed to meet the requirements of handling real complexities in a restricted world and provides standard challenges in a common platform to foster Artificial Intelligence and Intelligent Robotics research [17].

Its most pragmatic goal is to develop a team of fully autonomous humanoid robot soccer players capable of winning a soccer game against the winner of the World Cup by 2050. This ambition although difficult to achieve, will surely drive significant technological breakthroughs while trying [33].

The main focus of RoboCup is Robotic Soccer (RoboCup-Soccer), although other application domains exist focusing on different scopes like disaster rescue, robotics education for young students and human assistance on everyday life tasks.

The RoboCupSoccer domain has 5 leagues [11]: there is a virtual (Simulation League) and several hardware (Small-Size, Medium-Size, Standard Platform and Humanoid) leagues.

This paper focuses on the RoboCupSoccer 2D Simulation League (RoboCupSoccer2D) although other simulation subleagues (3D, 3D Development and Mixed Reality) exist. This league enables a virtual soccer match between 2 teams of 11 simulated agents each with an optional online coach using a physical soccer simulation system. Agents have an environment-aware body and can act autonomously to perform reactive or pro-active actions in an individual or sociable manner, although interaction is highly constrained as described in Section III. The environment is partially observable through non-symbolic sensors, stochastic, sequential, dynamic and multi-agent without centralized control [11].

This league presents 3 strategic research challenges for multi-agent interaction [33]:

- Multi-agent learning of individuals (e.g. ball interception) and teams (e.g. adapt player positioning to opponents);
- Teamwork to enable to real-time planning, replanning and execution of tasks in a dynamic adversary environment;
- Agent modelling to reason about others (e.g. intentions).

TABLE I
LIST OF SOCCER SERVER CORE ACTIONS BY CATEGORY

Category	Actions
Movement	Dash*, Turn, Move
Ball control	Kick, Catch, Tackle
Perception control	Turn neck, Change view, Attention to
Communication	Point to, Say
Match information	Score

*Dash impacts players stamina which is continuously assessed through their energy (liveness), effort (movement efficiency) and recovery (energy renewal rate)

Soccer Server is an open-source client/server physical soccer simulation system [36][7] used in RoboCupSoccer2D. It uses well defined protocols to enable communication between clients (players and coaches) and itself to manage connections, gather world perceptions and control clients actions.

Firstly, all clients connect to the server and sending introductory initialization data to which the server replies with the current simulation settings (e.g. player characteristics). These settings can be tweaked in order to enhance the simulation.

During the match, each team can have an online coach that receives global error-free information about world objects and all the messages sent from the players and the referee. All communication is done exclusively via the server and coach-to-players communication is highly restricted.

The simulator provides a set of players with distinguished capabilities (heterogeneous players) from which the coach must build a team to play a soccer match. During the match players receive tailored multimodal sensor information (aural, vision and body) according to their standpoint. This information is received through messages (hear, see and sense body) sent regularly from the simulator, that can be inaccurate (e.g. vision accuracy varies inversely with objects distance). Based on these perceptions, players can act upon the world to inflict changes in it using the core actions depicted in Table I.

Also during the match, a referee (automated or human) can make rulings that change the play mode (e.g. free-kick) and are immediately relayed to all clients. The human referee is used to judge situations driven by player's intentions (e.g. player obstruction) which are still difficult to evaluate automatically.

The simulation executes in discrete time steps (cycles). Throughout each step players can take actions, restricted in number and by play mode (e.g. one kick per cycle), that will be applied to objects (players and the ball) at the end of the step. The next step is simulated by applying only the allowed actions to the state information (e.g. update objects positions) and eventually by solving conflicting situations (e.g. several players might kick the ball simultaneously).

Some of the research developed has shown that robotic soccer [1] and consequently RoboCup [35][34] can be used effectively to study MAS and coordination techniques in particular. In most cases these techniques can be generalized to other domains [6] (e.g. network routing [53]).

III. COORDINATION PROBLEMS IN SIMULATED ROBOTIC SOCCER

Robotic Soccer is an instance of Periodic Team Synchronization (PTS) domains [52] in which players have sporadic

opportunities to communicate fully in a safe offline situation (e.g. in the locker-room) while being able to act autonomously in real-time with little or no communication.

One of the most important tasks for players is to select and initiate an appropriate (possibly cooperative) behavior in a given context, using (or not) knowledge from past experiences in order to help their team to win. Good coordination methodologies can help achieve this goal, although their success is still highly dependent on players individual abilities (low-level skills) to execute adequate competitive decisions.

The coordination difficulties enforced by the simulator are:

- Many multimodal information can be sensed at once, making it difficult to process;
- Environment's unpredictability makes it difficult to predict future states;
- Clients can't rely on message reception due to communication unreliability;
- Low-bandwidth makes it difficult to convey meaningful knowledge in messages;
- Uncertainty in perceived world information may lead to conflicting behaviors between agents [39], due to invalid state knowledge representations.

More specifically the simulated robotic soccer domain presents researchers with the following types of challenges:

- Perception: Where, when and how should players use their vision? To whom should they listen to? How to estimate information of others?
- Communication: What, when and how should players exchange information? How should exchanged information be used?
- Action: Which action should the player perform that is best for the team? How to evaluate different types of actions (e.g. pass vs dribble)? How to execute a given elementary (e.g. kick) or compound action (e.g. dribble)?
- Coordination: How to structure coordination dependencies between players? With whom should a player coordinate his actions? How should actions be coordinated with others? How to adapt coordination in real-time? How can the coach be used to coordinate team players?

The answer to some of these questions and others more specific will be discussed in the remaining sections.

IV. TECHNOLOGIES FOR COORDINATION

A. Coordination by Communication

Sharing pertinent world information can be useful to achieve team coordination. In earlier Soccer Server versions communication constraints were relaxed and allowed the transmission of long messages. This extremely permissive condition motivated the development of techniques that relied on sharing lots of meaningful information about the world's state knowledge among teammates to make better informed decisions.

Currently, message size is restricted to a minimum and poses a new challenge that requires the cautious selection of pertinent information to convey at each instant. To circumvent the previous constraint an Advanced Communications

framework [42] was proposed in which a player maintains a communicated world state (separated from his perceived world state) using only information from teammates, without any prediction or perception information of his own. By comparing both worlds, a player assesses the interest of items of his perceived world state to his teammates and selects the most useful information (e.g. objects positions) to share. Information utility metrics were based on domain-specific heuristics but were later extended to accommodate the current situation and estimated teammate’s knowledge [12].

Other techniques were proposed that use little or no communication by adding knowledge assumptions (e.g. Locker-Room Agreements discussed in Section VI-A) to reason over players intentions based on assigned roles [20] (combined with Coordination Graphs discussed in Section VII-A), offline learned prediction models [54] and player’s beliefs [38][16] to adapt to their actions.

The trend in this domain will be towards little or no communication due to the constraints mentioned in Section III and also because communication introduces an overhead and delay that can degrade the player performance. The combination of implicit coordination with beliefs exchange yields better performance with communication loss than explicit coordination with intentions communication alone [16]. The exchange of beliefs among teammates allows a more coherent and complete global belief about the world. This global belief can then be used to predict players utilities and adapt actions to players predicted intentions to achieve the best (joint) action. As state estimation accuracy reaches an acceptable upper bound it will eventually replace explicit communication.

B. Coordination by Intelligent Perception

The smart usage of player sensors can be an efficient way to leverage coordination with other players, by collecting the most valuable information at each instant.

During the match players can assume three types of visualizations. These are chosen using a strategic looking mechanism based on their internal world state information and the current match situation [42]:

- Ball-centered: look at the ball to react quickly to its sudden velocity changes (e.g. kick by a player);
- Active: look at the target location of a desired action (e.g. a pass to perform);
- Strategic: look at a strategic location to improve the world’s state accuracy (e.g. find an open space for a pass).

The usefulness of the information gathered using the previous approaches is different and can be classified based on its intended usage scope, validity over time and motivation for player behavior in future actions as depicted in Table II.

Ultimately, this information can be combined to enhance the player’s world state accuracy and empower better decisions.

V. POSITIONING

A. Coordination for General Positioning

The selection of a good position to move into during the match is a challenging task for players due to the unpredictable

TABLE II
COMPARISON OF DIFFERENT VISUALIZATION APPROACHES

Approach	Usage scope	Information validity period	Target behavior
Ball-centered	Individual	Short	Reactive
Active	Individual or Collective	Short to Medium	Reactive or Deliberative
Strategic	Collective	Medium to Long	Deliberative

behavior of other players and the ball. The likelihood of collaboration in a soccer match is directly related to the adequacy of a player’s position (e.g. open pass lines for attack).

During a match, at most one player can carry the ball at each instant. For this reason, players will spend most of their time without the ball and trying to figure out where to move.

The first positioning techniques proposed allowed players to situate themselves in an anticipated useful way for the team in two different contexts [48]:

- Opponent marking: player moves next to a given opponent rather than staying at his default home position;
- Ball-dependent: player adjusts his location, within a given movement range, based on the ball’s current position;
- Strategic Positioning using Attraction and Repulsion [48] (SPAR): player tries to maximize the distance to all players and minimize the distance to the opponent goal, the active teammate and the ball. This algorithm enables players to anticipate the collaborative needs of their teammates by positioning themselves to open pass lines for the teammate with the ball.

The previous techniques are rather reactive and demand fast responses from players according to the target object behavior. This leads to quickly wearing out stamina because the current match situation isn’t adequately considered. To solve these issues, techniques were proposed that distinguish between active (e.g. ball possession) and strategic match situations [42]:

- Simple Active Positioning: players always assume an active and non-strategic position (e.g. ball recovery);
- Active Positioning with Static Formation: extends the previous so that players can return to their default home position in the static formation, if there isn’t a good enough active action to perform;
- Simple Strategic Positioning: uses only one situation and one dynamic formation;
- Situation Based Strategic Positioning [44] (SBSP): defines team strategy as a set of player roles (defining their behavior) and a set of tactics composed of several formations. Each formation is used for a different strategic situation and assigns each player a default spatial positioning and a role. Contrarily to SPAR, it allows the team to have completely diverse but suitable shapes (e.g. compact for defending) for different situations and teammates to have different positional behaviors;
- Delaunay Triangulation (DT): similar in idea to SBSP, it divides the soccer field into triangles according to training data [2] and builds a map from a focal point (e.g. ball position) to a desirable positioning of each player. It

also allows the use of constraints to fix topological relations between different sets of training data to compose more flexible team formations, Unsupervised Learning Methods (e.g. Growing Neural Gas) to cope with large or noisy datasets and Linear Interpolation methods (e.g. Goraud Shading) to circumvent unknown inputs. Despite its simplicity, DT has a good approximation accuracy, is locally adjustable, fast running, scalable and can reproduce results for identical training data. On the other hand it requires much memory to store all training data and has a high cost to maintain its consistency.

Another task addressed in a soccer match is the dynamic (or flexible) positioning of team players that consists on switching players positions within a formation [48] to improve the team's performance (e.g. save player's energy for quicker responses). However, if misused it can increase player's movement (e.g. player moves across the field to occupy its new position).

The methods proposed to aid players weigh the cost/benefit ratio for deciding to switch positions are based on:

- Role Exchange: continuously assesses the usefulness of exchanging positions based on tactical gains [42] (e.g. distance to a strategic position, adequacy of next versus current position and coverage of important positions). It extends previous work that used flexible player roles with protocols for switching among them [52] to accommodate the exchange of players positions and types in the formation and has been used in conjunction with SBSP;
- Voronoi Cells: distributes players across the field and uses Attraction Vectors to reflect players' tendency towards specific objects based on the current match situation and players' roles [8]. It claims to have solved a few restrictions in SBSP (e.g. obligation to use home positions and fixed number of players for each role);
- Partial (Approximate) Dominant Regions [31]: divides the field into regions based on the players time of arrival (similar to a Voronoi diagram based on the distance of arrival), each of which shows an area that players can reach faster than others. It has been used for marked teammates to find a good run-away position.

B. Defensive Coordination

The main goal of a defending team, without ball possession, is to stop the opponent's team attack and create conditions to launch their own. In general, defensive behaviors (e.g. marking) involve positioning decisions (e.g. move to intercept the ball). Defensive positioning is an essential aspect of the game, as players without the ball will spend most of their time moving somewhere rather than trying to intercept it.

Collaborative defensive positioning has been described as a multi-criteria assignment problem where n defenders are assigned to m attackers, each defender must mark at most one attacker and each attacker must be marked by no more than one defender [23]. The Pareto Optimality principle was used to improve the usefulness of the assignments by simultaneously minimizing the required time to execute an action and the threat prevented by taking care of an attacker [24]. Threats

are considered preemptive over time and are prevented using a heuristic-criterion that considers:

- Angular size of own goal from the opponent's location;
- Distance from the opponent's location to own goal;
- Distance between the ball and opponent's location.

This technique can achieve good performances while balancing gracefully the costs and rewards involved in defensive positioning, but it doesn't seem to deal adequately with uneven defensive situations:

- Outnumbered defenders shouldn't mark specific attackers but rather position themselves in a way that difficulties their progression towards to the goal's center;
- Outnumbered attackers: more than one defender should mark an attacker (e.g. ball owner) pursuing a strategy to quickly intercept the ball or compel the opponent to make a bad decision and lose the ball.

Marking consists on guarding an opponent to prevent him from advancing the ball towards the goal, making a pass or getting the ball. Its goal is to seize the ball and start an attack.

The opponent to mark can be chosen by the player (e.g. closest opponent), by the team captain following a preset algorithm (e.g. as part of the Locker-Room Agreement [48] discussed in Section VI-A), using matching algorithms [47] or Fuzzy Logic [46]. Choosing the opponent to mark based only on its proximity isn't suitable as it disregards relevant information (e.g. teammates nearby) and will lead to poor decisions. Also, the use of a fixed centralized mediator (e.g. coach) to assign opponents to teammates although faster to compute has a negative impact in players autonomy. With the exception of PTS periods, this approach isn't robust enough due to the communication constraints mentioned in Section III and because it provides a single point of failure.

A Neural Network trained with a back-propagation algorithm that uses a linear transfer function was proposed to decide the type of marking to perform based on the distance from the player to ball, the number of opponents and teammates within the player's field of view (FoV) and the distance from the player to his own goal [46]. The output accuracy of this method could be improved by considering other relevant information that lies outside the player's FoV (e.g. nearby opponents behind the player).

Aggressive marking behavior can also be learned using a NeuroHassle policy [14] based on a neural network trained with a back-propagation variant of the Resilient Propagation (RPROP) reinforcement learning technique.

C. Offensive Coordination

To improve position selection during offensive situations (e.g. the team owns the ball) players should find the best reachable position to receive a pass or score a goal.

The Pareto Optimality Principle was applied to enable systematic decision-making regarding offensive positioning [25] based on the following set of partially conflicting criteria for simultaneous optimization [41]:

- Players must preserve formation and open spaces;

- Attackers must be open for a direct pass, keep an open path to the opponent's goal and stay near the opponent's offside line to be able to penetrate the defense;
- Non-attackers should create chances to launch the attack.

A Simultaneous Perturbation Stochastic Approximation (SPSA) combined with a RPROP learning technique (RSPSA) was proposed to Overcome the Opponent's Offside Trap (OOOT) by coordinated passing and player movements [13]. The receiver of the OOOT pass should start running into the correct direction at the right point in time, preferably being positioned right before the offside line while running at its maximal velocity when the pass is executed.

VI. TEAM COORDINATION

A. Coordination for Strategic Actions

In real soccer, team strategies are rehearsed during mundane training of team players and applied during a match. The same strategies are often used in matches, but for some opponents they must be swapped to adapt to their unexpected behavior.

Strategies typically consist on a set of tactics composed by formations that map a strategic position and a distinguished role to each player to guide his behavior.

To deal with the challenges of PTS domains a Locker Room Agreement (LRA), based in the definition of a flexible team structure (consisting of roles, formations and set-plays), can be used for players to consent on globally accessible environmental cues as triggers for changes in strategy [48]. Team strategies are communicated with a timestamp for players to recognize changes and always keep the most recent ones to disseminate to others. The team's formation can be either static or change dynamically during the match on team synchronization opportunities (e.g. kick-in) or via triggered-communication where one teammate (e.g. team captain) makes a decision and broadcasts it to his teammates.

Set-plays are predefined plans for structuring a team's behavior depending on the situation. A high-level generic and flexible framework that defines a language for set-play definition, management and execution was proposed in [29]. A set-play involves players' references (individual or role based) and steps (states of execution) that can have conditions to be carried out. Each step is lead by the ball carrying player (in charge of making the most important decisions) and can have several transitions (possibly with conditions) for subsequent steps. The main transition of a step defines a list of directives consisting of actions that should (or not) be performed. The execution of a set-play requires a tight synchronization between all participants to enable a successful cooperation. To cope with the simulator communication restrictions, only the lead player is allowed to send messages. This technique could be improved to achieve implicit coordination through a kind of belief state exchange, because the player that owns ball decides when to start the set-play and informs the involved parties. From that moment on and while the set-play follows its default path, communication among players could be dropped until a deviation is decided by the ball owner because all involved parties know the steps.

Another method proposed for high-level coordination and description of team strategies is Hierarchical Task Network (HTN) planning [37] which is to be embedded in each player. It combines high level plans (making use of previous domain knowledge to speed up the planning process) with reactive basic operators, so that players can pursue a global strategy while staying reactive to changes in the environment. This method separates the expert knowledge specified as team strategies from the player implementation making it easier to maintain. The objective of HTN is to perform tasks which can be either complex or primitive. Complex tasks are expanded into subtasks until they become primitive.

B. Hierarchical Coordination

In real life soccer, natural hierarchical relations exist among different team members and imply a leadership connotation (e.g. a coach instructs strategy to players).

A coach and trainer are privileged agents used to advise players during online games and offline work out (training) situations respectively. The need of communication from coach to players motivated the definition of coaching languages.

CLang [7] is the standard coaching language used in RoboCup since 2001 to promote a new RoboCup competition focused only on coaching techniques, but it lacks the ability to specify a team's complete behavior with sufficient detail.

Coach Unilang [43] was proposed to enable the communication of behavioral changes to players during games using different kinds of strategic information (e.g. instructions, statistics, opponent's information and definitions) based on real soccer concepts. Players can ignore received messages, interpret them as orders (must be used and will replace knowledge) or as advices (can be used with a given trust level).

Strategy Formalization Language [32] extends CLang by representing team behavior in a human-readable format easily modifiable in real-time by abstracting low-level concepts.

The main coaching techniques developed make use of:

- Neural Networks (previously trained with adequate data) to recognize opponent's team formation and provide appropriate counter formation to players [55];
- Matching Algorithms that continuously builds a table that assigns a preliminary opponent to mark to each teammate and briefs all players periodically [47].

The ability to recognize tactics and formations used by opponent teams reveals part of their strategy and can be used to implement counter strategies. To address this opportunity training techniques make use of:

- Sequential Pattern Data Mining using Unsupervised Symbolic Learning of Prediction Rules for situations and behavior during matches [26];
- Triangular Planar Graphs to build topological structures for discovering tactical behavior patterns [40].

VII. LOCAL COORDINATION

A. Coordination for Action Selection

Deciding what the player should do at a given moment in a soccer game is critical. Player's individual decision

should depend on the actions performed (or expected) of other players and balance their risks and rewards. However, these dependencies can change rapidly in dynamic environment as a result of the continuously changing state, thus efficient and scalable methods must be developed to solve this issue.

