Engineering Applications of Artificial Intelligence 24 (2011) 895–910

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Agent-based virtual organization architecture

S. Rodriguez [a], V. Julián [b,*], J. Bajo [c], C. Carrascosa [b], V. Botti [b], J.M. Corchado [a]

[a] Departamento Informática y Automática, Universidad de Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain
[b] Departamento Sistemas Informáticos y Computación, Universidad Politécnica de Valenciam, 46022 Valencia, Spain
[c] Universidad Pontificia de Salamanca, C/Compañía 5, 37002 Salamanca, Spain

ARTICLE INFO

Article history:
Received 25 January 2010
Received in revised form
22 October 2010
Accepted 7 February 2011
Available online 26 March 2011

Keywords:
Multiagent systems
Virtual organizations
Dynamic architectures
Adaptive environments
Service Oriented Architectures

ABSTRACT

The purpose of this paper is to present the applicability of THOMAS, an architecture specially designed to model agent-based virtual organizations, in the development of a multiagent system for managing and planning routes for clients in a mall. In order to build virtual organizations, THOMAS offers mechanisms to take into account their structure, behaviour, dynamic, norms and environment. Moreover, one of the primary characteristics of the THOMAS architecture is the use of agents with reasoning and planning capabilities. These agents can perform a dynamic reorganization when they detect changes in the environment. The proposed architecture is composed of a set of related modules that are appropriate for developing systems in highly volatile environments similar to the one presented in this study. This paper presents THOMAS as well as the results obtained after having applied the system to a case study.

1. Introduction

Computation as interaction paradigm can be considered the most promising technological evolution in the areas of Computer Science and Communication in the last few years (Luck et al., 2005). According to this new paradigm, computation is something that occurs by means of and through communication among computational entities. In this approximation, large systems can be seen, in a natural way, in terms of the services offered or demanded, and consequently, in terms of the entities (probably agents) providing or consuming services (Luck and McBurney, 2008). It is necessary to remark that these entities may not have been designed in a joint way or even by the same development team. Entities may enter or leave different organizations in different moments and due to different reasons. Moreover, they may form coalitions or organizations between themselves to attain their current goals. Obviously, the development of these types of systems is complex and, therefore, it is necessary to analyse in detail the intrinsic characteristics of these typical application environments. Recent tendencies have conducted to the use of virtual organizations (VOs), which can be considered as a set of individuals and institutions that need to coordinate resources and services across institutional boundaries (Foster et al., 2001; Boella et al., 2005). Therefore, a VO is an open system formed by the grouping and collaboration of heterogeneous entities and there is a separation between form and function that requires defining how a behaviour will take place. Multiagent systems (MAS) technology, which allows forming dynamic agent organizations, is particularly well suited as a support for the development of these open systems. An open MAS organization modelling makes it possible to describe structural composition (i.e. roles, agent groups, interaction patterns, role relationships) and functional behaviour (i.e. agent tasks, plans or services), and it can incorporate normative regulations for controlling agent behaviour, dynamic entry/exit of components and dynamic formation of agent groups. The development of open MAS is still a recent field of the multiagent system paradigm, in this sense, it is necessary to investigate new methods to model open agent-based virtual organizations, as well as innovative techniques to provide advanced organizational abilities to virtual organizations.

The main goal of this study is to propose a new open multiagent system architecture consisting of a related set of modules that are suitable for the development of virtual organizations in open dynamic environments. This study focuses on the definition of a high-level design of a related abstract architecture to build virtual organizations. In this design, all components required to cover all characteristics and needs of systems of this kind are determined. This new architecture has been named THOMAS (MeTHods, Techniques and Tools for Open Multiagent Systems) and provides a novel perspective to model virtual organizations. Based on this proposal, an agent execution framework has been implemented, which allows to produce software wholly independent of any internal agent platform, and as such, that is fully addressed for open systems.

