2015

An agent-mediated platform for business processes

Hoa Khanh Dam
University of Wollongong, hoa@uow.edu.au

Aditya Ghose
University of Wollongong, aditya@uow.edu.au

Mohammad Qasim
University Of Wollongong

Publication Details

Dam, H. Khanh., Ghose, A. & Qasim, M. (2015). An agent-mediated platform for business processes. International Journal of Information Technology and Web Engineering, 10 (2), 43-61.

An agent-mediated platform for business processes

Abstract
Business processes have been widely becoming crucial assets of organisations across various industries and domains. The flexibility in dealing with changes when business processes are executed has significant impact on the success of an organisation's business operations, especially in the current ever-changing business environment. In this context, agent-based systems offer a promisingly powerful platform for business process execution. In this paper, the authors propose an agent-mediated platform for business processes with the aim to contribute to bridge the gap between business process management and agent-oriented development. They present a conceptual mapping method for a seamless transition from business process models in Business Process Modelling Notation (BPMN) to agent-oriented models in the Prometheus methodology, which is implemented using the ATLAS Transformation Language. The authors also developed an Eclipse-based plug-in which allows the designer to import BPMN models into the Eclipse-based Prometheus Design Tool.

Disciplines
Engineering | Science and Technology Studies

Publication Details
Dam, H. Khanh., Ghose, A. & Qasim, M. (2015). An agent-mediated platform for business processes. International Journal of Information Technology and Web Engineering, 10 (2), 43-61.

An Agent-Mediated Platform for Business Processes

Hoa Khanh Dam, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, Australia

Aditya Ghose, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, Australia

Mohammad Qasim, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, Australia

ABSTRACT

Business processes have been widely becoming crucial assets of organisations across various industries and domains. The flexibility in dealing with changes when business processes are executed has significant impact on the success of an organisation's business operations, especially in the current ever-changing business environment. In this context, agent-based systems offer a promisingly powerful platform for business process execution. In this paper, the authors propose an agent-mediated platform for business processes with the aim to contribute to bridge the gap between business process management and agent-oriented development. They present a conceptual mapping method for a seamless transition from business process models in Business Process Modelling Notation (BPMN) to agent-oriented models in the Prometheus methodology, which is implemented using the ATLAS Transformation Language. The authors also developed an Eclipse-based plug-in which allows the designer to import BPMN models into the Eclipse-based Prometheus Design Tool.

Keywords: *Agent-Oriented Software Engineering, Business Process Design, Business Process Execution, Business Process Management, Multi-Agent Systems*

1. INTRODUCTION

A business process is defined as consisting of a set of activities, performed by their relevant roles or collaborators, to intentionally achieve the common business goals (Smith & Fingar, 2003). Business processes are the core assets of any enterprise since they generate revenue and often represent a significant proportion of costs. A recent study (Hill, Cantara, Deitert, & Kerremans, 2007) has shown that the business process management (BPM) software market reached nearly $1.7 billion in total software revenue in 2006 and this number continues to grow. The flexibility and reactivity in process execution through IT-systems have significant impact on the success of an organisation's business operations, especially in the current constantly changing business environment. Existing BPM systems, which require a priori representation of a business process and all potential deviations from that process, however, do not provide adequate support

DOI: 10.4018/IJITWE.2015040103

to achieve these requirements in a satisfactory way (Burmeister, Arnold, Copaciu, & Rimassa, 2008; Fingar, 2008).

A software agent (Wooldridge, 2002) is a piece of software which is situated in an environment, autonomous (i.e. acts on its own), social (interacts with other similar entities), and being reactive (responding to changes in its environment) and/or being proactive (working to achieve its goals). Multi-Agent systems (MAS) provide powerful and flexible execution platform for business processes. On such a platform, the executing process is able to pursue persistent goals over time despite previously failed attempts due to the availability of multiple ways of dealing with a given goal, as in the Beliefs-Desires-Intentions (BDI) platform (Rao & Georgeff, 1995). The agent view also provides an intuitive, well-suited level of abstraction for modelling and implementing process execution. For instance, a process participant can be explicitly represented as an agent at the execution level, which reflects more accurately the structure of an organisation. Existing process execution platforms such as the Business Process Execution Language[1] (BPEL) do not capture such structural information at the business level (Endert, Küster, Hirsch, & Albayrak, 2007).

Since the late 1980s, the field of agent technology has attracted a substantial amount of interest from researchers (Jennings, Sycara, & Wooldridge, 1998; Luck, McBurney, Shehory, & Willmott, 2005). In particular, there have been a number of different agent theories, architectures, and languages proposed in the literature. Agent technology, however, still faces many challenges in being adopted by the industry despite its popularity and attractiveness as a research area (Weyns, Parunak, & Shehory, 2009). Therefore, closing the gap between the business community, BPM in particular, and agent technology can bring substantial benefits to both sides: agents gaining better industry traction whilst BPM having a powerful solution to deal with its current challenges (Ghose, 2009).

Although it is possible to execute business processes with traditional business process management systems, the use of MAS for implementation of business processes offers several key advantages. Business processes tend to be initially designed in such a high-level, abstract manner which does not provide in depth understanding of the process and its ability to achieve desired goals. Such processes usually capture only the normal/main behaviour and tend to miss out on exceptional alternatives. We believe that translating business process models to an agent-based platform would provide a mediation facility to help business process designers explore alternative flows in the process. Therefore, in this paper our focus is on proposing a mapping between business process models specified in Business Process Modelling Notation[2] (BPMN) to concepts and artifacts of the Prometheus agent-oriented methodology (Padgham & Winikoff, 2004). We have chosen BPMN since it is a standard for business process modelling and has been widely used and supported in numerous modelling tools. The choice of Prometheus is due to various reasons: its wide use in both industrial and academic settings, considerable detailed support for most of the software engineering development phases, and the availability of tool support, Prometheus Design Tool[3] (PDT). We take the model transformation approach and implemented the mapping using Atlas Transformation Language (Jouault, Allilaire, Bezivin, & Kurtev, 2008). This transformation forms the main part of an Eclipse-based plugin that we have developed for PDT to allow the designer to import BPMN models.

The organization of this paper is as follow. In section 2, we provide a brief description of BPMN and the Prometheus methodology. We then discuss our approach to transform BPMN models to Prometheus models in section 3. Section 4 provides a description of a plug-in for Prometheus Design Tool that supports the transformation. Related work is presented in section 5. Finally, we conclude and discuss some directions for our future work in section 6.

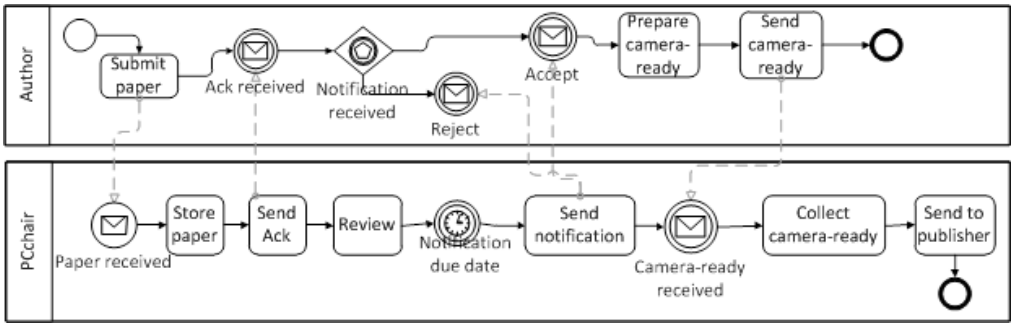
2. BACKGROUND

2.1. Business Process Modelling

The Business Process Modelling Notation (BPMN) is a standard for business process modelling (Object Management Group, 2009). It provides graphical notation for specifying various types of activities, decision responsibilities, control and data flow in business process within one organization and in cross-organizational settings. There are more than 60 products[4], both commercial and open-source, providing the implementation of the BPMN standard. BPMN has been widely used in the industry due to its powerful notation which is readily understandable by both the business stakeholders and the technical developers. Figure 1 shows a BPMN diagram describing a typical conference management process. The author submits a paper to the PC Chair who will store the paper, send an acknowledge receipt to the author, distribute the papers to reviewers[5] . After the notification due date, the PC Chair sends out the notification which can be either reject or accept. If the paper is accepted, the author prepares a camera-ready version and sends it to the PC Chair who will collect all the camera-ready papers and send them to the publisher for printing proceedings. We use this as an example to illustrate the key components of BPMN. Conference management systems are industrial systems that involve many process participants. The example that we use here has many key characteristics in a business process model which also exist in an industrial business process.