The action selection mechanisms proposed make use of:

- An idealized world model combined with observed player's state information to predict the best action [50];
- An option-evaluation architecture for different actions with comparable probabilistic scores [49];
- Player roles and a measurement opponents interference in the current situation using a multi-layer perceptron [18];
- Coordination Graphs (CGs) [19] where each node represents a player and its edges (possibly directed) define dependencies between nodes that have to coordinate their actions. This approach is based on the assumption that in most situations only a few players (typically nearby) need to coordinate their actions, while the remaining are capable of acting individually. To solve coordination dependencies in CGs algorithms like Variable Elimination (VE) [17], Max-Plus (MP) [21] and Simulated Annealing (SA) [9] were proposed. VE requires communication to always find an optimal solution but only upon termination and with a high computational cost (due to its action enumeration behavior for neighbors). MP solves VE high computational cost and makes the solution available at anytime, but it can only find near optimal solutions (except for tree-structured CGs) and restricts coordination to pairs of players. SA improves MP being able to work without communication and not restricting coordination between pairs, but it can only find approximate solutions with an associated confidence;
- Fuzzy logic and bidirectional neural networks to determine the odds and priorities of action selection based on human knowledge [57];
- Case-Based Reasoning to explicitly distinguish between controllable and uncontrollable indexing features, corresponding to players positions [45].

B. Coordination for Behavior Acquisition

Teams often use flexible (to some extent) predefined strategies set on the LRA. However they can prove fruitless, when playing against opponents that exhibit incompatible behaviors. Modelling the opponent's behavior thus becomes a necessity to allow convenient adaptation. However, as most players' are unseen for quite some time this task becomes a challenge.

With adequate models of players behavior, a player can improve his world model accuracy and consequently make better decisions by anticipating collaborative needs of teammates (e.g. open a line of pass).

Machine learning techniques have been proposed to address the issue of player adaptation to unforeseen situations [3][1].

Layered learning [48] has been proposed to enable learning low-level skills and ultimately use them to train higher-level skills that can involve coordination. The highest layer of the previous approach uses a Team-Partitioned Opaque-Transition

Reinforcement Learning (TPOT-RL) technique to allow team players to learn effective policies and thus cooperate to achieve a specific goal. This technique divides the learning task among teammates, using coarse action-dependent features and gathers rewards directly from environmental observations. It is particularly suitable for this domain which presents huge state spaces (most of them hidden) and limited training opportunities.

Policy gradient RL was proposed to coordinate decision making between a kicker and a receiver in free-kicks [30][15].

Two other important subtasks of a soccer game, Keepaway and Breakaway, have been used to study specific behavioral coordination issues. Keepaway is a game situation where one team (the keepers), tries to maintain ball possession within a limited region, while the opposing team (the takers) attempt to gain possession. Breakaway is another game situation with the purpose of the attackers trying to score goals against defenders. RL techniques have proven its their usefulness to improve decision-making in these tasks [28][51]. The recognition of the potential for RL techniques, lead to the proposal of the following methods to accelerate them:

- Preference Knowledge-Based Kernel Regression (KBKR) to give advice about preferred actions [28];
- Heuristic Accelerated Reinforcement Learning (HARL): using predefined heuristic information based on Minimax-Q [4] and Q-Learning [6];
- Case Based-HARL: heuristics are derived from a case base using Q-Learning [5].

C. Ball Passing Coordination

Passing is a crucial skill in soccer and it reflects the cooperative nature of the game. Without sophisticated passing skills, it will be difficult for a team to win a match. The number of passing possibilities for the ball carrying player can be overwhelming and thus efficient methods must be employed for real-time decision-making.

The main criteria used to decide where to pass the ball are:

- Tactical value of the pass destination;
- Chance of opponent intercepting the pass;
- Confidence on the receiver's position and interception;
- Location and orientation upon ball reception;
- Situations originated if the ball is intercepted;
- Passing travel distance;
- Initial and final player congestion on pass execution;
- Chance of providing a shoot opportunity.

Instead of relying on the previous predefined criteria that embeds the passing strategy, this strategy can be learned using Q-Learning [27].

To balance the implicit risks and gains of the previous criteria with the costs and real-time constraints of adequate decision-making developed techniques apply a weighted sum based on the player's type [42], Fuzzy logic [46] and the Pareto Optimality Principle [22].

To improve the efficiency of the previous position searching methods, a Rational Passing Decision based on Regions [56] classification (e.g. tactical, dominant, passable and falling) was proposed. Each region captures qualitative knowledge of

passing in a natural and efficient way. This technique has a low computational complexity, allows the player to decide rationally without precise information and balances success and reward of passing. However, these pros depend highly on the regions characteristics, specifically their dimension.

Voronoi Diagrams [10] were proposed to limit the number of possible meaningful passes, but are unable to find (or learn) the selection of an optimal pass.

VIII. CONCLUSION

Since the start of the RoboCup initiative, several coordination techniques were proposed that tackle core MAS coordination issues in simulated robotic soccer.

The majority of these techniques has dealt with the problem of adequate player positioning, due to its impact on the successful execution of other actions (e.g. passing) during a match. Also many of presented techniques are interdependent (e.g. CG and VE) and rely heavily on coordination technologies. In general, positioning techniques have evolved from reactive to more deliberative approaches, meaning that players now put the team's goals in front of his own because it is the only way for successful coordination to be achieved. Due to its complexity, this problem as been studied in more narrower scopes (e.g. defensive and offensive situations like opponent marking and ball passing respectively) with good results. However, situations where the number of teammates and opponents is uneven still don't seem to be adequately addressed by any of these.

Besides positioning, other techniques were proposed to cope with the remaining player's actions (e.g. marking).

Coordination technologies have evolved a lot since the start of RoboCup mostly due to added functionalities and constraints in the latest simulator releases. Although the use of communication and intelligent perception can assist team coordination through the sharing of pertinent world information and enhance the player's world state accuracy respectively, the simulator constraints discourage relying solely on them.

Team strategies are usually very complex and are typically embedded into players knowledge prior to a game (e.g. using LRA). The strategic approaches have also evolve from fixed policies to more flexible and dynamic policies that are based on real-time match information and previous opponent knowledge. Coaching was used to tweak team strategy mostly by giving advices to players and allow a quicker adaptation to opponent's behavior. Training methods have been used as a foundation to build into team members effective knowledge that can accelerate team coordination during real-time match situations (e.g. learning opponent behavior).

Action selection and behavior acquisition must rely on a good understanding of what can be achieved by intelligent perception and communication techniques.

Machine learning techniques (e.g. Q-Learning) were successfully used for behavior acquisition and adaptive coordination when faced with unpredicted constraints or situations. Due to their high computational cost and thus unfeasibility for real-time decision making, acceleration techniques must

be used to increase their efficiency and make them adequate for online usage (e.g. HARL, KBKR). It can be argued that machine learning techniques can be more accurate than hand-coding rule-based (possibly conditional) techniques.

In order to succeed, a good coordination methodology should always consider the following aspects:

- Incorporate past knowledge (e.g. using LRA) to accelerate initial decisions for usual situations, driven from direct human expertise or by offline learned prediction models. This knowledge can be tailored for specific opponents;
- Knowledge should be adaptable according to opponent behavior in real-time;
- Use alternative techniques to complement and replace technologies based on communication and perception.

ACKNOWLEDGMENT

This work was financially supported by Polytechnic Institute of Viseu under a PROFAD scholarship.

REFERENCES

- [1] A. Agah and K. Tanie, 'Robots Playing to Win: Evolutionary Soccer Strategies', in *IEEE ICRA*, volume 1, pp. 632–637, Albuquerque, NM, USA, (1997). IEEE.
- [2] H. Akiyama and I. Noda, 'Multi-Agent Positioning Mechanism in the Dynamic Environment', in *RoboCup 2007: Robot Soccer World Cup XI*, eds., U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, volume 5001 of *LNAI*, 377–384, Springer, Berlin, (2008).
- [3] T. Andou, 'Refinement of Soccer Agents' Positions using Reinforcement Learning', in *RoboCup-97: Robot Soccer World Cup I*, ed., H. Kitano, volume 1395 of *LNAI*, 373–388, Springer-Verlag, Berlin, (1998).
- [4] R. Bianchi, C. Ribeiro, and A. Costa, 'Heuristic Selection of Actions in Multiagent Reinforcement Learning', in *IJCAI-07*, pp. 690–696, Hyderabad, India, (2007). Morgan Kaufmann Publishers Inc.
- [5] R. Bianchi, R. Ros, and R. Mantaras, 'Improving Reinforcement Learning by Using Case Based Heuristics', in *Case-Based Reasoning Research and Development*, eds., L. McGinty and D. Wilson, volume 5650 of *LNAI*, 75–89, Springer, Seattle, WA, (2009).
- [6] L. Celiberto and J. Matsuura, *Robotic Soccer: The Gateway for Powerful Robotic Applications*, volume 2 of *Proceedings of ICINCO-2006*, IST, IC&C, Setubal, 2008.
- [7] M. Cheny, K. Dorer, E. Foroughi, F. Heintz, Z. Huangy, S. Kapetanakis, K. Kostiadis, J. Kummeneje, J. Murray, I. Noda, O. Obst, P. Riley, T. Stevens, Y. Wangy, and X. Yiny, *RoboCup Soccer Server Users Manual*, For Soccer Server Version 7.07 and later, The RoboCup Federation, 2003.
- [8] H. Dashti, N. Aghaeepour, S. Asadi, M. Bastani, Z. Delafkar, F. Disfani, S. Ghaderi, S. Kamali, S. Pashami, and A. Siahipirani. Dynamic Positioning based on Voronoi Cells (DPVC), July 2005 2005.
- [9] J. Dawei and W. Shiyuan, 'Using the Simulated Annealing Algorithm for Multiagent Decision Making', in *RoboCup 2006: Robot Soccer World Cup X*, eds., G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, volume 4434 of *LNAI*, 110–121, Springer, Berlin, (2007).
- [10] H. Endert, T. Karbe, J. Krahmann, and F. Trollmann. Dainamite - Team Description, 2009.
- [11] RoboCup Federation. RoboCup: Overview, 01-10-2010 2010.
- [12] R. Ferreira, L. Reis, and N. Lau, 'Situation Based Communication for Coordination of Agents', in *Scientific Meeting of the Portuguese Robotics Open*, eds., L. Reis, A. Moreira, E. Costa, P. Silva, and J. Almeida, pp. 39–44, Porto, (2004). FEUP Edições.
- [13] T. Gabel and M. Riedmiller. Brainstormers 2D - Team Description, 2009.
- [14] T. Gabel, M. Riedmiller, and F. Trost, 'A Case Study on Improving Defense Behavior in Soccer Simulation 2D: The Neurohassle Approach', in *RoboCup 2008: Robot Soccer World Cup XII*, eds., L. Iocchi, H. Matsubara, A. Weitzenfeld, and C. Zhou, volume 5399 of *LNCS*, 61–72, Springer, Berlin, (2009).

- [15] H. Igarashi, K. Nakamura, and S. Ishihara, 'Learning of Soccer Player Agents using a Policy Gradient Method: Coordination between Kicker and Receiver during Free Kicks', in *IJCNN*, ed., X. He, H. and Xu, pp. 46–52, Hong Kong, (2008). IEEE.
- [16] M. Isik, F. Stulp, G. Mayer, and H. Utz, 'Coordination without Negotiation in Teams of Heterogeneous Robots', in *RoboCup 2006: Robot Soccer World Cup X*, eds., G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, volume 4434 of *LNAI*, 355–362, Springer, Berlin, (2007).
- [17] W. Jin, W. Tong, W. Xiao, and M. Xiangping, 'Multi-Robot Decision Making based on Coordination Graphs', in *ICMA*, pp. 2393–2398, (2009).
- [18] H. Kim, H. Shim, M. Jung, and J. Kim. Action Selection Mechanism for Soccer Robot, 1997.
- [19] J. Kok, M. Spaan, and N. Vlassis, 'Multi-Robot Decision Making using Coordination Graphs', in *11th ICAR*, eds., A. Almeida and U. Nunes, pp. 1124–1129, Coimbra, Portugal, (2003).
- [20] J. Kok, M. Spaan, and N. Vlassis, 'Non-Communicative Multi-Robot Coordination in Dynamic Environments', *Robotics and Autonomous Systems*, **50**(2-3), 99–114, (2005).
- [21] J. Kok and N. Vlassis, 'Using the Max-Plus Algorithm for Multiagent Decision Making in Coordination Graphs', in *RoboCup 2005: Robot Soccer World Cup IX*, eds., A. Bredendfeld, A. Jacoff, I. Noda, and Y. Takahashi, volume 4020 of *LNAI*, 359–360, Springer, Berlin, (2005).
- [22] V. Kyrlyov, 'Balancing Gains, Risks, Costs, and Real-Time Constraints in the Ball Passing Algorithm for the Robotic Soccer', in *RoboCup 2006: Robot Soccer World Cup X*, eds., G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, volume 4434 of *LNAI*, 304–313, Springer, Berlin, (2007).
- [23] V. Kyrlyov and E. Hou, 'While the Ball in the Digital Soccer is Rolling, where the Non-Player Characters should go in a Defensive Situation?', in *Future Play*, eds., B. Kapralos, M. Katchabaw, and J. Rajnovich, pp. 90–96, Toronto, Canada, (2007). ACM.
- [24] V. Kyrlyov and Eddie Hou, 'Pareto-Optimal Collaborative Defensive Player Positioning in Simulated Soccer', in *RoboCup 2009: Robot Soccer World Cup XIII*, eds., J. Baltes, M. Lagoudakis, T. Naruse, and S. Shiry, volume 5949 of *LNAI*, Springer, Berlin, (2010).
- [25] V. Kyrlyov and S. Razykov, 'Pareto-Optimal Offensive Player Positioning in Simulated Soccer', in *RoboCup 2007: Robot Soccer World Cup XI*, eds., U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, volume 5001 of *LNAI*, 228–237, Springer, Berlin, (2008).
- [26] A. Latner, A. Miene, U. Visser, and O. Herzog, 'Sequential Pattern Mining for Situation and Behavior Prediction in Simulated Robotic Soccer', in *9th RoboCup International Symposium*, eds., A. Latner, A. Miene, U. Visser, and O. Herzog, Osaka, Japan, (2005).
- [27] X. Li, W. Chen, J. Guo, Z. Zhai, and Z. Huang, 'A New Passing Strategy based on Q-Learning Algorithm in RoboCup', in *ICCSSE*, volume 1, pp. 524–527. IEEE, (2008).
- [28] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild, 'Giving Advice about Preferred Actions to Reinforcement Learners via Knowledge-Based Kernel Regression', in *AAAI-05 and IAAI-05*, eds., M. Veloso and S. Kambhampati, pp. 819–824, Pittsburgh, Pennsylvania, (2005). AAAI Press / The MIT Press.
- [29] L. Mota and L. Reis, 'Setplays: Achieving Coordination by the Appropriate use of Arbitrary Pre-Defined Flexible Plans and Inter-Robot Communication', in *ROBOCOMM-2007*, pp. 1–7, Athens, (2007). IEEE Press.
- [30] K. Nakamura and H. Igarashi, 'Learning of Decision Making at Free Kicks using Policy Gradient Methods', in *Robotics and Mechatronics*, (2005).
- [31] R. Nakanishi, K. Murakami, and T. Naruse, 'Dynamic Positioning Method Based on Dominant Region Diagram to Realize Successful Cooperative Play', in *RoboCup 2007: Robot Soccer World Cup XI*, eds., U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, volume 5001 of *LNAI*, 488–495, Springer, Berlin, (2008).
- [32] A. Nie, A. Hönemann, A. Pegam, C. Rogowski, L. Hennig, M. Diedrich, P. Hügelmeyer, S. Buttlinger, and T. Steffens, 'ORCA - Osnabrueck RoboCup Agents Project', Technical report, Institute of Cognitive Science, (2004).
- [33] I. Noda, M. Asada, H. Matsubara, M. Veloso, and H. Kitano, 'RoboCup as a Strategic Initiative to Advance Technologies', in *IEEE ICSMC*, volume 6, pp. 692–697, Tokyo, Japan, (1999). IEEE Press.
- [34] I. Noda, H. Matsubara, K. Hiraki, and I. Frank, 'Soccer Server: A Tool for Research on Multi-Agent Systems', *Applied Artificial Intelligence*, **12**(2-3), 233–250, (1998).
- [35] I. Noda and P. Stone, 'The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS research', *Autonomous Agents and Multi-Agent Systems*, **7**(1-2), 101–120, (2003).
- [36] I. Noda, S. Suzuki, H. Matsubara, M. Asada, and H. Kitano. Overview of RoboCup-97, 1998.
- [37] O. Obst and J. Boedecker, 'Flexible Coordination of Multiagent Team Behavior using HTN Planning', in *RoboCup 2005: Robot Soccer World Cup IX*, eds., I. Noda, A. Jacoff, A. Bredendfeld, and Y. Takahashi, 521–528, Springer, Berlin, (2006).
- [38] E. Pagello, A. D'Angelo, F. Montesello, F. Garelli, and C. Ferrari, 'Cooperative Behaviors in Multi-Robot Systems through Implicit Communication', *Robotics and Autonomous Systems*, **29**(1), 65–77, (1999).
- [39] J. Penders, 'Conflict-based Behaviour Emergence in Robot Teams', in *Conflicting Agents: Conflict Management in Multi-Agent Systems*, Multiagent Systems, Artificial Societies, and Simulated Organizations: International Book Series, 169–202, Kluwer Academic Publishers, Norwell, (2001).
- [40] F. Ramos and H. Ayanegui, 'Discovering Tactical Behavior Patterns supported by Topological Structures in Soccer Agent Domains', in *AAMAS-2008*, eds., L. Padgham, D. Parkes, J. Müller, and S. Parsons, volume 3, pp. 1421–1424, Estoril, Portugal, (2008). IFAAMAS.
- [41] S. Razykov and V. Kyrlyov, 'While the Ball in the Digital Soccer is Rolling, where the Non-Player Characters should go if the Team is Attacking?', in *Future Play*, Ontario, Canada, (2006). ACM.
- [42] L. Reis, *Coordination in Multi-Agent Systems: Applications in University Management and Robotic Soccer*, Phd, 2003.
- [43] L. Reis and N. Lau, 'Coach UNILANG - A Standard Language for Coaching a (Robo)Soccer Team', in *RoboCup 2001: Robot Soccer World Cup V*, eds., A. Birk, S. Coradeschi, and S. Tadokoro, volume 2377 of *LNAI*, 183–192, Springer, Berlin, (2002).
- [44] L. Reis, N. Lau, and E. Oliveira. Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents, 2001.
- [45] R. Ros, J. Arcos, R. de Mantaras, and M. Veloso, 'A Case-based Approach for Coordinated Action Selection in Robot Soccer', *Artificial Intelligence*, **173**(9-10), 1014–1039, (2009).
- [46] M. Simões, B. Silva, A. Cerqueira, and L. Silva. Bahia2D - Team Description, 2009.
- [47] F. Stolzenburg, J. Murray, and K. Sturm, 'Multiagent Matching Algorithms with and without Coach', *Decision Systems*, **15**(2-3), 215–240, (2006).
- [48] P. Stone, *Layered Learning in Multi-Agent Systems*, Phd, 1998.
- [49] P. Stone and D. McAllester, 'An Architecture for Action Selection in Robotic Soccer', in *AAMAS-06*, pp. 316–323, Montreal, Quebec, Canada, (2001). ACM.
- [50] P. Stone, P. Riley, and M. Veloso, 'Defining and using Ideal Teammate and Opponent Agent Models', in *IAAI-00*, (2000).
- [51] P. Stone, R. Sutton, and G. Kuhlmann, 'Reinforcement Learning for RoboCup Soccer Keepaway', *Adaptive Behavior*, **13**(3), 165–188, (2005).
- [52] P. Stone and M. Veloso, 'Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork', *Artificial Intelligence*, **110**(2), 241–273, (1999).
- [53] P. Stone and M. Veloso. Team-Partitioned, Opaque-Transition Reinforcement Learning, 1999.
- [54] F. Stulp, M. Isik, and M. Beetz, 'Implicit Coordination in Robotic Teams using Learned Prediction Models', in *ICRA*, IEEE ICRA, 1330–1335, IEEE, New York, (2006).
- [55] U. Visser, C. Drucker, S. Hubner, E. Schmidt, and H. Weland, 'Recognizing Formations in Opponent Teams', in *RoboCup 2000: Robot Soccer World Cup IV*, eds., P. Stone, T. Balch, and G. Kraetzschmar, volume 2019 of *LNAI*, 391–396, Springer-Verlag, Berlin, (2001).
- [56] X. Yuan and T. Yingzi, 'Rational Passing Decision Based on Region for the Robotic Soccer', in *RoboCup 2007: Robot Soccer World Cup XI*, eds., U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, volume 5001 of *LNAI*, 238–245, Springer, Berlin, (2008).
- [57] R. Zafarani and M. Yazdchi, 'A Novel Action Selection Architecture in Soccer Simulation Environment using Neuro-Fuzzy and Bidirectional Neural Networks', *International Journal of Advanced Robotic Systems*, **4**(1), 93–101, (2007).