* Corresponding author: Tel.: +34 96 387 70 07 +73583; fax: +34 96 387 73 59.
E-mail address: vinglada@dsic.upv.es (V. Julián).

0952-1976/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2011.02.003

In order to evaluate the THOMAS architecture we have chosen a case study consisting in the implementation of an open virtual organization, focused in designing and building a system for guiding and advising users in a shopping mall. The case study makes use of the THOMAS architecture to obtain an open virtual organization that can be viewed as a temporary alliance between various partners (clients of the mall and shoppers) and services (offered basically by the shops) to support certain activity or a set of activities in an open market (the mall). In an environment as dynamic and evolving as this, it becomes increasingly necessary to establish risk management security policies that can control conditions, performance and conflictive situations, thus contributing to improve the quality of services for the clients and enhancing their bargain. We have developed a planning service that has been integrated within the developed MAS (Bajo et al., 2009) and modelled through THOMAS. It involves a self-adapting mechanism that facilitates the automatic assignment of tasks within the shopping mall.

The rest of the paper is structured as follows: Section 2 focuses on the analysis of related work; Section 3 shows the THOMAS approach describing its main components and the execution framework; Section 4 describes a case study which makes use of this approach and, finally, results and conclusions are shown in Sections 5 and 6.

2. Related work

Social aspects of multiagent system organizations are increasingly important for specifying agent interactions in open and dynamic worlds. Research into open MAS organizations has ranged from basic organizational concepts, such as groups, communities, roles, functions (Jennings and Wooldridge, 2002; Zambonelli and Parunak, 2002; Ferber and Gutknecht, 1998; Odell et al., 2005), organizational modelling (Horling and Lesser, 2005; Ferber and Gutknecht, 1998; Dignum et al., 2005), Human Organization Theory (Argente et al., 2006), structural topologies (Horling and Lesser, 2004; Argente et al., 2007), to normative research, including internal representation of norms (Lopez et al., 2006), deontic logics (Dignum and Dignum, 2007) and institutional approaches (Esteva et al., 2001; Arcos et al., 2005). But there is no such agent platform, which covers all the needed execution management of an agent organization. Therefore, new approaches are needed in order to support the evolution of those infrastructures and to enable their growing and updating through execution time, due to the inherent features of those open environments. In general, some standards and platforms for agent interoperability, which take into account those requirements, need to be defined.

One of the key problems for open MAS organizations development is the vision of agents and organizations as service-provider entities. This vision needs a main effort in the integration of agents and web services directed at masking services for redirection, aggregation, integration or administration purposes (Greenwood and Calisti, 2004). Although the evolution of MAS and web services has been completely different, both technologies have pursued common goals such as providing dynamic and open architectures. In order to solve their differences and to give interoperation and integration possibilities, there are some approaches that focus on finding the best way to facilitate this interoperation: WSIG, WD2JADE and WSAI. The Web Service Integration Gateway Service (WSIG) architecture (Greenwood and Calisti, 2004) allows agents and web services to register and publish their service descriptions to potential consumers. It contains an intermediary entity that provides translations between FIPA agents and web services in a transparent and bidirectional way. WS2JADE (Nguyen and Kowalczyk, 2005) integrates Web Service and agents, proposing a Web service to FIPA Agent Service gateway. This architecture is based on an interconnecting layer (with special agents capable of communicating with web services) and a management layer, responsible of creating and managing the dynamic interconnection between entities. Finally, the Web Services Agent Integration (WSAI[1]) allowed web service clients to use agent services, being also possible to publish an agent as a web service.