There are three major categories of notational elements in BPMN: *Flow Objects*, *Connecting Objects* and *Swimlanes*. The elements known as *Flow Objects* are: *Events*, *Activities*, and *Gateways*. An event denotes something that happens during the course of a business process, e.g. receipt of a message, expiration of a timer, detection of an error, etc. Each event usually has a cause (trigger) and/or an impact (result). An internal icon within an event symbol is used to differentiate the various types of triggers and results. For instance, a *TimerEvent* represents a specific time-date or a specific cycle (e.g., every Monday at 9am) while a *MessageEvent* represents a message arriving or being sent from a participant. With regard to when events affect the flow, there are three types of Events: *Start* (a trigger of the process), *End* (the result of a process), and *Immediate* (something that happens between the start and end events). The business process represented in Figure 1 has a number of events. For instance, "Ack Received" is a Message and Immediate Event whilst "Paper received" is also a Message but Start Event triggering the business process of a "PC chair". There is also a Timer Event "Notification due date" denoting the arrival of the date for paper notification.

Figure 1. A process of submitting papers to a conference



Activity is another type of Flow Objects which describes the kind of work which must be performed. A *Task* is an atomic activity while a *Sub-Process* is a composite activity, which contains additional levels of business process detail. In Figure 1, "Submit paper" and "Prepare camera-ready" are examples of activity tasks. A *Gateway* is used to control the divergence and convergence of *Sequence Flow* and determines branching, forking, merging, and joining of paths. There are different types of gateways (exclusive, inclusive, complex, and parallel) and the behaviour of each type gateway specifies how many of the gates will be available for the continuation of flow. Figure 1 has an event-based exclusive gateway "Notification received" modelling decisions that are based on the events received, i.e. accept or reject.

Flow Objects are connected in three different ways: by sequence flows, message flows or association. *Sequence Flows* are used to show the order in which a series of flow objects have to be completed. A *Message Flow* describes the exchange of messages between two process participants while an *Association* is used to associate information (e.g. an *Artifact* or text) with Flow Objects. Process participants are represented in a BPMN diagram as a *Pool*. For instance, there are two pools, i.e. "Author" and "PC Chair", in the process in Figure 1 and they exchange messages with one another, e.g. the "Submit paper" task in the "Author" pool sends a message to the "PC chair" pool. In order to separate different processes, a pool can be sub-partitioned into swimlanes. Finally, Artifacts are used to provide additional information about the process. A typical artifact in BPMN is Data Object which represents data required or produced in an activity.

2.2. Intelligent Software Agents and Prometheus Methodology

The concepts associated with agents, also called *software agents* or *intelligent agents*, have been discussed for many years within the Artificial Intelligence community. A software agent (Wooldridge, 2002) is generally considered to have the following basic characteristics:

- **Situatedness:** Agents are embedded in an environment in terms of using their sensors to perceive the environment and using their effectors to affect the environment. For instance, a robot soccer player can be designed as an agent, which is situated in a soccer field. One of the robot's sensors is a group of cameras that keep track of where the ball and other players are. The robot agent also has several effectors such as its legs or its body, which are used to kick or pass the ball;
- **Autonomy:** Agents are able to operate independently, i.e. decide which action they should take, independent of humans or other agents. As a result, agents cannot be directly invoked like objects [Odell, 2002]. In our robot soccer player example, when a robot agent has the ball, the decision whether to kick the ball for a goal or to pass the ball to its teammates is totally up to the agent. This is an example of the autonomy of agents due to the fact that those decisions are made without direct intervention of humans or other robot soccer agents on the field;
- **Reactivity:** Agents can perceive their environment and respond in a timely fashion to changes that occur in it. For example, when our robot soccer player detects the ball being within its control area, it has to quickly perform some actions (to respond to that event) such as passing or shooting the ball;
- **Pro-activeness:** Agents are pro-active if they have goals that they pursue over time. In our example, the major goal of a robot soccer player is to win the game, which can be achieved by scoring goals and defending against conceding goals. The agents pursue this goal by performing actions (e.g. passing the ball to their other teammates, kicking for goals, etc.) that contribute toward accomplishment of the goal;

- **Social ability:** Agents can interact with other agents and humans with the aim of accomplishing their goals. In our example, social ability is reflected by the fact that each robot soccer agent should be able to communicate and coordinate with their teammates, or their coaches (which may be humans).

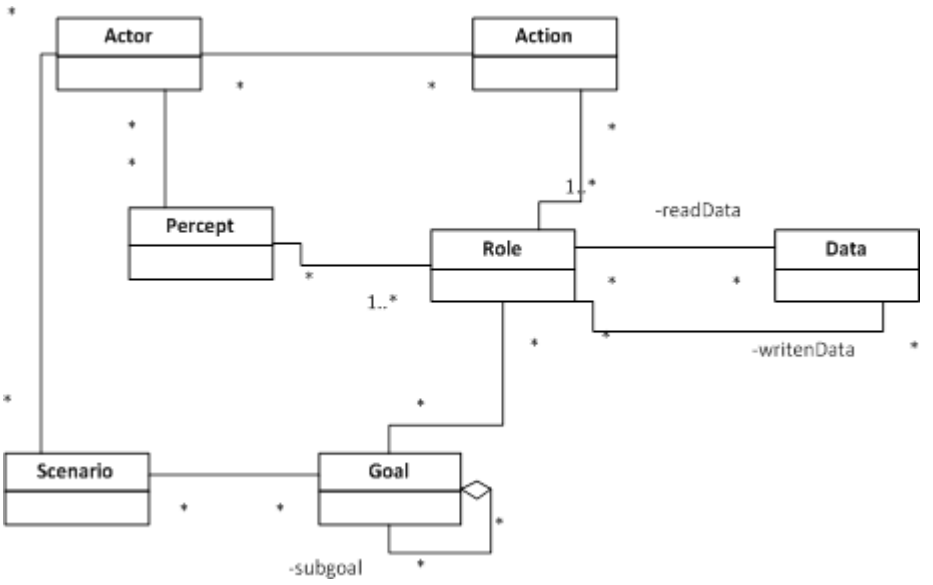
Note that it may not be necessary for an agent to have all the above characteristics. Agents must be autonomous and social but it can be reactive or proactive or both and it can be situated or communicative. Agent Oriented Software Engineering (AOSE) is a promising approach to software engineering that uses the notion of agents as the primary method for analysing, designing and implementing software systems (Jennings, 2001). The effectiveness of AOSE resides in its ability to translate the distinctive features of agents (e.g. autonomy, reactiveness, pro-activeness, etc.) into useful properties of (complex) software systems and to provide an intuitive metaphor that operates at a higher level of abstraction compared to the object oriented model. Agent-oriented software engineering has been growing rapidly in the past few years. There is strong international interest in proposing methodologies, notations and programming languages for the analysis, design and implementation of agent systems (Henderson-Sellers & Giorgini, 2005; Bordini, Dastani, Dix, & Fallah-Seghrouchni, 2005; Bergenti, Gleizes, & Zambonelli, 2004). For example, a large number of agent-oriented methodologies have been proposed such as Tropos (Bresciani, Perini, Giorgini, Giunchiglia, & Mylopoulos, 2004), Gaia (Zambonelli, Jennings, & Wooldridge, 2003), O-MaSE (DeLoach, 2005), ADELFE (Bernon, Gleizes, Peyruqueou, & Picard, 2002; Rougemaille, Migeon, Maurel, & Gleizes, 2008) and Prometheus (Padgham & Winikoff, 2004).