ELDAMeth: A Methodology For Simulation-based Prototyping of Distributed Agent Systems

Giancarlo Fortino and Wilma Russo

Department of Electronics, Informatics and Systems (DEIS)

University of Calabria

Via P. Bucci, cubo 41C, I-87036 Rende (CS), Italy

{g.fortino, w.russo}@unical.it

Abstract—In application domains, such as distributed information retrieval, content management and distribution, e-Commerce, the agent-based computing paradigm has been demonstrated to be effective for the analysis, design and implementation of distributed software systems. In particular, several agent-oriented methodologies, incorporating suitable agent models, frameworks and tools, have been to date defined to support the development lifecycle of distributed agent systems (DAS). However, few of them provide effective methods for dynamic validation to analyze design objects at different degrees of refinement before their actual implementation and deployment. In this paper, ELDAMeth, a simulation-based methodology for DAS that enables rapid prototyping based on visual programming, automatic code generation and dynamic validation, is presented. ELDAMeth can be used both stand-alone for the modeling and evaluation of DAS and coupled with other agent-oriented methodologies for enhancing them with simulation-based validation. In particular, the proposed methodology is based on the ELDA (Event-driven Lightweight Distilled StateCharts-based Agents) agent model, and provides key programming abstractions (event-driven computation, multi-coordination, and coarse-grained strong mobility) very suitable for highly dynamic distributed computing and on a CASE tool-driven iterative process fully supporting the modeling, simulation, and implementation phases of DAS. A simple yet effective case study in the distributed information retrieval domain is used to illustrate the proposed methodology.

Keywords – agent oriented software engineering; simulation; CASE tools; mobile agents; multi-coordination; statecharts

I. INTRODUCTION

The ubiquitous diffusion and usage of the Internet have promoted the development of new kinds of distributed applications characterized by a huge number of participants, high decentralization of software components and code mobility, which are typical of application domains such as distributed information retrieval, content management and distribution, and e-Commerce. In these application domains, the agent-based computing paradigm [19] has been demonstrated to be effective for the analysis, design and implementation of distributed software systems. In particular, in the context of the agent-oriented software engineering (AOSE), several agent-oriented methodologies based on suitable agent models, frameworks and tools, have been defined to support the development lifecycle of distributed

agent systems (DAS). The key elements, identified through an in-depth analysis of such methodologies, for the provision of an effective development of distributed agent systems are the agent model, the development methodology and the supporting CASE tool.

The agent models aim at providing abstractions for the modelling of the agent behavior and interactions. Basically they can be classified in two large groups: (i) models based on intelligent agent architectures [19, 21] ranging from reactive agents (e.g. Brook's subsumption architecture) to deliberative agents (e.g. BDI agents); (ii) models based on the mobile active object concept encompassing mobile agent architectures [4]. Models of the first group are mainly oriented to problem-solving, planning and reasoning systems whereas models of the second group are more oriented to distributed computation in open and dynamic environments like the Internet. In the context of Internet computing, agent models and related frameworks based on lightweight architectures, asynchronous messages/events and state-based programming such as JADE [2], Bond [3], and Actors [1], have demonstrated great effectiveness for modeling and programming agent-based distributed applications. In particular, such models define suitable abstractions for the modelling of reactivity and proactiveness of agent behaviors and interactions. However, they mainly consider messages (and related message-based protocols and infrastructures) as a means of interaction among agents and mobility as an auxiliary feature of agents. The exploitation of coordination models and infrastructures based not only on messages but also on events, tuples, blackboards and other coordination abstractions [6] can provide more effectiveness in designing complex agent interactions and more efficiency in their actual implementation. Moreover, mobility, which can provide a powerful means for dynamic organization of distributed components modeled as mobile agents, also enables and demands for new non-message-based coordination models.

The agent-oriented development methodologies aim at supporting the development lifecycle of agent-based systems from analysis to deployment and maintenance. They can be classified into general-purpose and domain-specific methodologies. The general-purpose methodologies such as Gaia [27], PASSI [7], Tropos [5], Ingenias [23] are suitable for the development of multi-agent systems in different application domains whereas the domain-specific methodologies can be

more effectively exploited in a given, very specific application domain. Apart from their context of use, they are all based on a meta-model of multi-agent system, which loosely or tightly depends on a reference agent model, and on a phase-based iterative development process. Agent oriented methodologies for Internet-based distributed agent systems should incorporate not only a MAS meta-model and its related agent model suitable for distributed computation but also effective prototyping methods able to validate the design models before their implementation and deployment in a large-scale distributed testbed. In particular, dynamic validation based on simulation is emerging as a powerful means for functional and non functional validation of designed agent systems in a large-scale controlled environment. To date a few agent-oriented, simulation-based development methodologies have been proposed in the literature, such as Electronic Institutions [26], DynDEVS/James [17], CaseLP [20], GAIA/MASSIMO [12], PASSIM [8], TuCSoN/pi [16], Joint Measure [25], Ingenias/Repast [24]. They incorporate simulation to support the design phase of the MAS development lifecycle with the main focus on the validation and performance evaluation of the designed MAS model. Moreover, the importance of two additional features of agent-oriented methodologies, high degree of integration with other methodologies and availability of a CASE tool supporting the process phases, has become relevant in the AOSE community. The former feature would allow for an easy integration with other methodologies for the purpose of enriching already existing methodologies or creating new and more effective ones. The latter would allow for automating the development process phases and their transitions so providing more robust development and rapid prototyping.

In this paper we propose a novel methodology, named ELDAMeth, which provides all the aforementioned important features for the development of DAS: effective agent model for distributed computing systems, simulation-based agent-oriented methodology, integration with other methodologies, and CASE tool support. In particular, ELDAMeth relies on the ELDA (Event-driven Lightweight Distilled Statecharts Agents) agent model and related frameworks and tools, and on an iterative development process seamlessly covering the modeling, simulation and implementation phases of DAS and supported by a visual CASE tool. ELDAMeth can be used both stand-alone and in conjunction/integration with other agent-oriented methodologies which provide support to the analysis, (high-level) design, and implementation phases. ELDAMeth is exemplified through a case study concerning distributed information retrieval based on mobile agents.

The rest of the paper is organized as follows. Section II presents ELDAMeth, providing an overview of the modeling abstractions and tools, whereas the simulation phase of ELDAMeth is detailed in section III. In section IV the case study is described from modeling to simulation. Finally conclusions are drawn and future work anticipated.

II. ELDAMETH: A SIMULATION-BASED PROTOTYPING METHODOLOGY FOR DAS

ELDAMeth is a methodology specifically designed for the simulation-based prototyping of DAS. It is based on the

ELDA (Event-driven Lightweight Distilled StateCharts Agent) agent model and related frameworks and tools, and on an iterative development process covering modeling, simulation and implementation phases of DAS. ELDAMeth can be used both *stand-alone* and in conjunction/integration with other agent-oriented methodologies which support the analysis and (high-level) design phases. In Figure 1, the development process of ELDAMeth is represented which consists of the following three phases:

- The *Modeling* phase produces an ELDA-based MAS design object that is a specification of a MAS fully compliant with the ELDA MAS meta-model. The design object can be produced either by (i) the ELDA-based modeler which uses the ELDA MAS meta-model and the ELDATool [11, 9], a CASE tool supporting the development phases of ELDA-based MAS, or by (ii) translation and refinement of design objects produced by other agent-oriented methodologies such as PASSI [7, 8], GAIA [27, 12], MCP [14], and others [23, 5, 19]. In particular, while the translation process centers on (semi) automatic model transformations based on the MAS meta-model of the employed methodology and the ELDA MAS meta-model, the refinement process is usually carried out manually by the ELDA-based Modeler by using the ELDATool. The defined design objects can be automatically translated, through the ELDATool, into ELDA-based MAS code objects according to the ELDAFramework, which is a set of Java classes formalizing all the modeling abstractions of the ELDA MAS meta-model. The code objects are then used in the *Simulation* phase.
- The *Simulation* phase produces the Simulation Results in terms of MAS execution traces and performance indices that must be carefully evaluated with respect to the identified functional and non-functional requirements. Such evaluation can lead to a further iteration step which starts from a new (re)modeling activity. In particular, the Simulation Results come from the execution of the ELDA-based MAS simulation object carried out through ELDASim, a Java-based event-driven simulation framework for ELDA agents. The simulation object is obtained by synthesizing the ELDA-based MAS code object with the simulation parameters and performance indices, defined on the basis of the requirements, by means of ELDASim.
- The *Implementation & Deployment* phase produces code for the JADE framework which can be then deployed and executed on a distributed JADE platform. Starting from the ELDA-based MAS design object, the code production is supported by the JADE-based DistilledStateChartBehaviour framework [15], the JADE framework and the ELDATool. Of course, the execution results can be evaluated against the functional and non functional requirements and, possibly, trigger a new iteration.

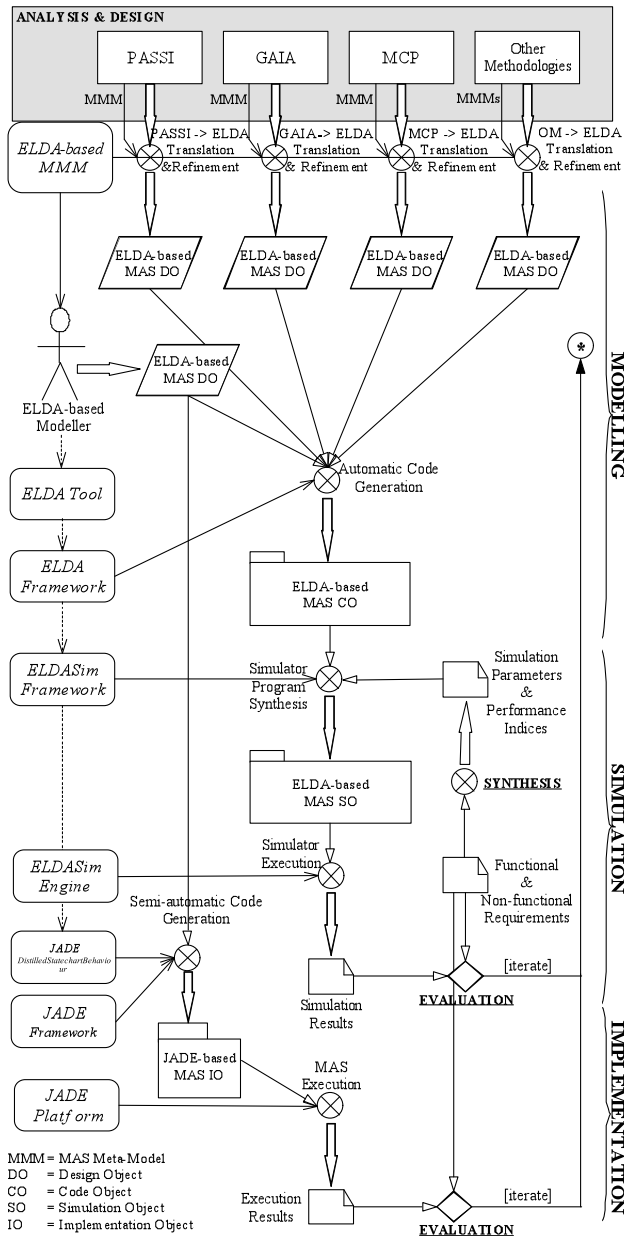


Figure 1. The ELDA Meth iterative development process.

In the following subsection a description of the ELDA-based modeling abstractions and tools is given (more details can be found in [13]) and the simulation phase is .

A. Modeling ELDA-based MAS

The modeling of agent-based systems based on the ELDA model is carried out through the ELDA MAS meta-model (ELDA MMM) which was specifically defined to provide design abstractions particularly suitable for DAS and specifically concerning agent lightness, multi-coordination and mobility. However, as agent-system domains could require specific design abstractions which haven't been

originally included within the ELDA MMM, the structure of the ELDA MMM was designed to be extensible; in fact, ELDA MMM makes it possible to introduce new design abstractions (such as new services providers or new coordination spaces) which characterize a specific execution environment. In particular, the ELDA MMM is structured according to the view-based schema reported in Figure 2:

- *Agent View*, which represents the structure of an ELDA agent and its relationships with the coordination and system spaces. ELDA agents are event-driven lightweight agents that are a single-threaded autonomous entity interacting through asynchronous events, executing upon reaction, and capable of migration [13].
- *Event View*, which represents the structure of events. Events formalize both self-triggering events (Internal events) and requests to or notifications from the local agent server (Management, Coordination and Exception events). Events are further classified into OUT-events which are generated by the agent and always target the local agent server and IN-events which are generated by the local agent server and delivered to target agents.
- *SystemSpace View*, which represents the structure of the system space. The System Space provides system services for the management of agent lifecycles, timers and resources (e.g. consoles, databases, files, sensors). Agents interact with the System Space through Management events.
- *CoordinationSpace View*, which represents the hierarchy of the coordination spaces. The Coordination Space represents a local or global coordination structure based on a given coordination model through which agents interact. Several coordination spaces are currently defined such as message-based, local tuple space, publish/subscribe. Agents interact with the Coordination Space through Coordination events.
- *DSC View*, which represents the structure of a DSC, basically a hierarchical state machine with history pseudostates [13].
- *FIPATemplate View*, which represents the structure of the FIPA agent template [10] of the ELDA agent behavior.

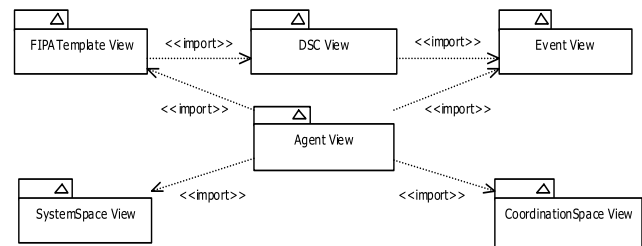


Figure 2. The ELDA MMM schema.

Models designed through the ELDA MMM can be coded through the ELDAFramework that is an object-oriented framework enabling developers to implement an ELDA-based application as it offers the implementation abstractions

representing the modeling concepts offered by the ELDA MMM.

To facilitate the use of ELDAMeth, an integrated development environment, named ELDATool [9, 11], is offered. It aims to support developers during the modelling, simulation, and implementation phases. In particular, ELDATool provides in an integrated fashion:

- a visual editor which allows to model behavior, interaction and mobility aspects of an agent-system according to the ELDA model;
- an automatic translator which implements the translation rules from ELDA meta-model to ELDAFramework;
- a visual editor to configure simulation parameters used to generate a simulation program based on ELDASim framework (see next sub section);
- an automatic translator which implements the translation rules from hybrid JADE and ELDA meta-models to the JADE DistilledStateChartsBehaviour.

The graphical design models are serialized into XML-like files. The tool also offers the functionality of automatic code generation by translating the XML-like files produced after the Modelling phase into Java code based on the ELDAFramework for simulation purposes or on the JADE framework for real execution.

Finally, to support the Simulation phase ELDATool offers a visual editor to configure simulation parameters which are used to generate the simulation program according to the ELDASim framework (see next subsection).

Currently, the ELDATool is implemented in Java as a collection of Eclipse plug-in to exploit several frameworks which fully support the development of visual editors; moreover, the high diffusion of Eclipse in the research community makes the tool immediately available to the Eclipse users and the learning process of the tool is therefore quicker.

B. Simulation of ELDA-based MAS

The development process of ELDAMeth includes a simulation phase (Figure 3) which consists of the following three activities:

1. *Performance Indices Definition*, which, on the basis of functional and non functional requirements, produces the definition of the performance indices which will be evaluated during the simulation;
2. *Simulation Implementation*, which aims at the realization of a simulation program which takes into account the previously identified indices, the definition of the controlled environment and the ELDAFramework-based DAS implementation. In particular, such program uses abstractions provided by the ELDASim (see below) to define:
 - the controlled execution environment (both features characterizing the computational nodes and the network) which mirrors the real execution environment;
 - the initial DAS configuration (agents and related locations);

3. *Simulation Execution*, which consists of the DAS execution within the controlled execution environment and of the collection of the defined performance indices which allow the analysis and the validation of the DAS under-development.

The simulation phase can be iteratively executed to modify, according to obtained simulation results, the modelling choices taken in former iterations. Simulation execution is supported by the ELDA simulation environment (ELDASim) which is a Java-based execution environment for ELDA agents that aims to validate and evaluate through simulation an ELDA model based solutions with respect to efficacy and efficiency aspects. To accomplish this, ELDASim is equipped with:

- The basics mechanisms of the distributed architectures supporting ELDA agents. In particular, agent servers, the network interconnecting agent servers, and several kinds of coordination infrastructures (asynchronous message-based, publish/subscribe, and tuple spaces) for fully supporting the distinctive multi-coordination feature of the ELDA model.
- The simulation of accomplishment time of time-consuming operations such as agent actions, agent management operations, coordination acts, and agent migrations.
- The capture of the traces of interactions (among agents and between agents and agent servers) in terms of exchanged events, filtered in an application-specific fashion.

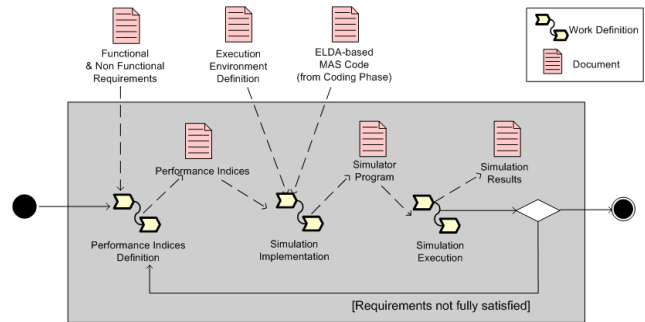


Figure 3. Schema of the Simulation phase.

III. A CASE STUDY: MOBILE AGENT-BASED DISTRIBUTED INFORMATION RETRIEVAL

In this section, a simple yet effective case study concerning with a distributed information retrieval task in a distributed computing system is proposed to exemplify ELDAMeth. In particular, the task consists in searching for specific information located exactly in one location within a network of federated information locations. The defined high-level solution is based on a coordinated set (or task force) of mobile agents which carry out the information searching task. A user (represented by an owner agent) starts searching by creating and launching a task force of mobile agents (called searcher agents) onto different random locations. As soon as the task force finds the desired information, the owner agent is notified

with the found information. A high-level design of such prototypical solution, which is to be properly translated and refined, is provided by the Multi-Coordination Process (MCP) [14]. In the following subsections the case study is described starting from the high-level modeling provided by MCP and then proceeding with the modeling, simulation and implementation phases.

1) *MCP-based high-level modeling*

The Multi-Coordination based Process (MCP) [14] is iterative and consists of the two phases (Modeling and Evaluation). The Modeling phase, on the basis of a coordination statement (CS) which derives from a preliminary analysis and includes a description of the agents along with their interactions (coordination requirements - CRs), and a set of coordination properties (CPs), provides alternative coordination solutions which fulfill the CS. In the Evaluation phase, a specific solution is chosen among such alternative coordination solutions which are evaluated through simulation and then compared on the basis of ad-hoc defined performance indices (e.g. time and resource consumption).