Regarding virtual organization modelling, recently, researchers have carried out several studies that offer new procedures and methodologies to enable designing open MAS, focused on the organizational aspects of the agent society. Some examples of these approaches are Gaia (Zambonelli et al., 2003), AGR (Ferber et al., 2004), MOISE (Hubner et al., 2002), OperA (Dignum, 2003) (based on ISLANDER (Esteva et al., 2001) framework), Tropos (Bresciani et al., 2004), PASSI (Cossentino, 2005), SODA (Molesini et al., 2006), MenSA (Ali et al., 2008), O-MASE (DeLoach, 2009), INGENIAS (Pavon et al., 2005) and VOM (Criado et al., 2009). Many of the recent works not only focus on the employment of organizational structures during the design process, but also on the regulation of open MAS (standing out E-Institutions (Noriega and Sierra, 2002; Aldewereld et al., 2006), Moise (Hubner et al., 2002), Moise^{Inst} (Gâteau et al., 2006)). Albeit some of the platforms tackle with organizational concepts as design patterns, they cannot be directly applied to the development of open multiagent systems where organizational structures can emerge and change dynamically at runtime. The abstractions and tools currently available are still not enough for many kinds of open MAS that deal with real world problems. Most of these works are, in a certain way, incomplete as they do not include all the phases and requirements for the entire development of systems of this kind (Argente et al., 2008). Moreover, none of them take into account the problems associated with the last development phases, in which the software engineer must translate the specific properties of the VOs, specified in the analysis and design models, to executable code for agent platforms.

Finally, the main problem for obtaining executable code for VOs is the lack of agent platforms supporting open agent system functionalities. The main function of an agent platform is to offer an execution framework for all agents that belong to it. In the last years, many research and commercial approaches have appeared trying to offer this execution framework. Examples of proposed agent platforms are JADE,[2] FIPA-OS,[3] Grasshopper (Baumer et al., 2000), Jack (Howden et al., 2001), ZEUS (Nwana et al., 1999), MadKit (Gutknecht and Ferber, 1997), EIDE (Esteva et al., 2001), RICA-J (Serrano et al., 2004), S-Moise+ (Hubner et al., 2006), Jack Teams (which is an extension to JACK) (Agent-Oriented-Software, 2004), SIMBA (Soler et al., 2002) and SPADE (Escriva et al., 2006). This paper does not try to analyse the different proposals, a detailed comparison of relevant agent platforms can be found in Argente et al. (2004). Taking only organizational aspects into account, although some of the platforms deal with organizational concepts as design patterns, they cannot be directly applied to the development of open MAS where organizational structures can emerge and change dynamically at runtime.

According to the current situation, any application implemented as an open multiagent virtual organization demands an execution environment with support for:

- *Organization representation*: in virtual organizations agents may need an explicit representation of the organization that has been defined. Thus, an agent is able to reason about it in order to initiate cooperation with other agents. Previous

[1] http://wsai.sourceforge.net/.
[2] http://jade.tilab.com/
[3] http:/fipa-os.sourceforge.net/.

works, which incorporate this idea, are S-Moise+ and Ameli (Esteva et al., 2004). Nevertheless, analysed organization approaches are static, we express the necessity of adaptive mechanisms for creating organizational structures that allow optimizing the coordination in open systems taking into account the heterogeneity of agents and services.