One of the most prominent agent-oriented software engineering methodologies which have been used and developed over a number of years is Prometheus (Padgham & Winikoff, 2004). The methodology is complete, described in considerable detail, and has tool support, i.e. Prometheus Design Tool (PDT). We briefly describe the methodology in this section and refer the reader to (Padgham & Winikoff, 2004) for further details. In section 3 we will also show some examples of artefacts (i.e. diagrams) that are developed using Prometheus. There are three phases in the Prometheus methodology: *system specification*, *architectural design* and *detailed design*.

The system specification phase involves: identifying *actors* and their interaction with the multi-agent system; developing *scenarios* illustrating the system's operation; identifying system *goals* and *subgoals*; and grouping goals into the basic *roles* of the system. Actors are any stakeholders who will interact with the system to achieve some goals, and can be humans or other software systems. For each actor, *percepts* which are inputs from the actor to the agent system are identified. In addition, outputs from the system to actors (*actions*) are identified. The interaction between each actor and the system is described using scenarios in Prometheus. Each interaction scenario is described in a structured form which includes a sequence of steps, where each step can be an action being performed by a role, a percept being received by a role, a goal being achieved by a role, or a sub-scenario[6] . Figure 2 shows an excerpt of a Prometheus metamodel adopted from (Dam, 2009) which describes the relationships between those key concepts at the specification phase of Prometheus.

Between the system specification phase and the detailed design phase where the system is modelled as computational entities which are suitable for a particular agent platform, Prometheus provides an intermediate phase called architectural design. The major purpose of the architectural design phase in Prometheus is to identify the agent types within the agent system and the interactions between these agent types. The main steps of this phase are: determining what *agent types* will be implemented by grouping related roles; developing the *interaction diagrams* and

Figure 2. An excerpt of a Prometheus metamodel



interaction protocols that define the intended valid sequences of messages between agents; and developing the *system overview diagram* which captures the system's overall (static) structure.

The final stage of the Prometheus methodology is the detailed design. The internal structure of each agent and how it will accomplish its goals within the overall system are addressed in this phase. Specifying agent internals in Prometheus is a process of progressive refinement, including the following activities: defining and developing *capabilities* (modules within agents) and their relationships; developing process diagrams depicting the internal processing of each agent related to the *protocol* specifications; and developing *plans*, *events*, and *data* and their relationship.

3. TRANSFORMATION FROM BPMN MODELS TO PROMETHEUS MODELS

Due to the high-level, abstract nature of business process models, there is a significant gap between those models and an executable agent system. Agent-oriented methodologies such as Prometheus provide concepts and techniques to close the gap between high-level system requirements and low-level executable agent system. The very first stage in Prometheus, the system specification phase, involves identifying system goals, defining the interfaces between the agent system with its environment, developing scenarios, and establishing system roles. There are close relationships between business process models and the concepts and artifacts in this system specification of the Prometheus methodology. In our previous work (Dam & Ghose, 2010), we have proposed some preliminary ideas of how details contained in BPMN models can be directly translated to Prometheus concepts. In this paper, we further develop those ideas and formalize them as a set of transformation rules. These transformation rules have been developed based on deep analysis of the semantics of BPMN models and Prometheus system specification models.

3.1. Introducing the System-to-Be

Most existing business process models available in an organisation describe an "as-is" system in terms of how different stakeholders are currently operating to achieve common business goals. For instance, the business process model in Figure 1 depicts the process of submitting papers, reviewing and collecting camera-ready papers which takes place between an author and a PC Chair, two typical stakeholders of a conference. The next step in the requirements phase is extending that conceptual model by including the target system or the system-to-be. A number of methodologies have proposed to follow these two steps at the requirement phase: for example the early requirements and late requirements phases proposed in the Tropos methodology (Bresciani et al., 2004) using the i* model (Yu, 1995).

For BPMN models, we propose to represent the system-to-be as a pool which may be divided in a number of swimlanes. The process within this pool and its dependencies with the other pools define the functionalities of the system-to-be. For instance, Figures 3 and 4 show BPMN models which are extended to include the system-to-be, a conference management system (CMS). The CMS is represented as a pool which contains three swimlanes: "Submission Management" for handling paper submissions, "Review Management" for dealing with the reviewing and "Proceeding Management" for the final paper printing. In Figure 3, "review" is shown as a sub-process activity and its details are presented in Figure 4.

The BPMN models were developed by adapting the description of a conference management system presented in (DeLoach, 2002): "*the Conference Management System is an open multiagent system supporting the management of various sized international conferences that requires the coordination of several individuals and groups. There are four distinct phases in which the system must operate: submission, review, decision, and final paper collection. During the submission phase, authors should be notified of paper receipt and given a paper submission*

Figure 3. A process between an author and the conference management system (CMS)

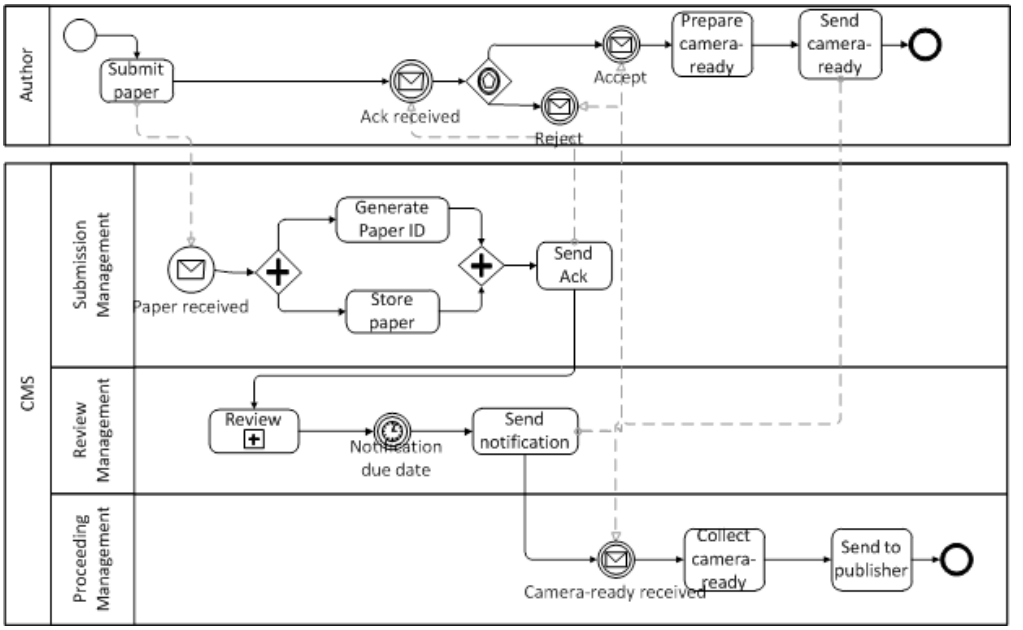
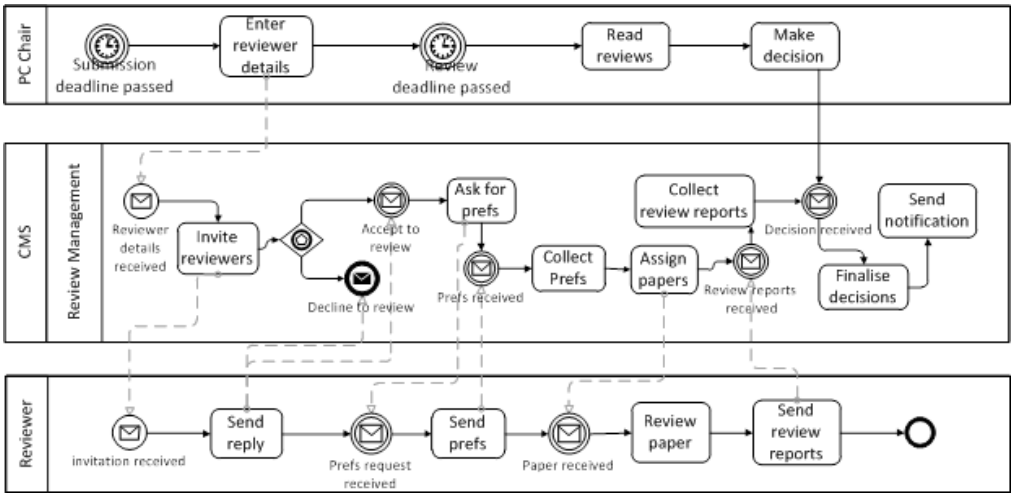