With reference to the case study, the proposed solutions for the coordination of the task force during its information retrieval task is based on the following CRs:

- CR₁: every time a searcher agent visits a location not yet searched by other agents of the same task force, it notifies the other agents that such location has already been searched so

avoiding unnecessary and resource-consuming duplicate searches;

- CR₂: as soon as a searcher agent finds the desired information on a given location, it reports the found information to the owner agent;

- CR₃: when a searcher agent finds the desired information on a given location, it signals such event to all the other searcher agents to stop them;

and on the following CPs:

- CP_a: the task force is constituted by at least two searcher agents;

- CP_b: the agents of the task force may or may not know each other whereas they know the identity of the owner agent and vice-versa;

- CP_c: the interactions among all the agents (searcher and owner) are always asynchronous.

- CP_d: the interactions required by CR₁ may be local or remote, that required by CR₂ and CR₃ are remote.

The defined solutions are reported in Figure 4. For each solution, the three coordination requirements are addressed by suitable interaction patterns (IPs) and related coordination models (CMs).

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR ₁	LBN [2..N, known, local, async]	QAMP	When a searcher agent searches in a location which has not been already searched by another agent of its task force, it creates a marker agent, stationing in this location, which can locally signal the other agents of its task force that a search has already carried out. As soon as an agent visits a location looks for a marker agent of its task force to avoid searching.
CR ₂	R2O [2, known, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR ₃	GBN [2..N, known, remote, async]	QAMP	A searcher agent which has found the desired information sends a notification message to all the other searcher agents of the task force to stop them.

(A)

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR ₁	GBN [2..N, known, remote, async]	QAMP	A searcher agent to notify that it has searched a given location sends a message containing the location identifier to all the other searcher agents of the task force.
CR ₂	R2O [2, known, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR ₃	GBN [2..N, known, remote, async]	QAMP	A searcher agent which has found the desired information sends a notification message to all the other searcher agents of the task force to stop them.

(B)

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR ₁	LBN [2..N, unknown, local, async]	LTS	When a searcher agent searches in a location which has not been already searched by another agent of its task force, it inserts a signaling tuple into the LTS to signal that this location has been searched. As soon as an agent visits a location and reads the signaling tuple, it avoids searching.
CR ₂	R2O [2, known, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR ₃	GBN [2..N, known, remote, async]	TPS	When a searcher agent finds the desired information, it publishes an event of a specific topic related to its task force which signals the stop of the retrieval task. All the other agents of the task force will be thus asynchronously notified since they subscribed to the specific topic at creation time.

(C)

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR ₁	GBN [2..N, unknown, remote, async]	TPS	A searcher agent to notify that it has searched a given location publishes an event of a specific topic related to its task force and containing the location identifier. All the other agents of the task force will be thus asynchronously notified since they subscribed to the specific topic at creation time.
CR ₂	R2O [2, known, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR ₃	GBN [2..N, known, remote, async]	TPS	When a searcher agent finds the desired information, it publishes an event of a specific topic related to its task force which signals the stop of the retrieval task. All the other agents of the task force will be thus asynchronously notified since they subscribed to the specific topic at creation time.

(D)

Figure 4. The result of the high-level MCP-based modeling: solutions A, B, C, D.

The IPs are selected from a repository containing some of the most used agent-oriented interaction patterns and specifically characterized (through the tuple [number of participants, participant identity, locus, temporality]) according to the coordination requirements and properties. In particular, the characterized IPs are:

- Location-based notification (LBN), which involves agents passing through a given location to be notified about events occurring/occurred in such location.
- Report to owner (R2O), which involves a child agent reporting to its owner agent when its task is completed.
- Group-based notification (GBN), which involves an agent notifying all agents of its group when a given event occurs.

The CMs used to implement the IPs are:

- Local Linda-like tuple space (LTS), which supports a high number of participants, allows temporal decoupling but only offers local interaction [22].
- Topic-based publish/subscribe (TPS), which supports a high number of participants, allows for distributed

interactions and does not require temporal coupling between participants [18].

- Queue-based unicast asynchronous message passing (QAMP), which supports a variable number of participants, allows for both local and remote interactions, does not require temporal coupling, but requires spatial coupling among participants [28].

2) ELDA-based Modeling

The four solutions have been modeled according to the ELDA MMM and integrated into an ELDA_{Sim}-based simulator program. In Figure 5 the SearcherAgent behavior of the solution A (see Fig.4A) is shown; such solution is representative of the message-based solution, whereas the fully multi-coordinated solution can be found in [13].

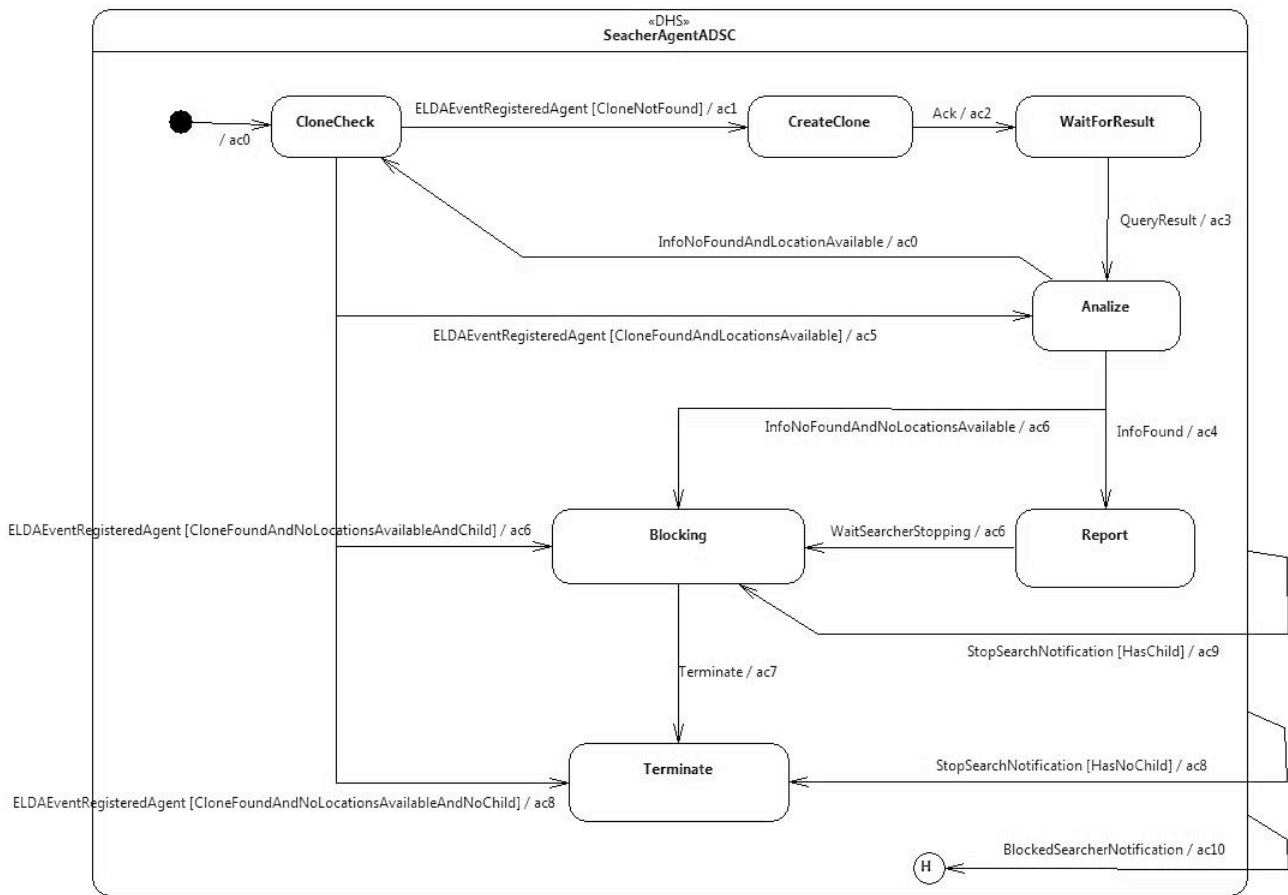


Figure 5. SearcherAgent behavior of solution A

Every time a Searcher Agent (SA) visits a new location, it checks for the presence of a marker agent by using the whitepage service made available by the agent server (action

ac0) and behaves as follows:

1. If no marker agent is present, the SA creates the marker agent (action ac1), submits the query to the servant agent

(action ac2), and waits for the query result to analyze it (action ac3). If no info is found the SA moves to a new location (if available); otherwise the SA goes into a pseudo-termination state (BLOCKING). If the info is found, the SA notifies the user agent and the other members of the task force through asynchronous messages (action ac4); then, it goes into the BLOCKING state. The user agent will thus receive a Report event whereas the taskforce members the StopSearchNotification event. Upon reception of a StopSearchNotification event, a SA stops its activity and goes into the BLOCKING state if it has previously created marker agents; otherwise, it terminates. In both cases, such SA will notify the other taskforce members of its state change (actions ac8 and ac9). Going into the BLOCKING state, the SA notifies its state change to the other taskforce members (action 6) so that each member knows the active agents in the taskforce. The notification is based on the BlockedSearcherNotification event that, once received, allows updating the list of active agents (action ac10). Moving from BLOCKING to TERMINATED the SA requests to cease its activity and also sends to all the marker agents it has previously created a termination request (action ac7).

2. If the marker agent is present and other locations are available, the SA migrates to a new location (action ac5).
3. If the marker agent is present and no other locations are available but the SA has created at least one marker agent in previously visited locations, it goes into the BLOCKING state (action ac6) to enforce the termination of such marker agents (see point 1 for the management of marker agent termination).
4. If a marker agent is present, no other locations are available and no marker agents have been previously created by the SA, the SA terminates (action ac8).

3) Simulation and performance evaluation

To evaluate the four solutions shown in Figure 4, in the *Performance Indices Definition* activity, the performance indices reported in Table 1 have been defined. The *Simulation Execution* activity relies on two simulation parameters (the number of locations and the number of *searcher agents*) and on the following settings of the network topology and information distribution:

- Locations are connected through a fully connected logical network composed of FIFO channels. In particular, channels are characterized by the same delay and bandwidth parameters modeled as uniform random variables.
- The information to be found is contained exactly at one location and the locations keep references (randomly generated) to other locations at information level to be all reachable.

Simulation runs are carried out with the number of locations equals to 100 and the number of searcher agents in the range

[2..20]. Moreover, for each simulation run, all four solutions are executed on the same network topology and information distribution. In Figures 6-10 the simulation results are reported; the obtained values of the performance indices are averaged over 50 simulation runs.

TABLE I. PERFORMANCE INDICES

Task Completion Time (T_{TC})	The duration between the spawning of the first created <i>searcher agent</i> and the first report message received by the <i>owner agent</i> .
Number of Messages (N_M)	The number of coordination messages transmitted by the agents across the network.
Notification Time (T_N)	The duration between the information finding and the notification to the last <i>searcher agent</i> .
Number of Visits (N_V)	The total number of locations visited by the <i>searcher agents</i> after the information finding.
Number of Searches (N_S)	The total number of the locations searched by the <i>searcher agents</i> after the information finding.

The T_{TC} performance index, which measures the speed with which the information search task is carried out, decreases as the number of searcher agents increases (see Figure 6). In fact, the use of more searcher agents augments the degree of parallelism which, consequently, increases the probability to find the searched information with a smaller number of migrations which are time-consuming. The performances of all the solutions are almost the same.

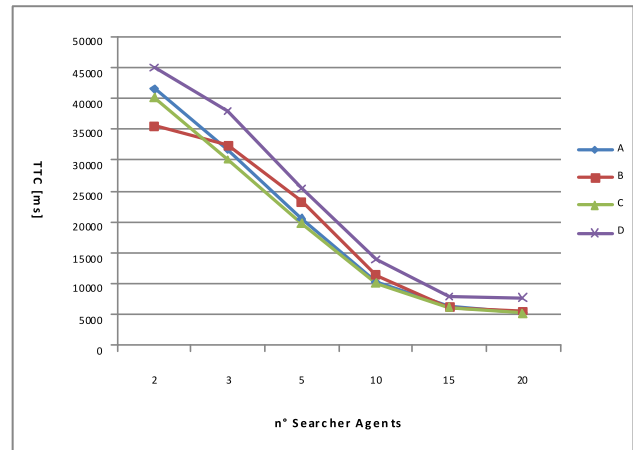


Figure 6: The Task Completion Time

The N_M parameter (see Figure 7), which measures the network load, is significantly better in the A and C solutions thus saving network resources with respect to the other solutions. In fact, as CR_1 is modeled according to the LBL interaction patterns in solutions A and C whereas the GBN interaction pattern is used in solutions B and D the number of coordination messages increases due to the GBN interaction pattern is adopted by B and D. The T_N performance index measures how fast all the searcher agents are notified after finding the information: the shorter T_N , the fewer are the resources consumed throughout the agent platform. The A and

C solutions outperform the other solutions (see Figure 8) as the network load is lighter than the ones of the B and D solutions. The N_V and N_S parameters are measures of the consumption of resources after the information is found. The values of such parameters should be kept as low as possible. As shown in Figures 9 and 10, the A and C solutions outperform the other solutions also for such indices. In particular, although the T_{TC} values of the A and C solutions

are similar to the solutions B and D, the other performance indices values are significantly better. It is worth noting that solution A not only is less straightforward than solution D but also requires the cloning of a marker agent for each visited location but such operation may not be allowed according to security policies of the locations..

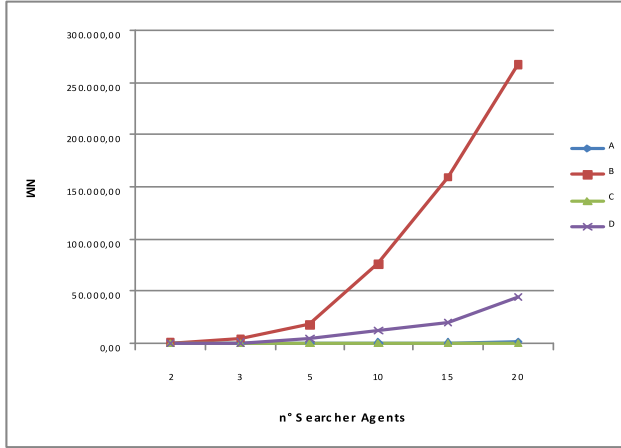


Figure 7: The Number of coordination messages

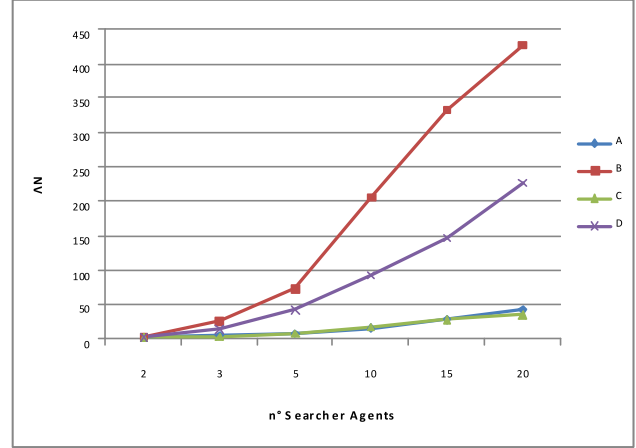


Figure 9: The Number of visits after finding information

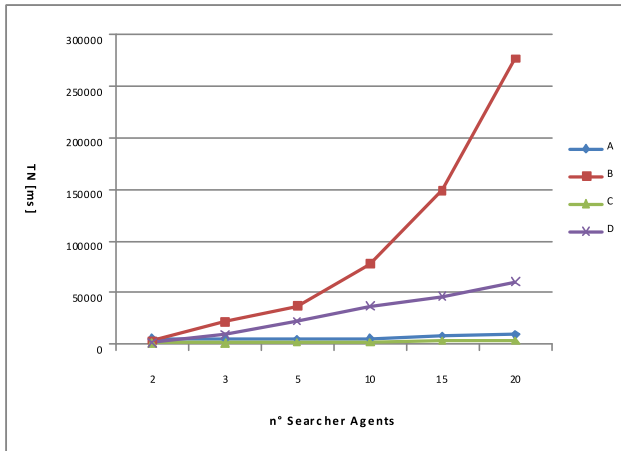


Figure 8: The Notification Time

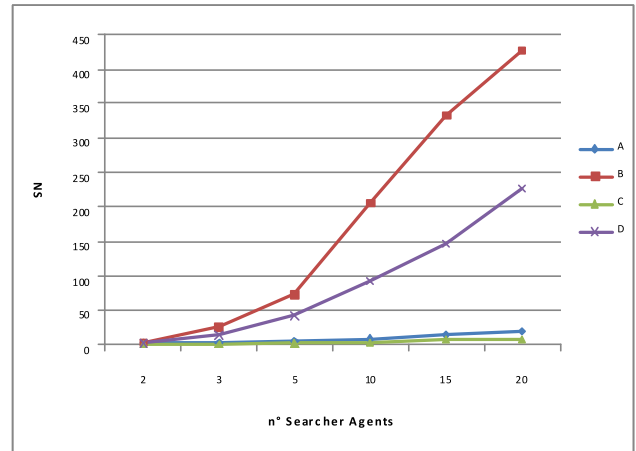


Figure 10: The Number of searches after finding information

IV. CONCLUSION

This paper has proposed ELDAMeth, a novel agent-oriented methodology supported by a CASE tool for the simulation-based prototyping of Internet-based distributed agents systems (DAS). In particular, the distinctive characteristics of ELDAMeth are: effective agent model for distributed computing systems, simulation-based agent-oriented methodology for design validation before implementation and deployment, integration with other methodologies to exploit their well-defined method fragments, and CASE tool support for supporting all development phases from modeling to simulation and implementation. Such distinctive characteristics make ELDAMeth very effective for

prototyping Internet-oriented DAS. ELDAMeth has been applied to prototype several kinds of DAS such as mobile e-Marketplaces, content delivery infrastructures, and information retrieval systems. In this paper we have shown a case study in the information retrieval domain which has demonstrated the suitability and great effectiveness of ELDAMeth for the rapid prototyping of Internet-based DAS. As future work we aim at providing full support to multi-coordination in the implementation phase: this would allow to translate multi-coordinated ELDA specifications into a real target platform represented by JADE and coordination infrastructures such as TucSon for tuple spaces and Elvin for publish/subscribe systems.

ACKNOWLEDGMENT

Authors wish to thank A. Garro, S. Mascillaro, G. Mazzitelli, and F. Rango for useful ideas, discussions and implementation efforts supporting the ELDAMeth project.