- *Control mechanisms*: the platform should have control mechanisms that ensure the satisfaction of the organizational constraints. It is necessary to define dynamic regulatory mechanisms that guarantee a globally efficient coordination in open systems taking into account the impossibility of controlling (the majority of) the agents and services directly.
- *Description of the organizations*: the organization should have an available description in a standard language. This allows external and internal agents to get specific information about the organization at runtime. This feature is not only useful in open systems but also when considering a reorganization process. A good example of organization specification can be found in the S-Moise+ platform.
- *AMS and DF*[4] *extension*: the AMS and the DF offered by traditional MAS platforms should be improved. The AMS should have information on the existing organizations and their members. The DF should publish the services offered by agents individually and the services offered by organizations. It should not only have the name of the service offered, but also its description to allow open systems. Finally, it may have composition meta-services capable of on-the-fly orchestration of services with regard to a given query.
- *Communication layer*: the kind of communication layer used in communicative acts is a very important feature. Some of them, such as FIPA-ACL (used by AMELI) and KQML (used by S-Moise+), are more suitable for open systems than TCP/IP, CORBA and RMI (Ferber et al., 2004).
- *Monitoring*: the platform should offer a mechanism for monitoring the states of agents and organizations, which helps in the validation and verification processes but also could be used by organizations and agents in the system to perceive their environment without having to actively notify each change to the rest of the entities, which could be interested in what they do.
- *Modelling concepts support*: the platform and the programming language should cover all of the concepts related to the virtual organization. For example, which types of topologies are defined within the platform, which kind of norms is modelled, etc. Not all platforms have complete modelling concept support. For example, AMELI is focused on the management of rules and norms but does not support the definition of complex topologies. Jack Teams platform allows the creation of composed "Teams" but it does not take into account other topologies.
- *Organizational API*: the platform should offer an API that makes it possible to create, destroy and modify organizations; consult and modify the organization description; add, query and delete agents of an organization; send messages to a whole organization, etc. (Argente et al., 2007; Criado et al., 2007).

The present paper represents a step forward on these requirements; the proposed architecture includes the integration of organizational and individual perspectives, and the dynamic adaptation of models to organizational and environmental changes. Our architecture offers dynamical services for allowing agents to take/leave roles at runtime or offer/demand services, accordingly to the established organizational norms. Moreover, all this functionality has been designed following a Service Oriented approach. THOMAS gives also support for the coordination among agents, which belong to different platforms. In addition, all services are registered and published by the architecture in order to allow agents to discover them and to know how to make use of them. THOMAS also offers structural services for allowing the organizational structure to be modified at runtime, supporting the organizational structure evolution and facilitating its growth and update through execution time.

[4] AMS (Agent Management System) and DF (Directory Facilitator) are the management components of a typical FIPA—compliant platform (www.fipa.org).

3. The THOMAS approach

This section describes the main characteristics of THOMAS architecture, paying special attention to the abstract architecture and the execution framework.

3.1. Abstract architecture

THOMAS (Giret et al., 2009; Carrascosa et al., 2009) is an architecture for open MAS that tries to update the FIPA abstract architecture applying ideas from the Service Oriented Architecture (SOA) field along with allowing dealing with agents grouped in virtual organizations (VO).

Agents' dealing with services is a concept present in the FIPA abstract architecture since its first instalments. But, it is a concept not enough developed. The information and mechanisms used by the traditional FIPA Directory Facilitator (DF) are not good enough to deal with open systems and system dynamics, having several limitations as their very basic service descriptions (name, type, protocol, ontology, language, ownership and properties) and allowed functionalities (register, deregister, modify and search) not considering any one of their virtual organizations. Another DF limitation is regarding service discovery, as its search algorithm does not use any semantic information and it does not consider service compositions. As Service Oriented Architectures have dealt these aspects in depth, THOMAS architecture proposes to incorporate this SOA vision to the way services are dealt in an agent platform.

On the other hand, FIPA abstract architecture does not permit to deal with virtual organizations, another key concept to work with open systems by modelling the organization aspects of the agent society. Therefore, there is no way to manage the creation/destruction of virtual organizations, the way an agent may become part of an existing virtual organization, and how the behaviour of agents within a virtual organization is controlled.

In this way, the THOMAS abstract architecture updates the FIPA approach by presenting all the architecture functionalities specified as SOA services, updating the Directory Facilitator (DF), that is, FIPA module in charge of managing services, to use services specified according to SOA standards. This module, called now Service Facilitator (SF), allows not only registering offering services, but also demanding. Moreover, these services can be related not only to agents, but also to organizational units. The SF provides three different kinds of services:

- *Registration*: to manage the registering, modifying and removing of services in the SF (*RegisterProfile*, *RegisterProcess*, *ModifyProfile*, *ModifyProcess*).
- *Affordability*: to link providers with their services (*AddProvider*, *RemoveProvider*).
- *Discovery*: to allow to search and compose services on demand (*SearchService*, *GetProfile*, *GetProcess*).