Figure 4. A reviewing process between a PC chair, a reviewer and the conference management system (CMS)



number. After the submission deadline has passed, the program committee (PC) has to review the papers by either contacting referees and asking them to review a number of the papers, or reviewing them themselves. After the reviews are complete, a decision on accepting or rejecting each paper must be made. After the decisions are made, authors are notified of the decisions and are asked to produce a final version of their paper if it was accepted. Finally all final copies are collected and printed in the conference proceedings". We modified this slightly to assume that the PC chair contacts directly the reviewers to distribute papers for reviewing.

3.2. Mapping BPMN to Prometheus

In this section, we present a set of transformation rules for BPMN elements and their counterparts in Prometheus. These rules are in the form of **LHS ⇒ RHS**, where LHS (left-hand side) is an element in the source model, and RHS (right-hand side) contains elements that are added to the target model. The rules are also preferred to be applied in the order from 1 to 6.

A BPMN pool represents a process participant, which as seen in Figures 3 and 4 can be the system or external entities interacting with the system (which we refer to as "non-system" pools). Such pools can be mapped directly to Prometheus actors by applying rule 1. Note that P is the set of pools in a given BPMN model and A is the set of actors in a Prometheus model. In addition, a.name denotes the name of actor a and ρ.name denotes the name of pool ρ. For instance, there are three actors of the Conference Management System: "author", "PC Chair", and "reviewer", corresponding to the three pools that interact with the CMS.

Rule 1 (Pool to Actor): Let $\rho \in P$ be a non-system pool of a given BPMN model.

$\rho \Rightarrow$ a (where a is an actor):
a.name::= ρ.name
A::= A∪{a}

A lane is situated in a pool and represents a group of similar behaviour within a process participant. A role in Prometheus also represents chunks of behaviour which includes a grouping of related goals, percepts, actions and data relevant to the behaviour. Therefore, BPMN lanes which belong to a system pool are mapped directly to roles in Prometheus by applying rule 2. Note that L is the set of lanes in a given BPMN model and R is the set of roles in a Prometheus model. For example, the three swimlanes in the CMS pool, namely "Submission Management", "Review Management" and "Proceeding Management", can be translated into three equivalent roles. The "Submission Management" role is responsible for dealing with paper submission (e.g. receiving papers, generating paper ids, sending acknowledgement, etc.), the "Review Management" role for the paper reviewing (inviting reviewers, allocating papers, making decisions, etc.), and the "Proceeding Management" for organising conference proceedings (e.g. collecting camera-ready papers, contacting publishers, etc.).

Rule 2 (Lane to Role): Let $\iota \in L$ be a lane of a given pool ρ in BPMN model and ρ is a system pool, i.e. ρ.type = system:

$\iota \Rightarrow$ r (where r is a role)
r.name::= ι.name
R::= R$\cup${r}

Message Event (a type of an activity) which represents the arrival of a message from a participant (i.e. a pool) may trigger the start of the process (Start Event) or cause the process to continue (Intermediate Event). Such message events, if takes place within a system pool, represent information from the environment that the system receives. Hence, those message events can be transformed into percepts in Prometheus. Start Message Events can be translated to percepts that trigger a particular scenario while Intermediate Message Events are mapped to other types of percepts. In addition, the actor corresponding to the pool where this message is sent provides the corresponding percept while the role corresponding to the lane where this message event is located is responsible for handling the percept. This transformation is formally described in rule 3. Note that A is the set of activities in a given BPMN model while P is the set of percepts in a Prometheus model. In addition, *p.actor* denotes the set of actors producing percept *p* and *p.role*[7] denotes the set of roles handling percept *p*. For example, in Figure 3 there are two Message Events in the CMS pool which can be translated to Prometheus percepts: "paper received" and "camera-ready received" while in Figure 4 they are "reviewer details received", "accept to review", "decline to review", "prefs received", "review reports received" and "decision received".

Rule 3 (MessageEvent to Percept): Let $\alpha \in A$ be a message event sent from pool $\rho 1$ and located in pool $\rho 2$, i.e. α.type = EventStartMessage | EventIntermediateMessage, and $\rho 2$.type = system. Also, let l be the lane where α is located. Assume that a and r are respectively the actor and role directly mapped to $\rho 1$ and l:

$\alpha \Rightarrow$ p (where p is a percept)
p.name::= α.name
P::= P$\cup${p}
p.actor::= p.actor $\cup${a}
p.role::= p.role $\cup${r}

An Intermediate Message Event can be used to represent the sending of a message to a process participant. If the sender is the system pool and the receiver is a non-system pool, then such a message event represents an output from the system to an actor. Therefore, those events can be transformed to actions, which is formally defined in rule 4. Note that AC is the set of actions in a Prometheus model. For example, in Figure 3 there are two Message Events in the CMS pool which can be translated to Prometheus actions: "ack received", "accept", and "reject" while in Figure 4 they are "invitation received", "prefs request received", and "paper received".

Rule 4 (MessageEvent to Action): Let $\alpha \in A$ be a message event sent from pool $\rho 1$ and located in pool $\rho 2$, i.e. α.type = EventStartMessage | EventIntermediateMessage, and $\rho 2$.type = system. Also, let l be the lane where α is originated. Assume that a and r are respectively the actor and role directly mapped to $\rho 1$ and l. and $\rho 2$ is a system pool, i.e. ρ.type = system:

$\alpha \Rightarrow$ ac (where ac is a percept)
ac.name::= α.name
AC::= AC $\cup$ {ac}
ac.actor::= ac.actor $\cup$ {a}
ac.role::= ac.role $\cup${r}

A process activity is an element on a process model that indicates certain things need to be done for the process to progress toward achieving its goals. In this sense, each activity can be mapped to a (sub-)goal in Prometheus. It is also noted that similarly to the labelling of activities in BPMN models, system goals in Prometheus are usually named in the optative mood (i.e. a desire), e.g. "inform authors". In addition, each end event of a complete process in a business process model leads to the achievement of a goal. Therefore, a process or sub-process can be mapped to a goal and its sub-goals are related to the activities within the process. Business process models also contain decision gateways. This can be translated to the AND/OR goal refinement in Prometheus. Doing such a transformation, one should be able to construct a system goal hierarchy. Figure 5 shows a goal diagram for the CMS. As can be seen, "manage conference" is the top goal which contains a number of sub-goals, including "submitting papers", "print proceeding" and "reviewing papers". Each of these sub-goals is translated from a corresponding process or sub-process. For instance, "reviewing papers" is a sub-process in the business process model and its tasks (e.g. "invite reviewers") are mapped to sub-goals.