REFERENCES

- [1] Astley, M., and Agha, G. A., Customization and Composition of Distributed Objects: Middleware Abstractions for Policy Management, ACM SIGSOFT 6th International Symposium on Foundations of Software Engineering (FSE), 1998.
- [2] Bellifemine, F., Poggi, and A., Rimassa, G. 2001. Developing multi agent systems with a FIPA-compliant agent framework. *Software Practice And Experience* 31, 103-128.
- [3] Boloni, L., and Marinescu, D. C., A multi-plane state machine agent model, Fourth International Conference on Autonomous Agents, Barcelona, Spain, pp. 80-81, ACM Press, 2000.
- [4] Braun, P. and Rossak, W., *Mobile Agents: basic concepts, mobility models, & the tracy toolkit*, Heidelberg, Germany, Morgan Kaufmann Publisher, 2005.
- [5] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent- Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi- Agent Systems*. Kluwer Academic Publishers Volume 8, Issue 3, Pages 203 - 236, May 2004.
- [6] Cabri, G., Leonardi, L., and Zambonelli, F., Mobile-agent coordination models for internet applications, *IEEE Computer*, 33, 2, pp 82-89, 2000.
- [7] Cossentino, M., From Requirements to Code with the PASSI Methodology, *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini (eds). Idea Group Inc., Hershey, PA, USA, 2005.
- [8] Cossentino, M., Fortino, G., Garro, A., Mascillaro, S. and Russo, W. 2008. PASSIM: a simulation-based process for the development of multi-agent systems. *Int. J. Agent-Oriented Software Engineering* 2(2), 132-170.
- [9] ELDATool documentation and software, <http://lisdip.deis.unical.it/software/eldatool>.
- [10] FIPA Agent Management Specification, Management for agents on FIPA agent platforms, <http://www.fipa.org/specs/fipa00023/SC00023K.html>.
- [11] Fortino, G., Garro A., Mascillaro S., and Russo W. 2007. ELDATool: A Statecharts-based Tool for Prototyping Multi-Agent Systems. In *Proceedings of Workshop on Objects and Agents (WOA'07, Genova, IT, Sept. 24-25, 2007)*, pp. 14-19.
- [12] Fortino, G., Garro, A., and Russo, W. 2005. An Integrated Approach for the Development and Validation of Multi Agent Systems. *Computer Systems Science & Engineering* 20, 4, 94-107.
- [13] G. Fortino, A. Garro, S. Mascillaro, W. Russo, "Using Event-driven Lightweight DSC-based Agents for MAS Modeling," in the special issue "Best of From Agent Theory to Agent Implementation 6 (AT2AI-6)", *International Journal on Agent Oriented Software Engineering* (Inderscience publisher), 4(2), 2010.
- [14] G. Fortino, A. Garro, S. Mascillaro, W. Russo, "A Multi-Coordination based Process for the Design of Mobile Agent Interactions," In *Proceedings of IEEE Symposium on Intelligent Agents (IEEE Symposium Series on Computational Intelligence)*, Nashville (TN), USA, March 30-April 2, 2009.
- [15] G. Fortino, F. Rango, W. Russo, "Statecharts-based JADE agents and tools for engineering Multi-Agent Systems", in *Proc of 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES2010)*, Cardiff, 2010.
- [16] L. Gardelli, M. Viroli, M. Casadei, A. Omicini, "Designing Self-Organising Environments with Agents and Artifacts: A Simulation-Driven Approach", *International Journal of Agent-Oriented Software Engineering (IJAOSE)*. Volume 2(2). Page 171--195. 2008.
- [17] Himmelspach J, Röhl M & Uhrmacher AM (2008): Component based models and simulation experiments for multi-agent systems in James II. In the Proc. of the 6th Int'l Workshop "From Agent Theory to Agent Implementation" (AT2AI) jointly held with AAMAS", Estoril, Portugal, 13 May, 2008.
- [18] Loke, S. W., Padovitz, A., Zaslavsky, A., and Tasic, M., Agent Communication Using Publish-Subscribe Genre: Architecture, Mobility, Scalability and Applications, *Annals of Mathematics, Computing & Teleinformatics*, 1, 2, pp 35-50, 2004.
- [19] Luck, M., McBurney, P., and Preist, C. 2004. A manifesto for agent technology: towards next generation computing. *Autonomous Agents and Multi-Agent Systems* 9, 3, 2004, 203-252.
- [20] Martelli M., Mascardi, V. and Zini, F., Specification and Simulation of Multi-Agent Systems in CaseLP, Appia-Gulp-Prode Joint Conf. on Declarative Programming, L'Aquila, Italy. pp. 13-28, 1999.
- [21] Nwana, H.S., *Software Agents: an overview*, *Knowledge Engineering Review*, 11, 3, pp 205-244, 1996.
- [22] A. Omicini, F. Zambonelli, "Tuple Centres for the Coordination of Internet Agents," *ACM Symposium on Applied Computing (SAC'99)*, 28 February - 2 March 1999.
- [23] Pavón, J., Gómez-Sanz, J.J. and Fuentes, R. (2005) 'The INGENIAS methodology and tools', in B. Henderson-Sellers and P. Giorgini (Eds.) *Agent-Oriented Methodologies*, Idea Group Publishing, pp. 236-276.
- [24] Pavon, J., Sansores, C., and Gomez-Sanz, J. J. 2008. Modelling and simulation of social systems with INGENIAS. *Int. J. Agent-Oriented Softw. Eng.* 2, 2 (Feb. 2008), 196-221.
- [25] Sarjoughian, H.S., Zeigler, B.P. and Hall, S.B., A Layered Modeling and Simulation Architecture for Agent-based System Development, *IEEE*, 89, 2, pp. 201-213, 2001.
- [26] Sierra, C., Rodríguez-Aguilar, J. A., Noriega, P., Esteva, M. and Arcos, J.L., *Engineering Multi-agent Systems as Electronic Institutions*, *Novática*, 170, 2004.
- [27] Wooldridge, M., Jennings, N. R., and Kinny, D., The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3, 3, pp 285-312, 2000.
- [28] Zhou, X.Y., Arnason, N., and Ehikioya, S.A., A proxy-based communication protocol for mobile agents: protocols and performance, *IEEE Conference on Cybernetics and Intelligent Systems*, vol. 1, pp 53-58, 1-3, Dec. 2004.

Design and Simulation of a Wave-like Self-Organization Strategy for Resource-Flow Systems

Jan Sudeikat,
Wolfgang Renz, Thomas Preisler and Peter Salchow
Multimedia Systems Laboratory (MMLab),
Faculty of Engineering and Computer Science,
Hamburg University of Applied Sciences,
Berliner Tor 7, 20099 Hamburg, Germany
Email: {jan.sudeikat, wolfgang.renz,
thomas.preisler, peter.salchow}@haw-hamburg.de

Jan-Philipp Steghöfer, Hella Seebach
and Wolfgang Reif
Institute for Software and Systems Engineering,
Augsburg University,
Universitätsstrasse 6a, 86135 Augsburg, Germany
Email: {steghoefer, seebach,
reif}@informatik.uni-augsburg.de

Abstract—In resource-flow systems, e.g. production lines, agents are processing resources by applying capabilities to them in a given order. Such systems profit from self-organization as they become easier to manage and more robust against failures. This paper proposes a decentralized coordination process that restores a system’s functionality after a failure by propagating information about the error through the system until a fitting agent is found that is able to perform the required function. The mechanism has been designed by combining a top-down design approach for self-organizing resource-flow system and a systemic modeling approach for the design of decentralized, distributed coordination mechanisms. The systematic conception of the inter-agent process is demonstrated. Evaluations of convergence as well as performance are performed by simulations.

I. INTRODUCTION

A key driver in the development of autonomous and economic systems is the handling of complexity in large applications that consist of a great number of interacting entities. Traditional management and failure-handling approaches are no longer applicable as they do not scale well with the size of the systems and the communication required by a central management becomes prohibitive, even with modern high-speed networks. Therefore, engineers and computer scientists turn to self-organization as a means to deal with large complex systems and to keep up with the growth of such applications.

In this paper, we present a self-organizing process for the class of self-organizing resource-flow systems. This class can be applied to a great variety of domains such as production automation and logistics and systems in it can be modeled with the Organic Design Pattern (ODP) [1]. The decentralized process proposed here is analyzed and modeled with the tools provided by the SodekoVS project [2]. Changes in the configurations of agents propagate through the system like a wave until the system in its entirety has restored a stable state. During reconfiguration, parts of the system that are not affected by the process or have already been reconfigured are still able to resume their normal work. Evaluations show the quick convergence to stable states and the reconfigurations only affect system partitions.

This paper also shows how to pragmatically combine a top-down approach for the design of agent-based systems with a bottom-up approach for the design of inter-agent coordination. While the exact interplay of the two concepts is not fully elaborated here, it already becomes clear that both approaches are not necessarily orthogonal but that it is beneficial to combine both views.

This paper is structured as follows: in the following section, the ODP, as a conceptual model for self-organizing resource-flow systems, is discussed and a prominent application scenario, i.e. production automation, is introduced. In Section III, a programming model for self-organization is introduced. Subsequently, the intended coordination dynamics of self-organizing resource-flow systems are presented (see Section IV) and the realization of a decentralized role allocation strategy is discussed and evaluated (Section V). Finally, we conclude and give prospects for future work.

II. DESIGN OF SELF-ORGANIZING RESOURCE-FLOW SYSTEMS

In production automation systems, resources are transported between machines to subject these work peaces to a specific sequence of work steps. The sequence of machines is typically static. The machines that process the resources are highly specialized and only have one particular capability, i.e. an individual operation, they can apply to the resources. The transport of resources is fixed as well, e.g. by a static layout of conveyor belts. This rigid structure simplifies the management but has far-reaching implications, since reconfigurations are obstructed. The complete system has to be halted when internal errors make a single system component inoperable. Adjustments of the production process have to be carried out by stopping the system and retooling machines.

A visionary alternative are flexible, agent-based production lines that enable failure tolerance. Machines can autonomously reconfigure and transports of resources are carried out by *autonomous guided vehicles* (AGVs). Such a scenario is depicted in Fig. 1 where robots process a car body which is transported between the processing stations by autonomous carts.

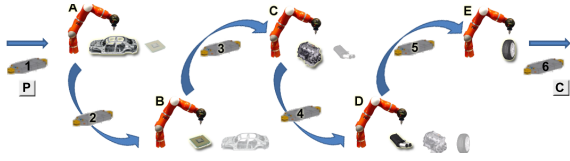


Fig. 1. Robots with different capabilities (icons to the right of the robot) process a car by applying one of their capabilities each (highlighted icon).

There are other domains where similar resource-flows occur, e.g., logistic scenarios or web service orchestration, we call this class of systems *Self-Organizing Resource-Flow Systems*. Their basic structure can be described with the ODP [3] which defines the elements that constitute the system and their relationship as shown in Fig. 2. *Task* define the required processing of *resources*. Processing steps are carried out by *agents*. The states of resources are modified by applying *capabilities*. Agents have a set of capability available and exchange resources, based on the shop layout (*inputs* and *outputs*). Which capability an agent applies and with which agents it exchanges resources is determined by a *role*¹. Roles have a precondition that describes where the resource is coming from, which state it has, and which task has to be performed on it. They also have a postcondition that describes to which agent the resource has to be given and which state and task it has after the agent has processed it. Most importantly, the role defines the *capabilitiesToApply*, i.e., what an agent is supposed to do with the resource.

To fulfill the tasks for a resource (i.e., to apply the correct capabilities in the correct order), a resource-flow is established by the allocation of roles to agents that determines how the resource is moved through the system and processed on the way. This means the combination of the roles of the agents by their pre- and postconditions respectively, is a connected chain of agents along which resources in the system are forwarded and processed. There is usually one such chain or resource-flow for each task that has to be fulfilled in a system. Each agent, however, can participate in more than one resource-flow and thus be involved in several tasks at the same time.

The interactions between the agents to handle resources are also defined on the abstract system class level and can be inherited by applications based on the ODP. They describe, amongst other things, the handover of resources and the detection of agent failures with a heartbeat mechanism. This way, both a formal analysis of the system class [3], generalized mechanisms to deal with problems in the system class such as deadlocks [4] as well as a generic runtime environment [5] become feasible.

The ODP also contains an *Observer/Controller* (O/C). This element of the system structure is the abstract extension point for the self-organization or reconfiguration mechanism. Correct system behavior is defined by invariants that have to

¹Please note that some of the terminology used in ODP has a slightly different semantics than the same terms in agent-oriented software engineering due to the historic roots of ODP.

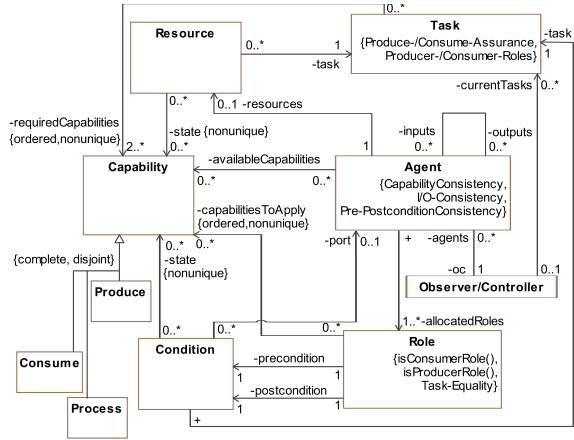


Fig. 2. The elements of the ODP for Resource-Flow Systems.

hold during the entire runtime of the system. Whenever the invariants are violated, the system has to be reconfigured to fulfill the invariants again (*Restore Invariant Approach* [3]). The individual agents are able to monitor local invariants and thus implement the observation part of the O/C. The controller part of the O/C is then responsible to calculate a new allocation of roles that restore the resource-flow and ensure that each agent has a role that fits its capabilities and its input/output relationship with other agents. How this calculation is done, however, is not specified at this point.

III. SYSTEMIC PROGRAMMING OF SELF-ORGANIZATION

In the research project "Selbstorganisation durch Dezentrale Koordination in Verteilten Systemen"² (SodekoVS) [2], a programming technique is developed that allows to equip software systems with self-organizing features. The self-organizing inter-agent process is described by discrete design elements [6]. This enforces a conceptual separation of the agent functioning and the coordination, i.e. the correlation of agent activities.

First, a modeling level for the description of inter-agent self-organization is provided. This modeling level supplements agent-oriented software engineering practices with an orthogonal description level that concerns the dynamic properties of agent-based software systems [6]. The driving force of self-organizing dynamics are distributed feedback loops among system elements [7]. These result from the mutual influences among system elements and control how fluctuations in the system context are disseminated and collectively responded to. The *systemic* modeling level addresses the description of these networks of influences and it has been found that the visualization of the mutual interdependencies of system elements is useful for the anticipation of the dynamics that software systems are able to exhibit [8], [6]. Using a graph-based modeling approach, *System Dynamics* [9] modeling concepts are specialized for describing Multi-agent systems (MAS).

²Self-Organisation by Decentralized Coordination in Distributed Systems

These models are given as an *Agent Causal Behavior Graph* (ACBG) [10]. The nodes in this graph-based modeling level represent system variables that characterize the macroscopic state of a MAS. These describe the number of agents that show a specific behavior, e.g. play a role. In addition, the current value of an interaction rate can be denoted with a specific node type. The links among these variables denote mutual influences and interdependencies. In this respect, *influences* denote additive or subtractive contributions to node values, e.g. when the activity of an agent increases or decreases the stock of a warehouse. *Interdependencies* describe causal relations where the activities of agents are mutually linked, e.g. the number of hypothetical service requesters in a system is expected to be positively linked to the number of activations of service providers. When the number of requesters increases, the number of activations increases as well and vice versa.

Secondly, a programming model that allows the enactment of ACBG-based prescriptions of self-organization processes [11], [10] facilitates application development. The key element is a distributed architecture for the enactment of decentralized inter-agent processes (cf. Fig. 3) [11]. This architecture serves as a reference model for the integration of ACBG-based processes in MAS. It provides a conceptual framework for fitting in different self-organization mechanisms and follows a layered structure. The topmost layer (*Application Layer*) contains the realization of an agent-based application. The contained agents are understood as self-contained providers of functionalities (*Application Functionality*). The contained agents individually control their activities and an underlying *Coordination Layer* enables the purposeful affecting of agents to concert the localized activities and establish collective behaviors.

The Coordination Layer describes an *event-based distributed system* [12], which allows to realize mutual influences among system elements. These influences correspond to relations in ACBG-based models of inter-agent processes, thus the layer is a means to enact the described processes in MAS. The establishment of inter-agent influences, particularly for the construction of self-organizing systems, is based on two types of mechanisms [13], i.e. techniques for the information exchange among agents (e.g. reviewed in [14]) and mechanisms for the (adaptive) adjustments of agents (among others classified in [15]), due to the perceived information (see Section VI). The Coordination Layer contains two types of functional elements for the encapsulation of these aspects. *Coordination Media* are conceptual containers of so-related interaction infrastructures. Specific interaction modes, e.g. the mediation by an environment [16] or Linda-like tuple spaces, are encapsulated and reused by a generic interface [11]. *Coordination Endpoints* interact on behalf of agents via these media and are able to influence the agent execution. These elements are used to encapsulate and automate the coordination-related activities. These activities concern the interactions via Media, i.e. the invitation and participation of interactions, as well as the affectation of modifications in the agent models.

The ACBG-based modeling of dynamics of inter-agent

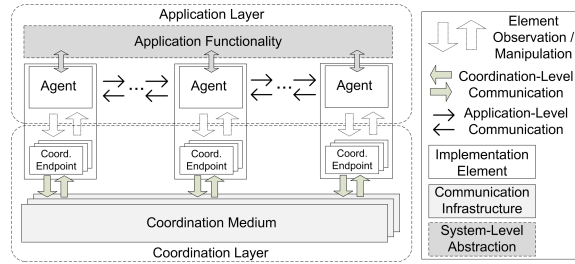


Fig. 3. The SodekoVS-Architecture for the embedding of decentralized coordination in MAS [17].

coordination is exemplified in the Sections IV and V. A configuration language [10] allows to map ACBGs to agent-based software systems. These mappings describe the realization of influences among agents, i.e. the coordination-related logic that controls the initiation, participation, and reaction to interactions as well as the media that mediate interactions. The detailing of these models, as a systematic programming effort, is not discussed in this article but details on the configuration of process enactments can be found in [17].

IV. SYSTEMIC MODEL OF ADAPTATION DYNAMICS

In [18], the systematic integration of decentralized coordination strategies in MAS has been discussed. The conception of the appropriate coordination is approached by modeling the problematic, unintended behavior of applications. Based on the identification of the *Problematic Dynamic*, a corresponding *Solution Dynamic* is derived that supplements the application behavior with additional interdependencies and inter-element feedbacks to correct the system behavior and alleviate unintended effects.

The Problem Dynamic of an ODP-based resource-flow systems is illustrated in Fig. 4 (right). Initially, agents are *running* and one or several roles are allocated to them which are executed in order to process resources. Random errors make it impossible for the agent to apply one or more of its roles. The adoption of roles that can not be applied is controlled by a fluctuating rate (*RF interrupt*) that is positively influenced by the availability of running, thus breakable, agents and the changing number of error events (*Error*). This rate describes the *resource-flows* (RF) that are interrupted, due to the breaking of agents. These failures within individual agents limit the number of running agents (negative link), thus the problematic system behavior is dominated by a negative feedback loop (α).

If not handled, this dynamic causes the number of agents that are not running to increase over time. The design of an appropriate Solution Dynamic concerns the derivation of agent behaviors that counteract this unintended effect. A very general structure is given on the left hand side of Fig. 4. Agents that have roles they can no longer apply are *Waiting for Reconfiguration*. The rate of interrupts positively influences the increase of this variable. The system is equipped with a reconfiguration mechanism, and for each of the waiting

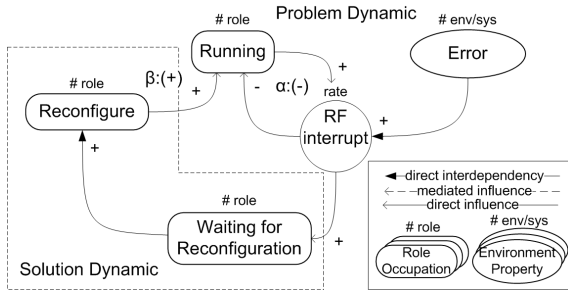


Fig. 4. The Problem and Solution Dynamic of the ODP.

agents a new configuration is determined. Thus the system shows a causal relation. In absence of waiting agents, no reconfigurations take place. Occurrences of waiting agents enforce subsequent reconfigurations (*Reconfigure*) to restore a set of executable roles. The reconfigurations thus increase the number of *Running* agents by complementing a counteracting feedback loop (β).