Last but not least, the THOMAS abstract architecture includes a new module called Organization Management Service (OMS) in charge of managing virtual organizations lifecycle (as AMS deals with agents one) along with some norm management.

More specifically, it controls how virtual organizations (called here Organizational Units) are created and the way entities participate inside such organizations (which entities, how they are related to each other and which roles they play). To do this, the OMS provides three different kinds of services:

- *Structural services*: to define/change the structure and norms of the organizational units (*RegisterRole*, *DeregisterRole*, *RegisterNorm*, *DeregisterNorm*, *RegisterUnit*, *DeregisterUnit*).
- *Informative services*: to get information of the current state of organizational units (*InformUnitRoles*, *InformAgentRoles*, *InformMembers*, *InformQuantity*, *InformUnit*, *InformRoleProfiles*, *InformRoleNorms*).
- *Dynamic services*: to allow the dynamic evolution of organizational units (*AcquireRole*, *LeaveRole*, *Expulse*).

Table 1 summarizes the different services offered in THOMAS. Moreover, Fig. 1 shows the final composition of the THOMAS abstract architecture that, along with the above-mentioned SF and OMS modules, presents also a Platform Kernel (PK) module that includes functionalities for managing agents lifecycle (module AMS of FIPA) and communications (module Network Layer of FIPA).

3.2. Execution framework

As has been detailed in the previous section, the THOMAS abstract architecture presents two new modules (OMS and SF) above a well-known module, as it is the PK. This last module controls not only the communication but also the life cycle of internal agents. It is a classical standard FIPA abstract architecture PK module. The novelty here is in those new modules, OMS and SF, and the way they are addressed to manage open systems. Moreover, they can be seen as a whole, as a platform-independent set of services in a framework to manage virtual organizations for open systems. That is called, *THOMAS Framework*.

THOMAS Framework allows any agent (from any kind of agent platform) to create its own virtual organization with the structure and norms to manage it, along with the offering and demanding services required. The framework will manage the structure, norms and life cycle of the virtual organization, along with the visibility of offering and demanding services and the fulfilment of the conditions to use them.

To be as accessible as possible, all the THOMAS Framework functionalities are offered in two different ways: as a set of

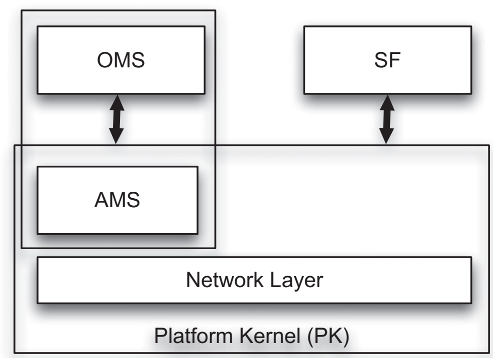

Fig. 1. THOMAS abstract architecture.

Table 1
Summary of THOMAS offered services.

SF services Type	SF service	Description
Registration	Register Profile	Creates a new service description
	RegisterProcess	Creates a particular implementation (process) for a service
	ModifyProfile	Modifies an existing service profile
	ModifyProcess	Modifies an existing service process
	DeregisterProfile	Removes a service description
	DeregisterProcess	Removes a service process
Affordability	AddProvider	Adds a new provider to an existing service process
	RemoveProvider	Removes a provider from a service process
Discovery	SearchService	Searches for a service that satisfies the user requirements
	GetProfile	Gets the description (profile) of a specific service
	GetProcess	Gets the implementation (process) of a specific service
OMS services		
Type	**OMS service**	**Description**
Structural	RegisterRole	Creates a new role within a unit
	RegisterNorm	Includes a new norm within a unit
	RegisterUnit	Creates a new unit within a specific organization
	DeregisterRole	Removes a specific role description from a unit
	DeregisterNorm	Removes a specific norm description
	DeregisterUnit	Removes a unit from an organization
Information	InformAgentRole	Indicates roles adopted by an agent
	InformMembers	Indicates entities that are members of a specific unit
	QuantityMembers	Provides the number of current members of a specific unit
	InformUnit	Provides unit description
	InformUnitRoles	Indicates which roles are the ones defined within a specific unit
	InformRoleProfiles	Indicates all profiles associated to a specific role
	Infor, RoleNorms	Provides all norms addressed to a specific role
Dynamic	AcquireRole	Requests the adoption of a specific role within a unit
	LeaveRole	Requests to leave a role
	Expulse	Forces an agent to leave a specific role