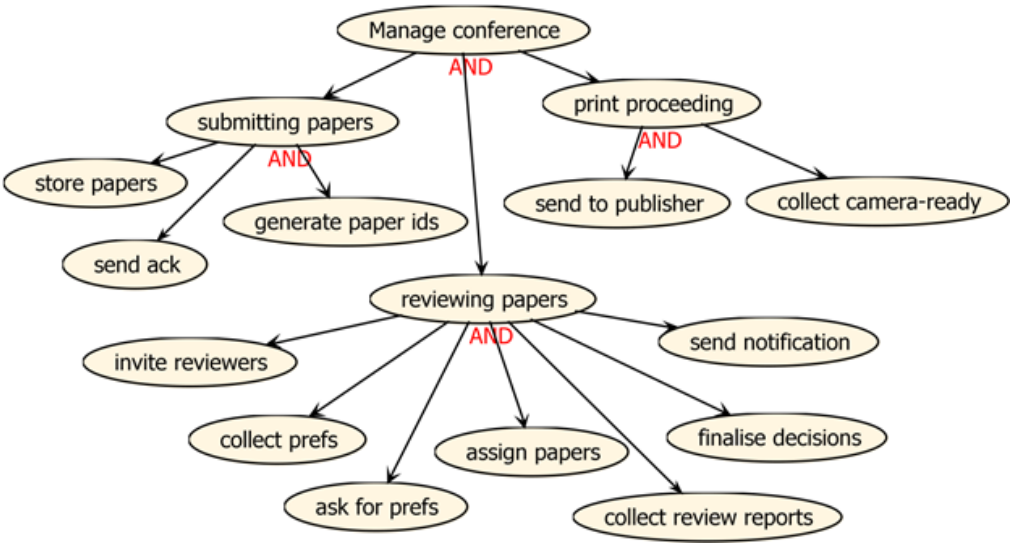
Rule 5 (Activity to Goal): Let $\alpha \in A$ be an activity (either a task or a subprocess) within a process π of a given BPMN model. Also, let g be the goal corresponding to process π:

$\alpha \Rightarrow$ g1 (where g1 is a goal)
g1.name::= α.name
G::= G$\cup${g1}
g.subgoal::= = g.subgoal $\cup${g1}

Each process is directly mapped to a scenario and consequently a high-level goal in Prometheus. The rule to transform a process to a scenario and a goal is described as below.

Rule 6 (Process to Scenario and Goal): Let $\pi \in B$ be a process or sub-process of a given BPMN model:

Figure 5. A goal overview diagram for the CMS (Note: all links are AND)



$\pi \Rightarrow$ s (where s is a scenario)
s.name::= π.name
S::= S$\cup${s}
g (where g is a goal)
g.name::= π.name
G::= G$\cup${g}

Each process corresponds to a scenario in Prometheus which consists of a sequence of steps including goal, action, percept, sub-scenario and others. Using the above rules, we can map elements in a process to steps in a corresponding scenario. For instance, a message event can be mapped into either a percept step or an action step, depending whether it is related to a sending or receiving message. The transformation from BPMN processes to Prometheus scenarios should also preserve the sequential constraint as imposed in the processes, i.e. one activity before the other. The path resulting from split gateways such as XOR or OR can be translated to a variation scenario. There is, however, no equivalent of an AND gateway in Prometheus scenarios. For these cases, the transformation does not care about the other of activities on each of the outgoing path of an AND gateway.

For example, based on the business process models in Figures 3 and 4 we can identify scenarios: "submitting papers", "reviewing papers", and "printing proceeding". Figure 6 shows an analysis overview diagram in Prometheus which is the outcome of applying the above transformation to the BPMN diagrams in Figures 3 and 4. Details of the "reviewing papers" are presented in Figure 7. As can be seen, this scenario reflects the same semantics as described by the BPMN model in Figure 4. For instance, the Start Message Event "reviewer details received" (see Figure 4 is mapped to the "reviewer details" percept and consequently is translated to a percept step in the "reviewing papers" scenario. "Invite reviewers" is a task and is translated to a goal step in the same scenario. Since in the BPMN model the event "reviewer details received" precedes the "invite reviewers", their corresponding scenario steps also follow the same order.

Figure 6. An analysis overview diagram for the CMS

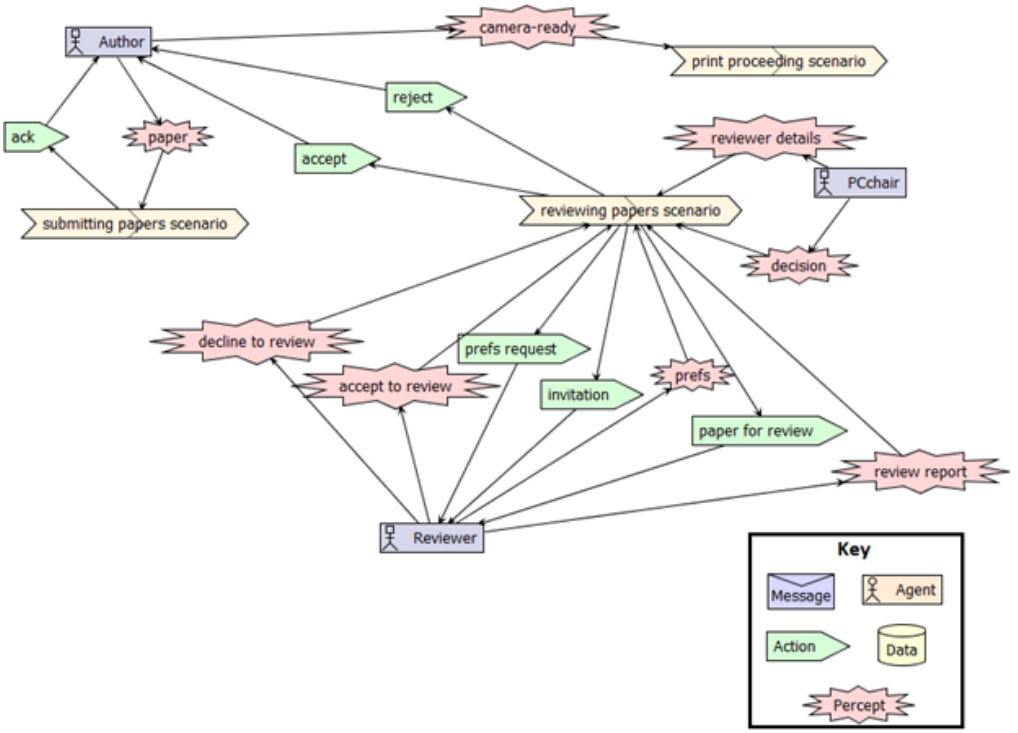


Figure 7 shows a scenario which is the outcome of applying the above transformation to the BPMN diagrams in Figures 3 and 4. Note that the scenario described in Figure 7 is for agreeing to review and then accepting a paper. In a different scenario (e.g. rejecting a paper or declining to review), the relevant action and/or percept would appear.