This Solution Dynamic deliberately omits the concrete mechanism with which new role allocations are determined. Also the locally applied techniques to the enactment of reconfigurations are abstracted. A method to express the problem of finding a fitting role allocation as a constraint-solving problem has been presented in [19] and solved with a centralized approach. Whenever an agent can no longer apply one of its roles or whenever an agent breaks, the resource-flow is interrupted. When the interruption is detected, the system reconfigures in order to restore the flow. During the course of the reconfiguration process, a new allocation of roles to agents is calculated and the roles are communicated to the agents which then apply them again. The next section describes an alternative reconfiguration mechanisms in which new role allocations are found in a decentralized fashion by propagating the demand for local reconfigurations through the system.

V. WAVE-LIKE DECENTRALIZED RECONFIGURATION

A completely decentralized reconfiguration approach is based on the idea that a wave of role re-allocation runs through the system in order to re-establish the resource-flow. Assuming that each agent is capable to exhibit a set of capabilities (see Section II), a correct resource flow can be (re-)established by the appropriate swapping of roles. Failing agents adopt actable roles and in return other agents help out by providing the unactable roles. The failing agent emits a wave of reallocations by sending requests for assistance along the resource flow. Each recipient has to decide locally if it is capable and will swap roles. Generally, a single swap of roles is not enough to reestablish the full sequence of activities and transitive changes of roles are required.

The operating principle is exemplified in Fig. 5 for a simple scenario that requires a transitive swap of roles. Three Robots are connected in series (1). For each agent the set of actable capabilities are listed and the topmost capability is used in the currently active role, e.g. *Robot 1* is currently configured

to play a role that involves *Cap. 1* but may also apply *Cap. 3*. Two types of Cart agents represent the initial provision (*Producer*) and the final collection of the processed workpieces (*Consumer*). Due to an error *Robot 1* can no longer apply *Cap. 1* and sends a request for assistance (2). This request is routed along the resource flow till it reaches an agent that is capable to execute the needed capability, here *Robot 2* which replies to the request (3). The reply is routed to the requesting Robot as well as the Carts that are connected to the swapping robots. Consequently, the Robots and Carts reconfigure their local roles. The robots update their roles and the resource flow is reestablished by adjusting the ports in the pre- and post-conditions of the roles of the connected Carts to ensure that workpieces reach the robots in the intended sequence. In the best case, the originating and the receiving agents can just switch their roles, thus restoring the resource-flow.

In Fig. 5, *Robot 2* is able to provide capability *Cap. 1* but *Robot 1* is not able to replace *Robot 2* as it is missing the currently utilized *Cap. 2* (3). Thus after the swap, *Robot 1* remains in a problematic state and requests assistance (4). This request is propagated again till it reaches a robot with the required capability (*Robot 3*) and the swap proceeds as above (5). Since *Robot 1* is able to replace the currently active capability of *Robot 3*, i.e. *Cap. 3*, the correct sequence of capabilities is finally reestablished (6). Here, the reconfiguration logic has been described for agents that only play on role at a time and the subsequent simulations concerns this simplified scenario. In principle, agents can be part of several resource-flows and in that case, the agents only reconfigure for roles that include the broken capability and keep processing resources of other tasks. Consequently, the informed Carts change only the ports of the affected roles accordingly.

A detailed ACBG of the outlined reconfiguration algorithm is illustrated in Fig. 6. This description refines the previously given Solution Dynamic (cf. Fig. 4) as it illustrates how the decentralized strategy relates to the dynamics of ODP. In addition, it indicates the system-level effects of the decentralized reconfiguration that are examined in Section V-A. When agents are *Waiting for Reconfiguration* due to error events, they show two behaviors. First, they are *Deficient* as one or more roles, which are required for the processing of resources, are inoperable. These roles are distinguished by the reason for deficiency. Agents can be rendered deficient by error events (*By Break*) or they deliberately decided to abandon a role in order to adopt another role on behalf of another agent (*By Change*). In the former case, the agents are rendered inefficient and in the latter case agents assist other agents. Secondly, these agents are considered to be *Non-Active*, i.e. they have the capacity to play another role. Agents can concurrently play several roles, therefore, the non-activity only denotes that the agent is underutilized, i.e. is capable to exhibit another role. This means that in agents that can play several roles, the *Running* and *Non-Active* roles do not exclude each other, but an agent can be associated to an active role (*Running*) and still have the capacities to play additional roles (*Non-Active*).

Agents are equipped with the ability to autonomously

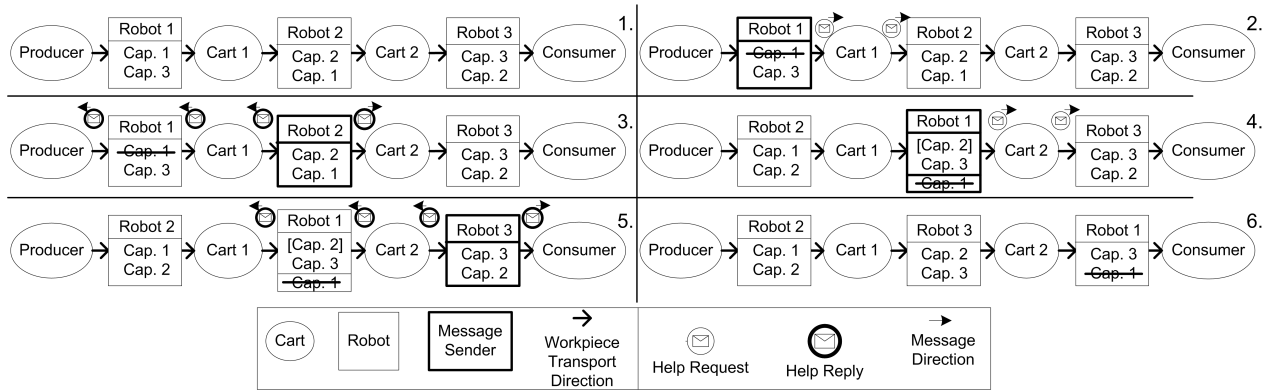


Fig. 5. Exemplification of the decentralized reconfiguration.

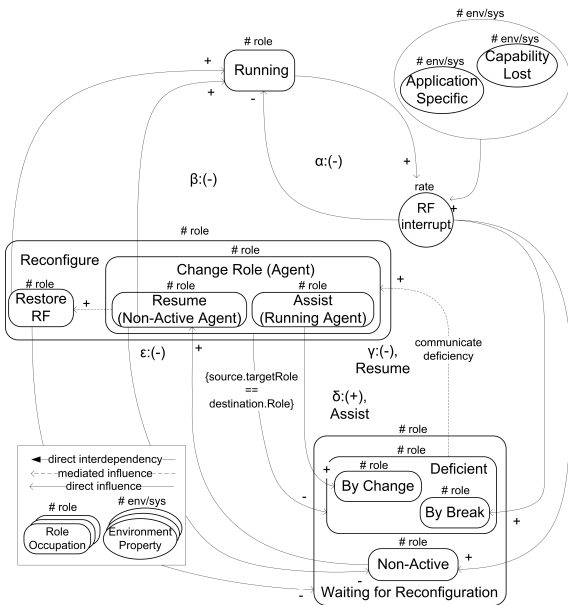


Fig. 6. ACBG for the Solution Dynamic of the wave-like, decentralized reconfiguration algorithm

change their allocation (*Change Role*). Deficient robots indicate their shortcoming to other agents (*communicate deficiency*) via a Coordination Medium (cf. Section III). The medium controls the sequential reception of the request along the flow of resources in the production line. Recipients decide locally whether to change their role-allocation or not, based on their individual abilities. The changing behavior is distinguished by the receiving agent that adjusts the local configuration. Non-Active agents *Resume* the executions since these become operational. Running agents that adjust their role *Assist* the requesting agent. These roles have different effects on the agent population. All changes remove deficiencies and the annotation $source.targetRole == destination.Role$ indicates that only those deficiencies are removed (destination) that

changing agents (source) commit to. Another commonality is that the adjustment of a role entails the restoring of the flow of resources among the agents (*Restore RF*). When an agent has adjusted its role, those agents that received resources from it or gave resources to it need to adapt as well. This activity is separated from the role change as the agents do not deliberately decide about these changes. These are reconfigurations within connected agents that are enforced as they are consequences of the deliberate changes. These reconfigurations may as well be transmitted via Coordination Media (see Section III).

Assisting another agent introduces new deficiencies, as the assisting agent is giving up one role that needs to be played by another agent (*Deficient by Change*). Thus the net amount of Non-Active agents is unaffected. However, these changes may be necessary in settings where agents can not swap roles directly and transitive changes of roles are required (as exemplified in Fig. 5). When Non-Active agents *Resume* to adopt a role, the number of Non-Active agents is reduced. Still, this requires the availability of Non-Active agents.

The changing activities of agents control the overall negative feedback (β) that increases the number of operational agents and reduces the number of deficient agents. Three auxiliary feedbacks influence the exhibited system dynamics (δ, γ, ϵ). First, agents that assist others create a reinforcing feedback loop (δ), which originates from the fact that an *assisting* agent adds additional deficiencies to the system. If an agent *resumes* its activities, deficiencies and non-active agents are removed instead, thus instituting a balancing feedback (γ). The ability to resume is limited by the amount of Non-Active agents, leading to a third balancing feedback (ϵ).

A. Estimating the Effects of Adaptation Dynamics

The outlined Solution Dynamic (cf. Fig. 4) denotes an inter-agent process that can be implemented with the systemic programming model (see Section III). Prior to the detailed design and embedding of this process, the effected system behavior is anticipated. One approach to estimate the dynamics of self-organizing MAS is their simulation in stochastic process

algebra models [20]. It is important to note that the resulting stochastic models abstract from the agent implementations and their internal reasoning. Instead, stochastic terms are used to describe the dynamics with which specific behaviors are adopted or left. The number of currently active process terms resembles the number of agents that show specific behavior, i.e. the variable-values of an ACBG-based process description.

Fig. 7 illustrates the simulation model used to anticipate the Solution Dynamic. The model is given in stochastic π -calculus [21] and simulated with the *Stochastic Pi Machine*³ (SPIM). Details on the simulation language and graphical notation can be found in [22]. Agents are modeled as processes that communicate/interact via channels. The stochastics of interactions are given by annotating processes with delays (τ) and assigning interaction rates to channels [21], [22]. The notation \bar{x} denotes the sending of data on the channel x and \underline{x} denotes the reception of data via the channel x .

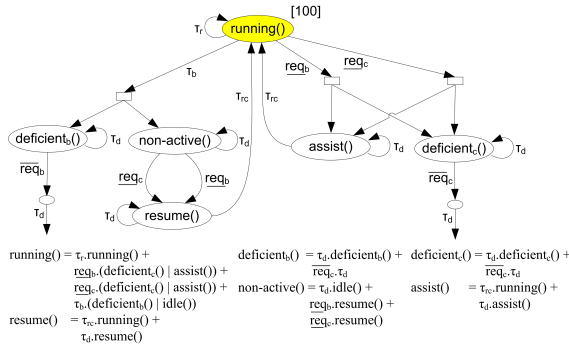


Fig. 7. Simulation Model of the Solution Dynamic in Stochastic π -Calculus.

In the simulation model, the number of agent behaviors that are exhibited are expressed by the number of active processes. Processes communicate via two channels. The channel req_b is used to communicate requests for a role-switch, due to an internal error, i.e. the internal breaking of the requester. When roles are requested to be switched in order to assist an agent, these requests are sent via the channel req_c . Operative agents are denoted by the *running* process. Internal errors occur with a fixed rate, as defined by the delay τ_b . Inoperative robots are represented the concurrent execution of the *deficient_b* and *non-active* processes. Deficient processes end when a request for re-assignment of the affected role is processed by a recipient. The *non-active* processes become *resume* processes when they receive any request for re-allocation. The time delay τ_{rc} resembles the time needed to reconfigure. Afterwards, the agent is in operation, i.e. exhibits the *running* process. When *running* agents receive a request to switch roles, they convert to the concurrent execution of the *assist* and *deficient_c* processes. The assistance transforms back to the running process, delayed by the time to reconfigure the agent (τ_{rc}). The *deficient_c* processes describe the search for another agent that is able to play the role that an assisting agent possessed before

³<http://research.microsoft.com/en-us/projects/spim/>

the assisting adjustment. Deficiency ends when another agent processes the request for a role change, communicated via the channel req_c . This simulation model abstracts from the agents that participate in the system. The time needed to *restore the resource flow* (cf. Fig. 6), i.e. the adjustments of the roles of the directly connected agents to reestablish the correct flow of resources among machines, is part of the time delay τ_{rc} . We assume that the rerouting of resource transportations is always possible.

Simulations indicate that, at a high enough level of redundancy, the system reliably recovers due to the decentralized switching of agent roles. This process describes a structure formation as the system maintains the operational system configuration. The fraction of recovering situations is predicted by this simulation to depend on the redundancy level in a similar way as is shown in the results section below.

B. Agent-based Realization

After the anticipation of the affected system behavior, this reconfiguration strategy has been integrated in a MAS by using systemic programming model (see Section III). The system implementation (*Application Layer*, see Figure 3) makes use of the freely available *Jadex*⁴ agent framework. The Robots and Carts within production lines are represented by *Jadex*-agents and the exchanged workpieces are mimicked by objects that are exchanged via *FIPA Agent Communication Language* (ACL) messages. A realization of the *Coordination Layer* (see Fig. 3) for this agent platform is utilized [11].

For this application scenario a tailored Medium realization is utilized that routes request and reply messages along the resource flow. Conceptually though, agents are aligned in a circle thus all agents can be reached independent of the location of the incapacitated agent. Endpoints encapsulate the logic to coordinate the reconfiguration process and interact via the Medium. Endpoints observe the agent-operation and initiate the reconfiguration process by sending a help request if an agent becomes deficient. The help request is forwarded through the medium. Each endpoint along the message path decides whether to adopt the deficient agent's role or continues forwarding the help request. If the endpoint decides to adopt the role, a reply is sent. The reply is sent in both directions through the medium to inform all agents which are affected (robots and connected carts) by the reconfiguration process. Again, each endpoint receiving the reply decides to change the agent configuration. The reply is sent backward through the medium until all affected agents are informed. If an endpoint receives multiple coordination messages these messages are queued and processed in the order of their arrival.

C. Implementation Test Results

The example system illustrated in Fig. 5 has been implemented and tested for measuring the handling of breaking capabilities. When the robot is rendered incapacitated by such an event, the associated endpoint notices this and initiates

⁴<http://jadex-agents.informatik.uni-hamburg.de/>

an interaction via the Coordination Medium that triggers the swapping of roles. The first swap involves the incapacitated agent that is *Deficient by Break*. In this system configuration, a second swap is required that resolves a transient *Deficient by Change* agent behavior.

In addition, we examine the relation of the redundancy within agents with the effectiveness of the reconfigurations. The effectiveness of the reconfiguration procedure is influenced by the number of alternative capabilities that are available to the individual agents. This level of redundancy is measured by the ratio of the number of *individual* capabilities (C_i), to the *absolute* number of capabilities (C) that are required for the processing of a workpiece ($\frac{C_i}{C}$). In the following, we assume a homogeneous setting, where the robots are equipped with an equal number of redundant capabilities. The composition of these capabilities is normally distributed. In Fig. 8, measuring results for a simple scenario with 10 different capabilities and agents are shown. Each single run processes of a fixed number of workpieces while, at two fixed instances of time, a randomly selected agent is incapacitated. A first estimate of the effectiveness of the reconfigurations is the number of exchanged messages (see Fig. 8). The number of messages increases quickly when the number of redundancies decreases. This measurement can be analytically fitted with $(c_1 * (1-x))^2 + c_2$.⁵ A complementary measurement is the number of hops that requests for assistance travel before a swapping agents is found. This describes the logical distance between the swapping agents. The results for this measurement have shown the same characteristics like the message count and can be fitted with the same function (but different constants). Thus the decentralized reconfiguration strategy is particularly suited for production lines where the capability types are often available.

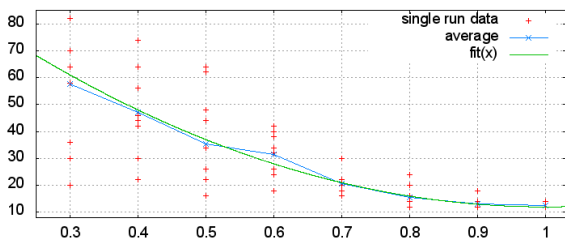


Fig. 8. Measurement results. The averaged number of messages are plotted over the redundancy of capabilities.

VI. RELATED WORK

Ant Colony Optimization has been used for decentralised control in production systems. [23] uses the mechanism to control autonomous vehicles, similar to our carts. The distribution of jobs to production machines has not been a concern there. The more complex problem of scheduling jobs to run on certain machines has, e.g., been tackled in [24] and [25].

⁵ c_1, c_2 are application dependent constants. In this case, these are 10 and 12 respectively.

However, these papers from operations research focus on optimization of a shop floor and do not take into account robustness and reconfiguration that happens at run-time. Also, only partial problems are investigated, either focusing on the routing of carts or the scheduling of production machines.

The work presented here concerns the run-time reconfiguration by self-organization. The maintained structure is a correct sequence of agents that are perturbed by individual failures that incapacitate agents. Prominent alternatives to the process presented here are centralized/distributed constraint solving techniques and market-based mechanisms. The correct configuration is described with constraints and an approach to use centralized constraint solving to restore functionality after a failure has already been published [19]. Consequently, distributed constraint solving techniques [26] can be used as well, e.g. as studied in [27]. An example for market-based mechanisms is given in [28]. A manufacturing line is provided with a flexible transport mechanism and work pieces and machine agents negotiate for the execution of working steps. In a way, the algorithm proposed in this paper is an *optimistic* and *minimalistic* version of a distributed constraint solver. By exchanging roles, the agents collectively restore the invariants of the system. The strategy is minimalistic since the number of required messages and the amount of shared information is reduced. It is optimistic because the assisting role changed are carried out before the complete solution is calculated. Its main advantage over traditional distributed constraint solvers is the minimal amount of calculation that is involved at the agents. They can therefore be very small with only minimal CPU power and RAM, making them cheap and easily replaceable.

Here, these design alternatives are qualitatively characterized by a subset of criteria from [29]. Their quantitative comparison is left for future work. A first aspect is the necessary *communication*. The presented process minimizes the message content and only the information that immediately necessary to resolve a local failure is communicated. The number of messages ranges from a single role swap (best case) to the successive swapping of roles by all agents (worst case). The dependence of this measure on structural properties is shown in Figure 8. The communications are only carried out when failures are present. Alternatives involve that coordination-related messages have to be exchanged during the normal system operation, e.g. as workpieces constantly negotiate their further processing [28]. Also the amount of *computations* and the considered/exchanged *knowledge* about the system state is minimized. The participation in the process involves only the local consideration whether an agent is capable to play a required role. Now further information about the global system state is processed. Centralized constraint solvers require the full knowledge about the system and decentralized solvers require the information from neighboring agents. In addition, the *adaptations* are carried out concurrently to the system operation. Unaffected partitions of the production line continue to work. Finally, the accuracy of the quality of the found solution, i.e. stable configuration, varies. The process follows the heuristic that role swaps of nearby agents are favored over

the swaps of (logically) distant agents. The explicit treatment of the underlying constraint problem allows in principle to prepare the optimization of the found solutions.

VII. CONCLUSIONS

In this paper, we have described a decentralized reconfiguration process to restore valid system configurations in self-organizing resource-flow systems. The reconfiguration algorithm works by exchanging roles with neighboring agents and by propagating change requests in a wave-like manner until all of them could be satisfied. The mechanism has been developed by combining a top-down process for the description of resource-flow systems and a bottom-up process for the design of agent coordination. Its performance has been demonstrated with a number of simulations.