4. IMPLEMENTATION

We have implemented a model transformation of the conceptual mapping proposed in section 3. We have used the ATLAS Transformation Language[8] (ATL) which includes both a language and a toolkit part of the Model-To-Model (M2M) Eclipse project. An ATL transformation has a set of rules that define how source model elements are matched to create and initialize the elements of the target models. ATL requires Ecore metamodels for both source and target models and thus we have developed an Ecore metamodel for Prometheus based on the metamodel presented in (Dam, 2009). Figure 8 shows the basic overview of the ATL transformation from a BPMN model to a Prometheus model, each of which conforms to its respective metamodel in Ecore. The Ecore BPMN metamodel which is part of the Eclipse SOA Tools Platform[9] is chosen.

The ATL transformation rules are written based on the mapping rules presented in section 3.2. Figure 9 given an example of an ATL rule that transforms a Pool in BPMN to an Actor in Prometheus (rule 1 in Section 3.2). The basic structure of an ATL consists of a header, some helper functions and the actual transformation rules. The header starts with the keyword module then the name of module, given that the names of the source and target models are defined.

Figure 7. The "reviewing papers" scenario of agreeing to review and accepting a paper

Scenario reviewing papers scenario

Name	reviewing papers scenario
Description	
Priority	Not Specified
Actors	
Initiated by	System
Trigger	

Steps

#	Type	Name	Role	Description	Data used	Data produced
1	Percept	reviewer details	Review management	Receive details of reviewer from the PC Chair		
2	Goal	invite reviewers	Review management	Invite reviewers to join the program committee		
3	Action	invitation	Review management			
4	Percept	accept to review	Review management			
5	Goal	ask for prefs	Review management	Ask for reviewer's preferences in terms of topics		
6	Action	prefs request	Review management			
7	Percept	prefs	Review management	Receive reviewers' preferences		
8	Goal	collect Prefs	Review management	collect all reviewers' preferences		
9	Goal	assign papers	Review management	Assign papers to reviewers		
10	Action	paper for review	Review management			
11	Percept	review report	Review management	Receive reviewer's reports		
12	Goal	collect review reports	Review management	Collect all reviewers' reports		
13	Percept	decision	Review management	Receive decision from PC Chairs		
14	Goal	finalise decisions	Review management	Finalize all decisions		
15	Goal	send notification	Review management	Send out notifications		
16	Action	accept	Review management			

Variation

The ATL transformation from BPMN to Prometheus is part of the *BPMN2PDT* plug-in that we have developed for the Eclipsed-based Prometheus Design Tool (PDT). This plug-in supports a seamless transformation from BPMN diagrams to Prometheus models that can be edited using Eclipse-based PDT. More specifically, it allows the designer to import a BPMN model (in the form of an XMI file) and it automatically produces a PDT model. Figure 10

Figure 8. Overview of the ATL transformation from BPMN to Prometheus

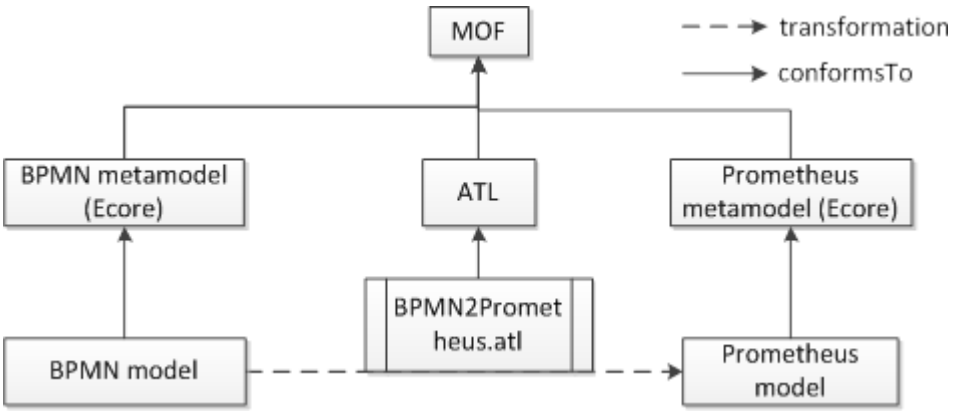


Figure 9. An ATL transformation rule from a pool to an actor

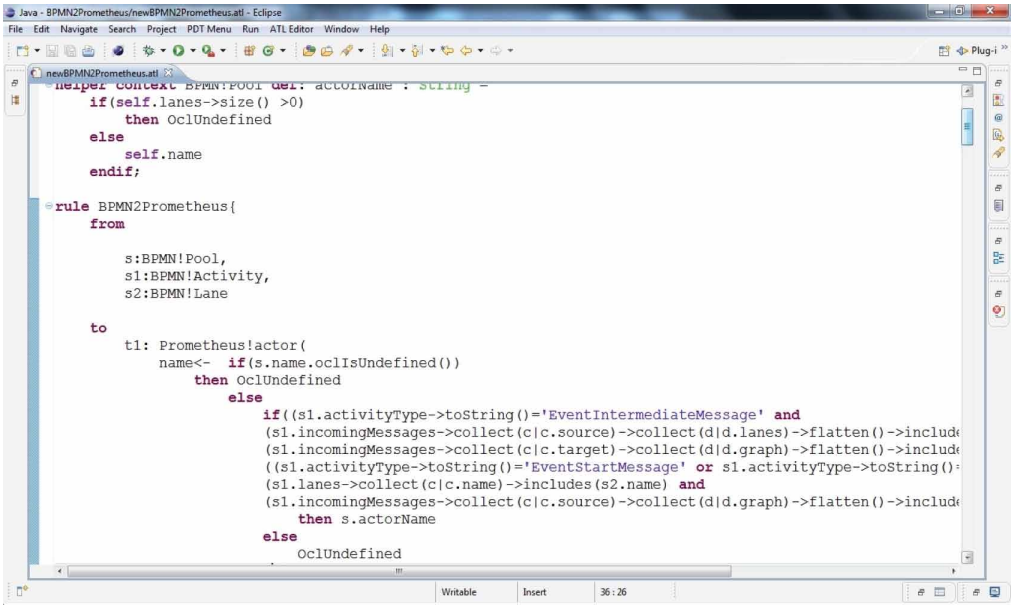
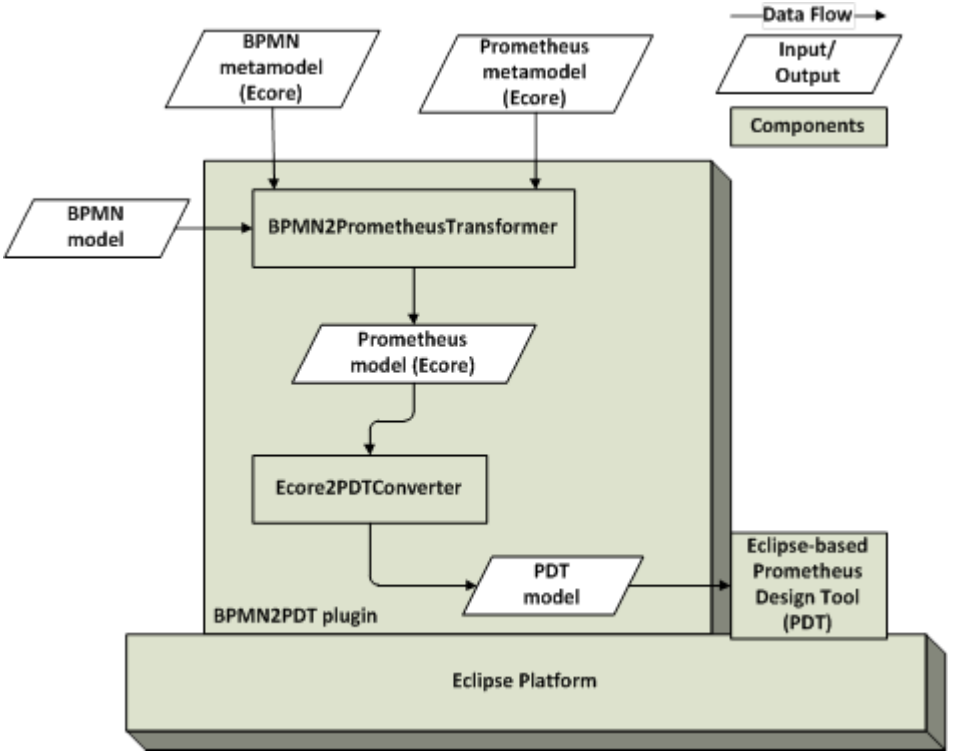