The most interesting feature of the decentralized process proposed here is that reconfigurations are organized locally in the production line, i.e. the rest of the system is not impaired by a failure. Thus, parts of the system that are not involved into a local reconfiguration can continue to run normally. The way the reconfiguration propagates also ensures that only a small amount of agents is in a state of non-processing resources at an instance of time. This feature will be prominent also when using the wave-like algorithm in non-linear production situations, which is a straight-forward generalization instance. Local heuristics taking into account exchange-success rates could be predefined or evolved using learning algorithms. Future work includes a more detailed study on the combination of bottom-up design of coordination methods as proposed in SodekoVS and of top-down design methodologies as promoted with the ODP. This will also include a comparison of their respective advantages and problems that occur when both worlds are combined.

ACKNOWLEDGMENT

This research is partly sponsored by the German research foundation (DFG) in the project SodekoVS and in the DFG special priority program “Organic Computing” (SPP 1183) in the project SAVE ORCA.

REFERENCES

- [1] H. Seebach, F. Ortmeier, and W. Reif, “Design and Construction of Organic Computing Systems,” *IEEE Congress on Evolutionary Computation, 2007*, pp. 4215–4221, Sept. 2007.
- [2] J. Sudeikat, L. Braubach, A. Pokahr, W. Renz, and W. Lamersdorf, “Systematically engineering selforganizing systems: The sodekovs approach,” *Electronic Communications of the EASST*, vol. 17, 2009.
- [3] M. Gudemann, F. Nafz, F. Ortmeier, H. Seebach, and W. Reif, “A specification and construction paradigm for Organic Computing systems,” in *Proceedings of the Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2008, pp. 233–242.
- [4] J.-P. Steghöfer, P. Mandrekar, F. Nafz, H. Seebach, and W. Reif, “On Deadlocks and Fairness in Self-organizing Resource-Flow Systems,” in *Proceedings of the 30th International Conference on Architecture of Computing Systems (ARCS), LNCS 5974*. Springer, 2010, pp. 97–100.
- [5] F. Nafz, F. Ortmeier, H. Seebach, J.-P. Steghöfer, and W. Reif, “A generic software framework for role-based Organic Computing systems,” in *SEAMS 2009: ICSE 2009 Workshop Software Engineering for Adaptive and Self-Managing Systems*, 2009.
- [6] J. Sudeikat and W. Renz, “Qualitative modeling of mas dynamics - using systemic modeling to examine the intended and unintended consequences of agent coaction,” in *Agent-Oriented Software Engineering X*. Springer, 2009, to be published.
- [7] Y. Brun, G. di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, *Software Engineering for Self-Adaptive Systems*. Springer-Verlag, 2009, ch. Engineering Self-Adaptive Systems through Feedback Loops, pp. 48–70.
- [8] W. Renz and J. Sudeikat, “Modeling feedback within mas: A systemic approach to organizational dynamics,” in *Organised Adaptation in Multi-Agent Systems*, 2008, pp. 72–89.
- [9] J. D. Sterman, *Business Dynamics – Systems Thinking and Modeling for a Complex World*. McGraw-Hill, 2000.
- [10] J. Sudeikat and W. Renz, “MASDynamics: Toward systemic modeling of decentralized agent coordination,” in *Kommunikation in Verteilten Systemen*, ser. Informatik aktuell, 2009, pp. 79–90.
- [11] —, “Decomas: An architecture for supplementing mas with systemic models of decentralized agent coordination,” in *Proc. of the 2009 IEEE/WIC/ACM Int. Conf. on Intel. Agent Tech.*, 2009, pp. 104–107.
- [12] G. Mühl, L. Fiege, and P. Pietzuch, *Distributed Event-Based Systems*. Springer-Verlag New York, Inc., 2006.
- [13] J. Sudeikat and W. Renz, *Applications of Complex Adaptive Systems*. IGI Global, 2008, ch. Building Complex Adaptive Systems: On Engineering Self-Organizing Multi-Agent Systems, pp. 229–256.
- [14] T. DeWolf and T. Holvoet, “Decentralised coordination mechanisms as design patterns for self-organising emergent systems,” in *Engineering Self-Organising Systems*, vol. 4335/2007, 2007, pp. 28–49.
- [15] G. D. M. Serugendo, M. P. Gleizes, and A. Karageorgos, “Self-organisation and emergence in mas: An overview,” in *Informatica*, vol. 30, 2006, pp. 45–54.
- [16] H. V. D. Parunak and S. Brueckner, “Engineering swarming systems,” in *Methodologies and Software Engineering for Agent Systems*. Kluwer, 2004, pp. 341–376.
- [17] J. Sudeikat and W. Renz, “Programming adaptivity by complementing agent function with agent coordination: A systemic programming model and development methodology integration,” *Communications of SIWN*, vol. 7, pp. 91–102, may 2009, iSSN 1577-4439.
- [18] —, “On the modeling, refinement and integration of decentralized agent coordination – a case study on dissemination processes in networks,” in *Self-Organizing Architectures*, 2010, pp. 251–274.
- [19] F. Nafz, F. Ortmeier, H. Seebach, J.-P. Steghöfer, and W. Reif, “A universal self-organization mechanism for role-based Organic Computing systems,” in *Proceedings of the Sixth International Conference on Autonomic and Trusted Computing (ATC-09)*, 2009.
- [20] M. Casadei, L. Gardelli, and M. Viroli, “Simulating Emergent Properties of Coordination in Maude: the Collective Sorting Case,” in *5th Int. Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA)*, 2006.
- [21] C. Priami, “Stochastic π -calculus,” *Computer Journal*, vol. 6, pp. 578–589, 1995.
- [22] A. Phillips, *Symbolic Systems Biology: Theory and Methods*. Jones and Bartlett Publishers, 2010, ch. A Visual Process Calculus for Biology.
- [23] V. A. Cicirello and S. F. Smith, “Ant colony control for autonomous decentralized shop floor routing,” in *International Symposium on Autonomous Decentralized Systems*, 2001.
- [24] N. Liouane, I. Saad, S. Hammadi, and P. Borne, “Ant systems & local search optimization for flexible job shop scheduling production,” *Int. Journal of Comp., Comm. & Control*, vol. 2, no. 2, pp. 174–184, 2007.
- [25] C. Gagné, M. Gravel, and W. L. Price, “Solving real car sequencing problems with ant colony optimization,” *European Journal of Operational Research*, vol. 174, no. 3, pp. 1427 – 1448, 2006.
- [26] M. Yokoo, E. Durfee, T. Ishida, and K. Kuwabara, “The distributed constraint satisfaction problem: Formalization and algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 5, pp. 673–685, 1998.
- [27] G. Clair, E. Kaddoum, M.-P. Gleizes, and G. Picard, “Self-regulation in self-organising multi-agent systems for adaptive and intelligent manufacturing control,” in *SASO '08: Proc. of the 2008 Sec. IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems*, 2008, pp. 107–116.
- [28] N. R. Jennings and S. Bussmann, “Agent-based control systems,” *IEEE Control Systems*, vol. 23, no. 3, pp. 61–74, 2003.
- [29] E. Kaddoum, M.-P. Gleizes, J.-P. George, and G. Picard, “Characterizing and evaluating problem solving self-* systems,” in *COMPUTATION-WORLD '09: Proc. of the 2009 Computation World*, 2009, pp. 137–145.

Generating Inspiration for Multi-Agent Simulation Design by Q-Learning

Robert Junges

Modeling and Simulation Research Center
Örebro University, Sweden
Email: robert.junges@oru.se

Franziska Klügl

Modeling and Simulation Research Center
Örebro University, Sweden
Email: franziska.klugl@oru.se

Abstract—One major challenge in developing multi-agent simulations is to find the appropriate agent design that is able generating the intended overall phenomenon respectively dynamics, but does not contain unnecessary details. In this paper we suggest to use agent learning for supporting the development of an agent model: The modeler defines the environmental model and the agent interfaces. Using rewards capturing the intended agent behavior, Reinforcement Learning techniques can be used for learning the rules that are optimally governing the agent behavior. However, for really being useful in a modeling and simulation context, a human modeler must be able to review and understand the outcome of the learning. We propose to use additional forms of learning as post-processing step for supporting the analysis of the learnt model. We test our ideas using a simple evacuation simulation scenario.

I. MOTIVATION

Methodological questions are more and more in the focus of research on agent-based simulation as the number of challenges in developing a good multi-agent simulation model are numerous. The central issue hereby concerns what behaviors the agents should exhibit so that the intended outcome is generated. What particular detail must be included, what part of the modeled behavior is not necessary? How to set the parameters involved? However, if it is not fully clear from the beginning how this local behavior should be - even if the original agents behavior can be easily observed - the development may result in a painful try and error procedure. The modeler may add, respectively remove behavioral elements, try different parameter values and test the overall outcome again and again. Such a procedure might be feasible for an experienced modeler who knows the critical starting points for modifications and is capable of using complex calibration tools for multi-agent simulation such as described in [1], but this cannot be assumed for less experienced modelers.

In this contribution we are suggesting to solve this search for the appropriate agent-level behavior by using agent learning. The vision is hereby the following procedure: the modeler starts by developing an environmental model as a part of the overall model, then, determines what the agent might be able to perceive and to manipulate and finally describes the intended outcome based on a reward function that evaluates the agents performance. The agents then use a learning mechanism for determining a behavior program that together generates the intended overall outcome in the given environment. This strat-

egy might be also described as a variant of an environment-driven strategy for developing multiagent simulations [2].

A major issue in this overall procedure refers to the selection of the particular learning agent architecture. An initial analysis of different learning techniques applicable for this problem has already been described in [3]. There, Learning Classifier Systems (LCS), Feed Forward Neural Networks (FFNN) and Reinforcement Learning (Q-Learning) have been evaluated with regards to learning performance and resulting behavior representation, using the same evacuation scenario problem as in the following. In this contribution we are further investigating Reinforcement Learning for its suitability in such a learning-driven model development process, focusing more on the interpretability of the state-action mapping produced. We are not focussing on mere optimization performance, but on softer factors that define the usability of Q-Learning in the model development setting: the completeness, the complexity and the generalization capabilities of the behavior learnt.

In the next section we will review existing approaches for learning agent architectures in simulation models. This is followed by a more detailed treatment of the learning-driven methodology and a presentation of the reinforcement learning architecture. In section IV and V we describe the used testbed, the experiments conducted with it and discuss the results. The paper ends with a conclusion and an outlook to future work.

II. LEARNING AGENTS AND SIMULATION

Adaptive agents and multi-agent learning have been one of the major focuses within distributed artificial intelligence since its very beginning [4]. Many different forms of learning have shown to be successful when working with agents and multiagent systems. Obviously, we can not cover all techniques for agent learning in this paper, the following paragraph shall give a few general pointers and then give a short glance on directly related work on agent learning in simulation settings. In general our contribution is special concerning the objective of our comparison: not mere learning performance but its suitability for the usage in a modeling support context.

Reinforcement learning [5], learning automata [6], evolutionary and neural forms of learning are recurrent examples of learning techniques applied in multi-agent scenarios. Besides that, techniques inspired by biological evolution have been applied for agents in the area of Artificial Life [7], [8], where

evolutionary elements can be found together with multiagent approaches. An example of a simulation of a concrete scenario is [9], in which simulated ant agents were controlled by a neural network that was designed by a genetic algorithm. Another experiment, with an approach similar to a Learning Classifier System (LCS) can be found in [10], where a rules set was used and modified by a genetic algorithm.

Although there is a wealth of publications dealing with the performance of particular learning techniques, especially reinforcement learning approaches, there are not many works focussing on the resulting behavioral model dealing with usability. An early example can be found in [11], where an evolutionary algorithm is applied to behavior learning of an individual agent in multi agent robots. Another example, from [12], describes a general approach for automatically programming a behavior-based robot. Using Q-Learning algorithm, new behaviors are learned by trial and error based on a performance feedback function as reinforcement. In [13], also using reinforcement learning, agents share their experiences and most frequently simulated behaviors are adopted as a group behavior strategy. [14] compares reinforcement learning and neural networks as learning techniques in an exploration scenario for mobile robots. The authors conclude that both learning techniques are able to learn the individual behaviors, sometimes outperforming a hand coded program, and behavior-based architectures speed up reinforcement learning.

III. AGENT LEARNING ARCHITECTURES FOR MODEL DESIGN

The basic idea behind a learning-driven design methodology consists in the transfer of the agent behavior design and test activity from the human modeler to the simulation system. Specially in complex models, a high number of details can be manipulated. This could make a manual modeling, debugging and tuning process cumbersome, especially when knowledge about the original system or experience for implicitly bridging the micro-macro gap is missing. Using agents that learn at least parts or initial versions of their behavior might be a good idea for supporting the modeler in finding an appropriate low level behavior model. Such a learning-based approach can also be part of something as the adoption of a Living Design [15] like methodology for multi-agent simulation models. Nevertheless, the first question on a way to such a learning-driven methodology, is about the selection of the appropriate learning technique – for this form of application, for a particular domain, or maybe just for a particular model. In this paper we focus on the suitability of a well know learning technique, Q-Learning, for such a modeling approach. Before we continue with focussing on this particular learning architecture, we discuss what we have identified as requirements for the applicability of an learning technique to our problem.

A. Requirements for Learning Agent Architectures

Not all agent learning architectures are equally apt for usage in the modeling support context. There are a number

of properties that an appropriate learning technique may be able to exhibit for indicating a successful application.

- 1) *Feasibility*: The learning mechanism should be able to cope with the level of complexity that is required for a valid environmental models. Thus, it should not be necessary to simplify or even to reformulate the problem just for being able to apply the learning mechanism; That means the theoretical prerequisites for applying the learning technology must be known and fulfilled by the environmental model in combination with the reward function. The learning architecture must be able to find a good-enough solution;
- 2) *Interpretability and Model Accessibility*: The mechanism should produce behavior models that can be understood and interpreted by a human modeler. The architecture shall not be a black box with a behavior that the human modeler has to trust, but must be accessible for detailed analysis of the processes involved in the overall agent system;
- 3) *Plausibility*: The mechanism in the learning architecture should be well-established and well-understood. The motivation is that its usage shall not impose additional complexity to the modeler for example in setting a number of configuration parameter. How the learning architecture works, shall be explainable to and by the modeler.

There is a variety of possible learning agent architectures that might be suitable for the aim presented here and the requirements identified – as discussed in section II. We selected Q-Learning, as a Reinforcement Learning technique, as we describe it in the next paragraph.

1) *Q-Learning*: Q-Learning [16] is a well-known reinforcement learning technique. It works by developing an action-value function that gives the expected utility of taking a specific action in a specific state. The agents keep track of the experienced situation-action pairs by managing the so called Q-table, that consists of situation descriptions, the actions taken and the corresponding expected prediction, called Q-value.

Q-Learning is able to compare the expected utility of the available actions without requiring a model of the environment. Nevertheless, the use of the Q-Learning algorithm is constrained to a finite number of possible states and actions. As a reinforcement learning algorithm, it also is based on modeling the overall problem as Markov Decision Processes. Thus, it needs sufficient information about the current state of the agent for being able to assign discriminating reward. Although there are a number of extensions that improve the convergence speed of Q-Learning [5], we include the standard Q-Learning algorithms in our experiment due to its simplicity.

We suppose that Q-Learning meets the requirements for the application by providing both sufficient performance (if applicable) adaptability and also gives interpretability of the result. This interpretability is achieved by its rule-based structure (represented by the state action mapping) with a clear evaluation of those rules, by means of the Q-Value. The

processing of this mapping, weighted by the provided utility value could be used as a bias for the interpretation of the rules, as an input for the behavior modeling.

IV. TESTBED

The scenario we use for evaluating the learning architecture approach is the same as in [17] where we already describe the integration of XCS-based agents into the agent-based modeling and simulation platform SeSAM. This pedestrian evacuation scenario is a typical application domain for multi-agent simulation (see [18] for a real-world application). Albeit the employed scenario may be oversimplified, we expected that the relative simplicity of the scenario will enable us to evaluate the potentials of the learning technique as well as to deduce the involved challenges.

A. Environmental Model

The main objective of the simulation concerned the emergence of collision-free exiting behavior. Therefore, the reward and interfaces to the environment were mainly shaped to support this. In contrast to [17], we did not test a large variety of configurations as it was not the goal of this research to find an optimal one, but a more modeling-oriented evaluation of the architecture.

The basic scenario consists of a room (40x60m) surrounded by walls with one exit and a different number of column-type obstacles (with a diameter of 3.5m). In this room a number of pedestrians have to leave as fast as possible without hurting themselves during collisions. We assume that each pedestrian agent is represented by a circle with 50cm diameter and moves with a speed of 1.5m/sec. One time-step in the discrete simulation corresponds to 0.5sec. Space is continuous. We tested this scenario using 1, 5, 10 and 20 agents, and the number of obstacles was set to 10. At the beginning of a test-run, all agents were located at random positions in the upper half of the room.

All experiments alternated between explore and exploit phases. During the explore phase, the agents randomly execute an action. In exploitation trials, the best action was selected in each step. Every trial consists of 100 iteration steps. Every experiment took 1000 explore-exploit cycles.

Reward was given to the agent a immediately after executing an action at time-step t . It was computed in the following way: $reward(a, t) = reward_{exit}(a, t) + reward_{dist}(a, t) + feedback_{collision}(a, t) + feedback_{damage}(a, t)$ with $reward_{exit}(a, t) = 1000$, if agent a has reached the exit in time t , and 0 otherwise; $reward_{dist}(a, t) = \beta \times (d_t(exit, a) - d_{t-1}(exit, a))$ with $\beta = 5$; $feedback_{collision}(a, t)$ was set to 100 if a collision free actual movement had been made, to 0 if no movement happened, and to -100 if a collision occurred; $feedback_{damage}(a, t)$ was set to -1000 if a collision with column obstacle has occurred, and 0 otherwise. Together, the different components of the feedback function stress goal-directed collision-free movements. It is goal-directed because the agents are positively rewarded every time an action

results in reaching the exit or getting to a state closer to the exit. Complementary, it is collision-free oriented because the agents are positively rewarded for moving without collisions and negatively rewarded every time an action results in a collision.

B. Agent Interfaces

As agent interfaces, the perceived situation and the set of possible actions have to be defined. Similar to [17], the perception of the agents is based on their basic orientation of the agent, respectively its movement direction. The overall perceivable area is divided into 5 sectors with a distinction between areas in two different distances as depicted in figure 1. For every area two binary perception categories were used: the first encoded whether the exit was perceivable in this area and the second encoded whether an obstacle was present - where an obstacle can be everything with which a collision should be avoided: walls, columns or other pedestrians.

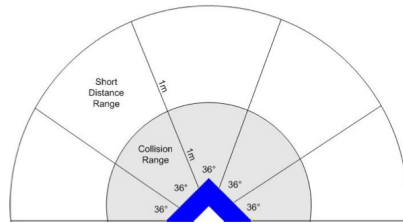


Fig. 1. Agent perception sectors

The action set is shaped for supporting the collision-avoidance behavior. We assume that the agents are per default oriented towards the exit. Thus, the action set consists of $A = \{Move_{Left}, Move_{SlightlyLeft}, Move_{Straight}, Move_{SlightlyRight}, Move_{Right}, Noop, Stepback\}$. For any of these actions, the agent turns by the given direction (e.g. $+36$ degrees for $Move_{SlightlyRight}$), makes an atomic step and orients itself towards the exit again. The combination of this action set and the perceptions of the agents represents an intentional simplification of the problem, as we implicitly represent the orientation task in the actions, in order to have a MDP. This simplification allows concentrating the learning on the collision avoidance, facilitating the learning process.

C. Architecture Configuration

The testbed was implemented in the visual modeling and simulation platform SeSAM (www.simsesam.de). The Q-Learning could be implemented by means of the standard high-level behavior language in SeSAM.