Figure 10. BPMN2PDT plug-in architecture



shows the implementation architecture of the BPMN2PDT plug-in has two components: BPMN2PrometheusTransformer and Ecore2PDTConverter, both of which are based on the Eclipse platform. The former contains an ATL transformation that takes a BPMN model and produces a Prometheus model in Ecore. The latter converts this Ecore-based Prometheus model to a model that can be input into the Eclipse-based PDT. This step is challenging since we need to ensure that the Ecore-based Prometheus model is well-formed and conforms to its metamodel. Future work involves reducing these two steps into one, i.e. converting a BPMN model directly to an Enclipse-based PDT model.

5. RELATED WORK

Having recognized the potential of multi-agent systems as a powerful platform for business process execution, a range of work has recently explored this area. The work in (Endert, Kuster, Hirsch & Albayrak, 2007) proposes an automated mapping from BPMN diagrams to BDI agent concepts. They follow a model transformation approach in which a set of transformation rules are defined to map elements of the source model (BPMN) to elements of the target model (BDI agents). Their model transformation is graph-based and consequently BPMN models need to be converted into an equivalent graph structure before the mapping is executed. Another work from the same group of authors (Endert, Kuster, Hirsch & Albayrak, 2007) uses the same approach to translate BPMN diagrams to a normalized form which is then transformed into a Petri-net. We, however, do not believe that it is feasible to provide an automated mapping directly from business languages (e.g. BPMN models) to an agent execution platform (e.g. BDI). This is due to a significant gap between two different models and levels of abstraction: BPMN notation does not have a formal behavioral semantics while implementation requires a behavioral specification.

The work in (Urzica & Tanase, 2009) takes a different direction in terms of only considering how to map BPMN concepts to agent interaction models, specifically AUML (Bauer, Muller & Odell, 2001). Their mapping covers a few simple BPMN elements such as swimlanes, message events, timer events and gateways. Since their focus is on communicate acts of agents, they do not take into account the internal of agents such as events and plans.

The ideas of using multi-agents as a simulation platform for business processes have also recently gained some attention. For instance, the work in (Pascalau, Giurca & Wagner, 2009) aims to design an agent-based simulation environment for a simple English auction which involves a number of parties (e.g. sellers, auction hosts and bidders). The auction processes are described in BPMN and they propose a mapping from BPMN models to Agent-Oriented Relationship (AOR) (Wagner, 2003) models. Their work is, however, still preliminary since the mapping rules only cover specific patterns in the English auction domain.

There have been some efforts in closing the gap between artifacts produced at the early requirement phase and Prometheus models. For instance, the work in (Cysneiros & Zisman, 2004) proposes a number of guidelines of how to generate Prometheus system specification artifacts from organisational models specified in i*(Yu, 1995). The guidelines are heuristic based and revolve around use case scenarios in Prometheus.

Proposing mapping from business process models to agent artifacts is one direction of work in the application of agents to business process management. Another direction is to extend existing business process models with agent concepts. For instance, the work (Burmeister, Arnold, Copaciu & Rimassa, 2008) proposes to extend BPMN with BDI concepts such as goals and plans. In this context, the process designer needs to model a business process in terms of

goals and plans. Details of those plans are represented using BPMN. This would allow a direct execution of such a goal-oriented business process model through actual BDI agents. However, extending such a standard like BPMN remains a challenging issue.

6. CONCLUSION AND FUTURE WORK

The BPM community is facing challenges in modelling and implementing business processes that are able to adapt themselves to a changing environment. On the other hand, although multi-agent systems potentially provide a powerful and flexible platform for process execution, they still fail to attract a wide industry adoption. Therefore, it is very important to bridge the gap between the BPM community and agent technology. In this paper, we have made contribution to such an effort.

We have argued that providing mapping that automatically transforms business languages directly to an agent platform is not feasible due to the significant gap between the two different models and levels of abstraction. Therefore, we have proposed to translate business languages in a form of BPMN models to artifacts of Prometheus, a prominent agent-oriented methodology. Such artifacts are then used to implement an agent system that realizes those business requirements. We have also developed a plug-in for the Prometheus Design Tool which allows the designer to import a BPMN model and automatically generates Prometheus model in PDT.

There are a number of directions for future work. Firstly, we plan to develop a more complete mapping from BPMN to Prometheus that covers other concepts in the two modelling languages. More specifically, we would like to explore how other types of events (e.g. timer, error, cancel) and decision gateways can be translated to the internal of a plan in Prometheus (e.g. triggering events, context conditions). Secondly, we would like to investigate how the process of designing and implementing an agent-based solution for a business process helps enrich it with alternative behaviours. In our view, the agent design exercise would encourage the designers to consider alternative behaviours and how to handle exceptions (e.g. in terms of subgoals and plans). Therefore, a major topic for our future work involves further exploration of this area.

REFERENCES

Bauer, B., Muller, J. P., & Odell, J. (2001). Agent UML: A formalism for specifying multiagent software systems. *International Journal of Software Engineering and Knowledge Engineering*, *11*(3), 207–230. doi:10.1142/S0218194001000517

Bergenti, F., Gleizes, M.-P., & Zambonelli, F. (Eds.). (2004). *Methodologies and software engineering for agent systems. The agent-oriented software engineering handbook*. Kluwer Publishing. doi:10.1007/b116049

Bernon, C., Gleizes, M.-P., Peyruqueou, S., & Picard, G. (2002), ADELFE: a methodology for adaptive multi-agent systems engineering. In Proceedings of the 3rd international conference on Engineering societies in the agents world III (pp 156-169) Paolo Petta, Robert Tolksdorf, and Franco Zambonelli (Eds.). Springer-Verlag, Berlin, Heidelberg

Bordini, R. H., Dastani, M., Dix, J., & Fallah-Seghrouchni, A. E. (Eds.). (2005). *Multi-agent programming: Languages, platforms and applications* (Vol. 15). Springer. doi:10.1007/0-387-26350-0_1

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 203–236. doi:10.1023/B:AGNT.0000018806.20944.ef

Burmeister, B., Arnold, M., Copaciu, F., & Rimassa, G. (2008, May). BDI-Agents for agile goal-oriented business processes. In Padgham, Parkes, M¨uller, & Parsons (Eds.), *Proceedings of the 7th international conference on autonomous agents and multiagent systems (AAMAS 2008)* (pp. 37–44). Estoril, Portugal.

Cysneiros, G., & Zisman, A. (2004). Refining Prometheus methodology with i*. *In Proceedings of the third international workshop on agent-oriented methodologies (OOPSLA 2004)*. Vancouver, Canada.