It was not our objective to find the optimal configuration for the tested architecture in the given scenario, we will not give a discussion of the effects of different parameter settings on the learning outcome should not be necessary. Clearly, we tested a number of configuration for finding a reasonable configuration. This is also true for the the appropriate overall configuration including different numbers of obstacles, sizes of scenarios or the particular numbers of the reward function.

In the context of this paper, we assume an initial Q-value of 0 for all untested state-action pairs. We set the learning rate to 0.5 and the discount factor to 0. It means that the agents' actions are selected based on recent experiences and not taking into consideration the future rewards (only the best action for the current state), respectively. This is another intentional simplification for the problem, as the agents don't need to maximize future rewards.

V. EXPERIMENTS AND RESULTS

In this section we analyze the results of the simulations, first with respect to learning performance showing that the learning technique is actually applicable to the test scenario, but then we focus on the analysis of what the agents actually did learn.

A. Performance Evaluation

The metric used for evaluating learning performance is the number of collisions. The time to reach the exit does not vary significantly, as a collision is not influencing the behavior directly, but indirectly via the reward the agent got. The collisions, with other pedestrians or obstacles, do not impose any effect on future movement. They only count as negative rewards. Obviously in the early stages, the agents don't have enough experience to learn from, and therefore a higher number of collisions is expected.

Table I presents the mean number of collisions for each tested situation. The values are aggregated only after the first 50 explore-exploit cycles for avoiding the inclusion of any warm-up data. The mean and deviation over the results of the different exploit cycles are given. Despite of having the runs repeated, we did not give means and standard deviations over different runs as currently the number of repetitions is too low. Clearly the number of collisions increases with the number of agents and obstacles.

TABLE I
MEAN NUMBER OF COLLISIONS PER RUN - ROWS REPRESENT THE NUMBER OF AGENTS AND COLUMN THE NUMBER OF OBSTACLES.

	10
1	0.01 ±0.23
5	1.39 ±1.78
10	6.66 ±3.88
20	25.17 ±8.77

Figure 2 illustrates the adaptation speed by depicting the number of collisions over time for an exemplary run with 5 agents and 10 obstacles. We can see that the number of collisions decreases fast in the beginning, but then the behavioral knowledge converges quite fast. After 50 cycles, there is no further improvement.

To have a better illustration of the learning process, we show in figure 3 the trajectories of the agents in exploit phases after a) 10, b) 100, c) 500 and d) 1000 exploit trials. In this figure we consider the situation with 5 agents and 10 obstacles. We can see the progress of adaptation with more and more collision-free and goal-directed movement. Experience hereby does not just mean positive reinforcement. Even if the agents

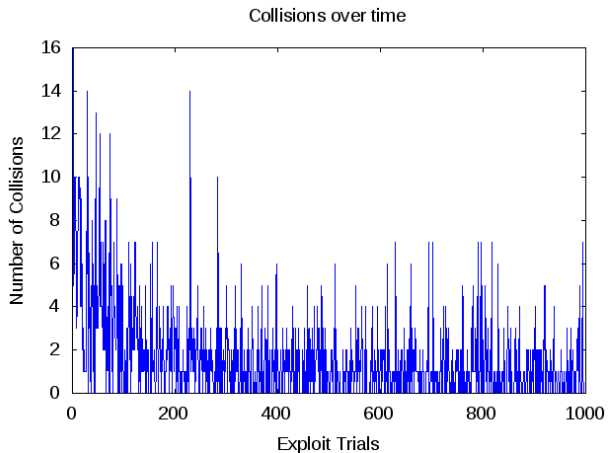


Fig. 2. Development of the number of collisions for an exemplary run with 5 agents and 10 obstacles

don't know what is the best action, they know which one to avoid by checking the negative rewarded actions.

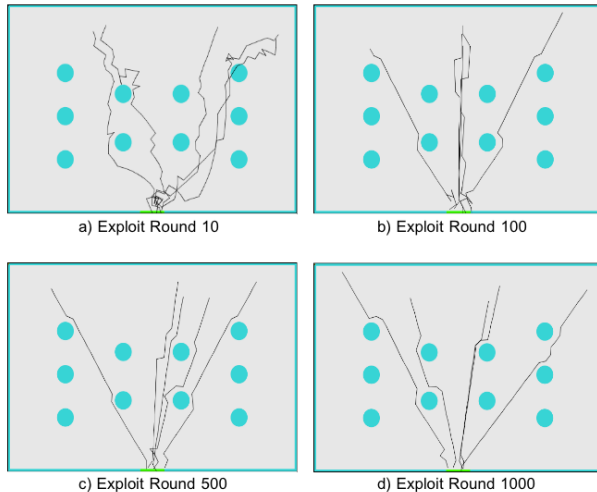


Fig. 3. Exemplary trajectories during exploit trials, for 5 agents and 10 obstacles

Alternating between explore and exploit trials plays an important role in the performance outcome. The agents must explore the possible actions set in order to maximize their experience in terms of the route to be chosen. At the end we can see the emergence of a collision-avoidance behavior.

B. Behavior Learning Outcome

In this section we are interested in analyzing the rules learned by the Q-Learning process in terms of the complexity of the resulting rule structure and potential use as source of inspiration in a modeling process.

In the following analysis we will examine two simulation scenarios: 1 agent and 10 obstacles; and 5 agent and 10

obstacles. In both cases we consider the outcome of one agent from an exemplary simulation.

1) *Raw Q-Learning Rules*: The rules generated by the learning process can be determined by taking for every situation the action with the highest q-value as it is done in the exploit phases. Depending on the situation, there might be no action with a positive q-value. The rules with a Q-Value of zero represent situation-action pairs that have not been tested during the simulation. Figure 4 depicts two out of 12 rules with the highest Q-value on the 1-agent scenarios.

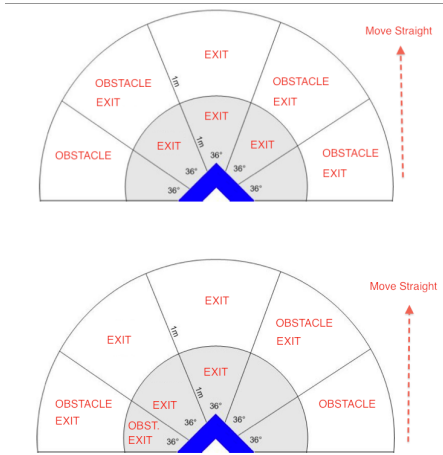


Fig. 4. Two out of 12 rules with the highest Q-value for the agent in the 1-agent scenario.

Figures 5 and 6 show the distribution of the reward prediction, i.e. the Q-value, for the complete rules set for the single agent, respectively a randomly selected exemplary agent from a simulation with 5 agents. One can see that there are only a few rules with a high Q-value.

It is obvious that the Q-value alone cannot be a selection criteria for rules forming a behavior model as the ones with the highest Q-value naturally contain situations where the agent directly perceives the exit. It is also possible to see that the agent in this case has a majority of rules with Q-Value 0, which means that a lot of state-action mappings have not been tested. This is not case for the simulation with 1 agent and 10 obstacles, as seen in figure 5, where the majority of rules have been tested. The agent has explored more, resulting in a more elaborated representation of the behavior. This difference is caused by the fact that the simulation with only 1 agent presented a smaller set of possible states to be tackled due to the simplicity of the interactions just with static obstacles.

Another important aspect about the agents' experience is that, since the agents are randomly positioned in the scenario at the beginning of each trial, the rules are not biased by a fixed position, so the rules set is more elaborated than it would be if they had to know only one best way to get to the exit.

The agent from the simulation with only one agent has a positive rules set – consisting of rules with positive, non-zero q-values – of 229 rules, while the agent from the simulation

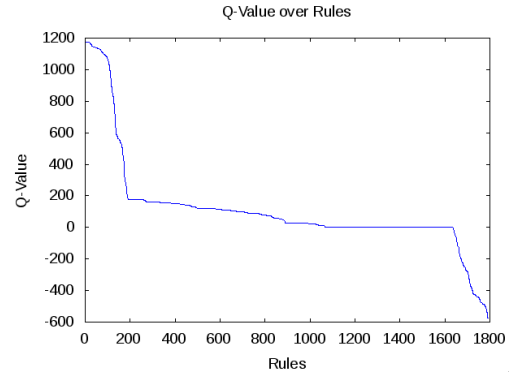


Fig. 5. Q-Learning value distribution for an exemplary agent from a simulation with 1 agents and 10 obstacles

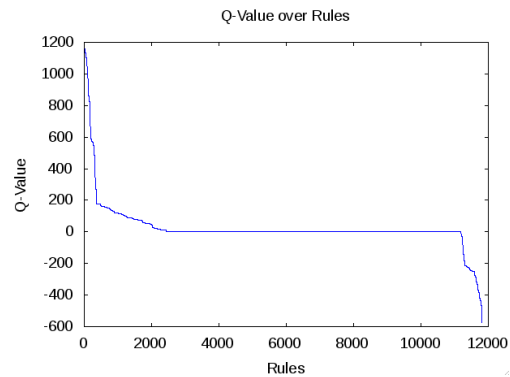


Fig. 6. Q-Learning value distribution for an exemplary agent from a simulation with 5 agents and 10 obstacles

with 5 agents has a number of 1507 true positive rules. This can be seen as an effect of the interaction with other agents, generating different situation to be visited, specially when it gets closer to the exit, the situation becomes more dense and the agents must avoid the collisions, and get to the exit.

Figures 7 and 8 show the distribution of these final rules over the possible actions, for the cases with 1 and 5 agents respectively. We can see the effect of the initial random positioning in each trial. We have a balanced distribution for the rules determining going to the left or right, which makes sense, since the agent must learn to find its way out of the scenario no matter where it has started. The majority of the rules indicate the *MoveStraight* action. This comes from the fact that the agent is reoriented towards the exit after the execution of any action. Unless the agent needs to avoid a collision, *MoveStraight* is the best action to choose.

We can identify the collision-avoidance behavior focussing on an exemplary element of the perceptions of the agent (1 agent scenario in this case). Considering action *MoveRight* and perception *ObstacleImmediatelyRight*, we see that there is a larger number of rules indicating *false* in this perception in all rules with the *MoveRight* action, see figure 9.

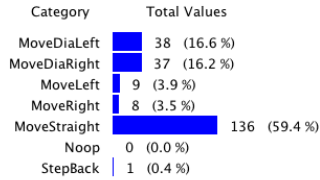


Fig. 7. Rules distribution over the actions for an exemplary agent from a simulation with 1 agent and 10 obstacles

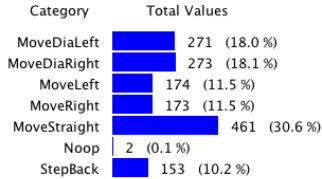


Fig. 8. Rules distribution over the actions for an exemplary agent from a simulation with 5 agents and 10 obstacles

2) *Processing the rules*: As the set of rules with truly positive Q-value in all scenarios is far too large to be transparently presented to a human expert, we suggest to use a post-processing step for improving the analysis of the rule set on a detailed level. As there are a number of candidates that may be suitable for generalizing the rule set in a way that all learnt rules are captured in a compact form.

For this aim, we tested three different machine learning algorithms – mainly classification learners – using all rules with non-zero, positive Q-Value: K Nearest Neighbors (KNN) [19], CART Decision Trees [20] and the CN2 rule inductor [21]. The K-Nearest Neighbors is arguably one of the simplest machine learning algorithms, while Decision Trees and CN2 are of particular interest to this work because of the interpretability provided by their resulting representation of the knowledge captured in the training set. We used KNN with a K value of 5 for the experiments. The Decision Tree is a simple CART with Gini’s index of impurity for node splitting. CN2 algorithm uses the Laplace method for rule quality estimation.

As mentioned above, the results of this post-processing step have to be evaluated with two criteria: How well they capture the given rule set and how good they are able to generalize the rule set for bringing the rules. The first can be measured in terms of classification accuracy, the second is the generalization and compactness of the resulting behavior description.

a) *Classification Accuracy*: Table II shows the classification accuracy for the above mentioned algorithms, both in the 1 agent and 5 agents experiments, using 10 fold cross validation in the training set. Table III shows the average classification accuracy, when model built from one agent’s experience is tested with another agent’s experience: We can see that the classification accuracy for the case with 1 agent outperformed the case with 5 agents. This is clearly an effect of the exploit-

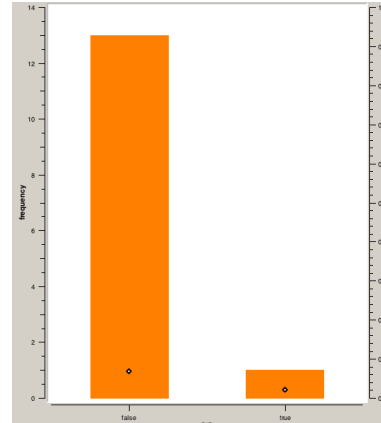


Fig. 9. Frequency of rules with perception *ObstacleImmediatelyRight* as false (left bar) and true (right bar) for action *MoveRight*

explore tradeoff. The agent from the 1 agent simulation has a lower number of states to visit during the simulation, and this reflects on the accuracy of the rules as they are tested more times and converge faster to the optimal solution (state-action mapping). The agents from the 5 agents scenario have a larger set of states that potentially may occur, reflected also in the number of rules. This requires more cycles to converge to an optimal solution.

TABLE II
CLASSIFICATION ACCURACY - 10 FOLD CROSS VALIDATION

	KNN	Decision Tree	CN2
1 agent	0.6593	0.6375	0.6334
5 agents	0.2907	0.2654	0.2980

TABLE III
CLASSIFICATION ACCURACY - VALIDATION AMONG DIFFERENT AGENTS

	KNN	Decision Tree	CN2
1 agent	0.6724	0.6983	0.6897
5 agents	0.3098	0.3230	0.3316

While they are all good models – as providing a solution to the problem (as seen in section V-A) – they can not be generalized to other good solutions (other agents’ experiences). The convergence of the solution, which determines its generalization to the problem is therefore a function of the configuration of the learning, and more important, a function of the explore-exploit distribution, the number of agents and the set of perceptions and actions, that determine the size of the state-action mapping.

Figure 10 shows the confusion matrix for the decision tree learnt from the simulation with 1 agent, testing with cross-validation: Rows represent the expected class (action) from the classification model, as presented in the Q-Learning mapping and columns represent the classification determined by the decision tree. We highlight the number of correctly classified instances. The majority of misclassified instances

learning algorithms: K-Nearest Neighbors, Decision Trees and CN2 rule inductor. The resulting, full behavior model for the Q-Learning is only partially helpful as a guidance for modeling in this case. Generalization still needs to be improved, as a part of the learning process or as a post-processing step. This could be achieved by using more flexible classification techniques, such as multi label classification, since in this process we have to deal with multiple good solutions. Another important aspect to be considered here is the tradeoff between explore and exploit, and how this scales to the complexity of the problem, in terms of the number of agents and the size of the state-action mapping in a multiagent simulation. This is a relation yet to be analyzed in detail level.

There are admittedly many more challenging application scenarios than an evacuation scenario where all agents have the same goal, the behavior repertoire is quite restricted, and there is no direct communication between agents. In such advanced environments, the learning and environment design will certainly pose additional challenges.

Our next steps include testing other learning techniques to investigate their performance, outcome and appropriateness for this methodology. A short analysis of Learning Classifier Systems and Neural Networks can be found in [3]. We plan to also test approaches such as evolutionary programming support vector machines, and other forms of reinforcement learning, respectively learning automata. An alternative for the post-processing step worth testing could be multi label classification [22], where we could gather the experience from different agents and find different best actions for a given situation, increasing generalization.

Besides that, we will pursue further self-modeling agent experiments. We are considering the application of the learning technique in other, more complex scenarios, such as an evacuation of a train with about 500 agents, complex geometry with exit signs and time pressure. We are also interested in a scenario where cooperation / collaboration is required, in order to investigate the possible emergence of the cooperation in the agent model, through the learning process. This experimentation should consider situations with and without direct communication between the agents.

REFERENCES

- [1] M. Fehler, Klügl, and F. Puppe, "Approaches for resolving the dilemma between model structure refinement and parameter calibration in agent-based simulations," in *AAMAS '06: Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems*. ACM Press, 2006, pp. 120–122.
- [2] F. Klügl, "Multiagent simulation model design strategies," in *MAS&S Workshop at MALLOW 2009, Turin, Italy, Sept. 2009*, ser. CEUR Workshop Proceedings, vol. 494. CEUR-WS.org, 2009.
- [3] R. Junges and F. Klügl, "Agent architectures for a learning-driven modeling methodology in multiagent simulation," in *MATES 2010: Proceedings of the 8th German Conference on Multiagent System Technologies (to appear)*, 2010.
- [4] G. Weiß, "Adaptation and learning in multi-agent systems: Some remarks and a bibliography," in *IJCAI '95: Proceedings of the Workshop on Adaption and Learning in Multi-Agent Systems*. London, UK: Springer-Verlag, 1996, pp. 1–21.
- [5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [6] A. Nowe, K. Verbeeck, and M. Peeters, "Learning automata as a basis for multi agent reinforcement learning," pp. 71–85, 2006.
- [7] C. Adami, *Introduction to artificial life*. New York, NY, USA: Springer-Verlag New York, Inc., 1998.
- [8] J. Grefenstette, "The evolution of strategies for multi-agent environments," *Adaptive Behavior*, vol. 1, pp. 65–90, 1987.
- [9] R. J. Collins and D. R. Jefferson, "Antfarm: Towards simulated evolution," in *Artificial Life II*. Addison-Wesley, 1991, pp. 579–601.
- [10] J. Denzinger and M. Fuchs, "Experiments in learning prototypical situations for variants of the pursuit game," in *In Proceedings on the International Conference on Multi-Agent Systems (ICMAS-1996*. MIT Press, 1995, pp. 48–55.
- [11] Y. Maeda, "Simulation for behavior learning of multi-agent robot," *Journal of Intelligent and Fuzzy Systems*, pp. 53–64, 1998.
- [12] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artificial Intelligence*, vol. 55, no. 2-3, pp. 311 – 365, 1992.
- [13] M. R. Lee and E.-K. Kang, "Learning enabled cooperative agent behavior in an evolutionary and competitive environment," *Neural Computing & Applications*, vol. 15, pp. 124–135, 2006.
- [14] R. Neruda, S. Slusny, and P. Vidnerova, "Performance comparison of relational reinforcement learning and rbf neural networks for small mobile robots," in *FGCNS '08: Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking Symposia*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 29–32.
- [15] J.-P. Georg, G. Picard, M.-P. Gleizes, and P. Glize, "Living Design for Open Computational Systems," in *International Workshop on Theory And Practice of Open Computational Systems (TAPOCS) at 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)*, M. Fredriksson, A. Ricci, R. Gustavsson, and A. Omicini, Eds. Linz, Austria: IEEE Computer Society, June 2003, pp. 389–394.
- [16] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.
- [17] F. Klügl, R. Hatko, and M. V. Butz, "Agent learning instead of behavior implementation for simulations - a case study using classifier systems," in *MATES 2008: Proceedings of the 6th German Conference on Multiagent System Technologies*. Springer Berlin / Heidelberg, 2008, pp. 111–122.
- [18] F. Klügl, G. Klubertanz, and G. Rindsfuser, "Agent-based pedestrian simulation of train evacuation integrating environmental data," in *KI 2009: Advances in Artificial Intelligence, 32nd Annual German Conference on AI, Paderborn, Germany, September 15-18, 2009. Proceedings*, ser. Lecture Notes in Computer Science, vol. 5803. Springer, 2009, pp. 631–638.
- [19] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
- [20] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, 1st ed. Chapman and Hall/CRC, January 1984.
- [21] P. Clark and T. Niblett, "The cn2 induction algorithm," *MACHINE LEARNING*, vol. 3, no. 4, pp. 261–283, 1989.
- [22] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int J Data Warehousing and Mining*, vol. 2007, pp. 1–13, 2007.