Dam, H. K., & Ghose, A. (2010). Agent-based development for business processes (early innovation track). *In Proceedings of the 13th international conference on principles and practice of multi-agent systems (PRIMA 2010)*. Kolkata, India.

Dam, K. H. (2009). *Supporting software evolution in agent systems*. PhD Thesis, RMIT University, School of Computer Science and IT.

DeLoach, S. (2002). Modeling organizational rules in the multi-agent systems engineering methodology. *In AI '02: Proceedings of the 15th conference of the Canadian society for computational studies of intelligence on advances in artificial intelligence* (pp. 1–15). London, UK: Springer-Verlag. doi:10.1007/3-540-47922-8_1

DeLoach, S. A. (2005). Engineering organization-based multiagent systems. In A. F. Garcia, R. Choren, C. J. P. de Lucena, P. Giorgini, T. Holvoet, & A. B. Romanovsky (Eds.), *Software engineering for multi-agent systems IV, research issues and practical applications* (Vol. 3914, pp. 109–125). Springer. doi:10.1007/11738817_7

Endert, H., Hirsch, B., Kuster, T., & Albayrak, S. (2007). Towards a mapping from BPMN to agents. In Aamas'07/socase'07: *Proceedings of the 2007 AAMAS international workshop and socase 2007 conference on service-oriented computing* (pp. 92–106). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-72619-7_7

Endert, H., Kuster, T., Hirsch, B., & Albayrak, S. (2007). Mapping BPMN to Agents: An Analysis. In *First international workshop on agents, web-services, and ontologies integrated methodologies* (awesome'07) (pp. 164– 180).

Fingar, P. (2008). *Business process management: The next generation* (Research Report). BPTrends.

Ghose, A. (2009). Industry traction for mas technology: Would a rose by any other name smell as sweet. [Inderscience]. *International Journal of Agent-Oriented Software Engineering*, *3*(4), 397–401. doi:10.1504/IJAOSE.2009.025318

Henderson-Sellers, B., & Giorgini, P. (Eds.). (2005). *Agent-oriented methodologies*. Idea Group Publishing. doi:10.4018/978-1-59140-581-8

Hill, J. B., Cantara, M., Deitert, E., & Kerremans, M. (2007). *Magic quadrant for business process management suites (Tech. Rep.)*. Gartner Research.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, 44 (4), 35–41. doi: http://doi.acm .org/10.1145/367211.367250

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, *1*(1), 7–38. doi:10.1023/A:1010090405266

Jouault, F., Allilaire, F., B'ezivin, J., & Kurtev, I. (2008, June). Atl: A model transformation tool. [Elsevier.]. *Science of Computer Programming*, *72*(1-2), 31–39. doi:10.1016/j.scico.2007.08.002

Luck, M., McBurney, P., Shehory, O., & Willmott, S. (2005). *Agent technology: Computing as interaction (a roadmap for agent based computing)*. AgentLink. Retrieved from http://www.agentlink.org/roadmap/al3rm.pdf

Object Management Group. (2009). *Business Process Model and Notation (BPMN)*, v1.2. http://www.omg.org/spec/BPMN/1.2

Odell, J. Objects and agents compared. *Journal of Object Technology*, 1(1):41_53, 2002

Padgham, L., & Winikoff, M. (2004). *Developing intelligent agent systems: A practical guide*. Chichester: John Wiley & Sons. doi:10.1002/0470861223

Pascalau, E., Giurca, A., & Wagner, G. (2009). Validating auction business processes using agent-based simulations. In Business process, services computing and intelligent service management (Vol. 147, p. 95-109).

Rao, A. S., & Georgeff, M. P. (1995). BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*. San Francisco.

Rougemaille, S., Migeon, F., Maurel, C., & Gleizes, M.-P. (2008). Model Driven Engineering for Designing Adaptive Multi-Agents Systems. In A. Artikis, G. P. O'Hare, K. Stathis, & G. Vouros (Eds.), *Engineering Societies in the Agents World VIII*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-87654-0_18

Smith, H., & Fingar, P. (2003). *Business process management: The third wave*. Meghan-Kiffer Press.

Urzica, A., & Tanase, C. (2009). Mapping BPMN to AUML: Towards and Automatic Process. In *Proceedings of the 17th international conference of control systems and computer science*, masts 2009 workshop (pp. 539– 547).

Wagner, G. (2003). The agent-object-relationship metamodel: Towards a unified view of state and behavior. *Information Systems*, *28*(5), 475–504. doi:10.1016/S0306-4379(02)00027-3

Weyns, D., Parunak, H. V. D., & Shehory, O. (Eds.). (2009). International journal of agent-oriented software engineering (ijaose) - special issue on the future of software engineering and multi-agent systems (Vol. 3) (No. 4). Inderscience Publishers.

Wooldridge, M. (2002). An introduction to multiagent systems. John Wiley & Sons (Chichester, England). (ISBN 0 47149691X)

Yu, E. (1995). *Modelling strategic relationships for process reengineering* (PhD Thesis). University of Toronto, Department of Computer Science.

Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multi-agent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12 (3), 317–370. doi: http://doi.acm.org/ 10.1145/958961.958963

ENDNOTES

1 http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0
2 http://www.bpmn.org
3 http://www.cs.rmit.edu.au/agents/pdt
4 http://www.bpmn.org/BPMN_Supporters.htm
5 There is a paper assignment and bidding process that we do not show here for brevity.
6 There is also an "other" step type which can be used to represent miscellaneous things such as waiting for a response.
7 This notation is based on the Object Constraint Language and uses the metamodel in Figure 2.
8 http://www.eclipse.org/atl
9 http://www.eclipse.org/stp

Hoa Khanh Dam is a Senior Lecturer at the School of Computing and Information Technology, University of Wollongong, Australia. He holds PhD and Master degrees in Computer Science from RMIT University, and Bachelor of Computer Science degree from the University of Melbourne in Australia. His work has won multiple Best Paper Awards (at WICSA, APCCM, and ASWEC) and ACM SIGSOFT Distinguished Paper Award (at MSR). His research has been published in the top venues in software engineering (ICSE, ASE, ICSM, ER, JSS), AI/intelligent agents (AAMAS, JAAMAS), and service-oriented computing (ICSOC, SCC, BPM). He served as Program Co-Chair for the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA) in 2014 and is on the Senior Program Committee (PC) panel for this conference series. He has won the Best Senior Program Committee Member Award at PRIMA 2013. Other major international conferences and journals that he has been involved with include AAMAS (PC), ICSOC 2015 (Publication Chair and PC), ASWEC and EDOC 2015 (Publicity Chair), and Journal of Systems and Software and the Science of Computer Programming journal (expert reviewer). Prior to his academic career, he spent a number of years in the industry at various positions, including technical architect, project manager and software engineer.

Aditya Ghose is Professor of Computer Science at the University of Wollongong. He leads a team conducting research into knowledge representation, agent systems, services, business process management, software engineering and optimization and draws inspiration from the cross-fertilization of ideas from this spread of research areas. He works closely with some of the leading global IT firms. Ghose is President of the Service Science Society of Australia and served as Vice-President of CORE (2010–2014), Australia's apex body for computing academics. He holds PhD and MSc degrees in Computing Science from the University of Alberta, Canada (he also spent parts of his PhD candidature at the Beckman Institute, University of Illinois at Urbana Champaign and the University of Tokyo) and a Bachelor of Engineering degree in Computer Science and Engineering from Jadavpur University, Kolkata, India.

Muhammad Qasim holds a Master degree in Computer Science from the University of Wollongong. His interests include Software Engineering/Project Management and Artificial Intelligence/Multimedia. He is currently a Case Manager for Samsung Electronics Australia, Service Division HHP, contracting through WDS, a Xerox Company.