independent services available via WSDL descriptions, as common web services, that can be allocated in any host or set of hosts; or via ACL messages that allow a higher level communication or even negotiation processes.

From a virtual organization point-of-view, all agents included in the framework must belong to an organization. Thus, the THOMAS framework provides obtaining virtual organizations in which any entity is automatically included, as well as a general role that allows the entity to ask for service descriptions so as to fulfil its needs. Throughout the service description, the client can be informed of the roles required to ask for any specific service or the roles needed to provide a specific service inside the organization.

As commented above, the open service-oriented virtual organizations that THOMAS is addressed to support, are very complex MAS, and it is necessary to make available for the designer as specialized tools as possible. A THOMAS-based system can be designed through a related set of methodological guidelines to develop this kind of systems, called GORMAS (Argente et al., 2009). Moreover, there exists a toolkit named EMFGORMAS (Garcia et al., 2009) aimed at linking the GORMAS specification with the THOMAS architecture. EMFGORMAS is an ECLIPSE add-on that enables the specification of the system by means of GORMAS models as shown in Fig. 2. Applying this toolkit a prototype of the designed system is obtained.

Nowadays, a first version of a framework (v0.1) based on THOMAS is available. More details about this implementation of the framework can be found at http://www.dsic.upv.es/users/ia/sma/tools/Thomas.

Next section describes in detail the case study for a shopping mall scenario.

4. Case study

A multiagent system based on THOMAS was designed to facilitate the interconnection between users (clients in the mall) and information on shops, on sales and on leisure activities (entertainment, events, attractions, etc.), delimiting services that it can be requested or offered. Through the case study it is intended to show how an organization-based multiagent system fits perfectly into dynamic environments, in which we have different types of agents and different types of services that can enter and exit the system dynamically. A shopping mall is a dynamic environment, in which shops change, promotions appear and disappear continuously, etc. The proposed system helps users to identify a shopping or leisure plan as well as to identify other users within a given shopping mall. A simple example to show what sort of services the system can offer or request is the route recommender. The mall's main entrance has been taken as the origin of coordinates. Different positions (user, shops, leisure areas) are represented by means of coordinates on a plane. Taking into account the user's interests, places to visit are selected and the routes that include those points are traced, and the optimum route is proposed. This is done bearing in mind the time available, and the shopping and leisure activities schedule. To this end, a unit of our system, *RecommendationUnit*, is comprised of services that request recommendations or suggestions based on user preferences and certain restrictions (time, money, etc.). It also includes planning and replanning the route that the user will follow based on the suggested recommendations, and determines the validity and value of the proposed routes. In this case, the service "*Planning*" can be provided, through the organization developed with THOMAS, for different types of agents, including agents that become part of the system after the design. And each of them can make planning differently.

The system controls which services each agent must provide. The internal functionality of these services is the responsibility of provider agents. However, the system imposes some restrictions regarding service profiles, service requesting orders and service results. In the system, agents can enter and exit the system dynamically performing a specific function within the organization. In this open system, the agents can calculate the guiding routes according to the client profile. The system is capable of replanning the routes in accordance with time and money available, and client preferences.

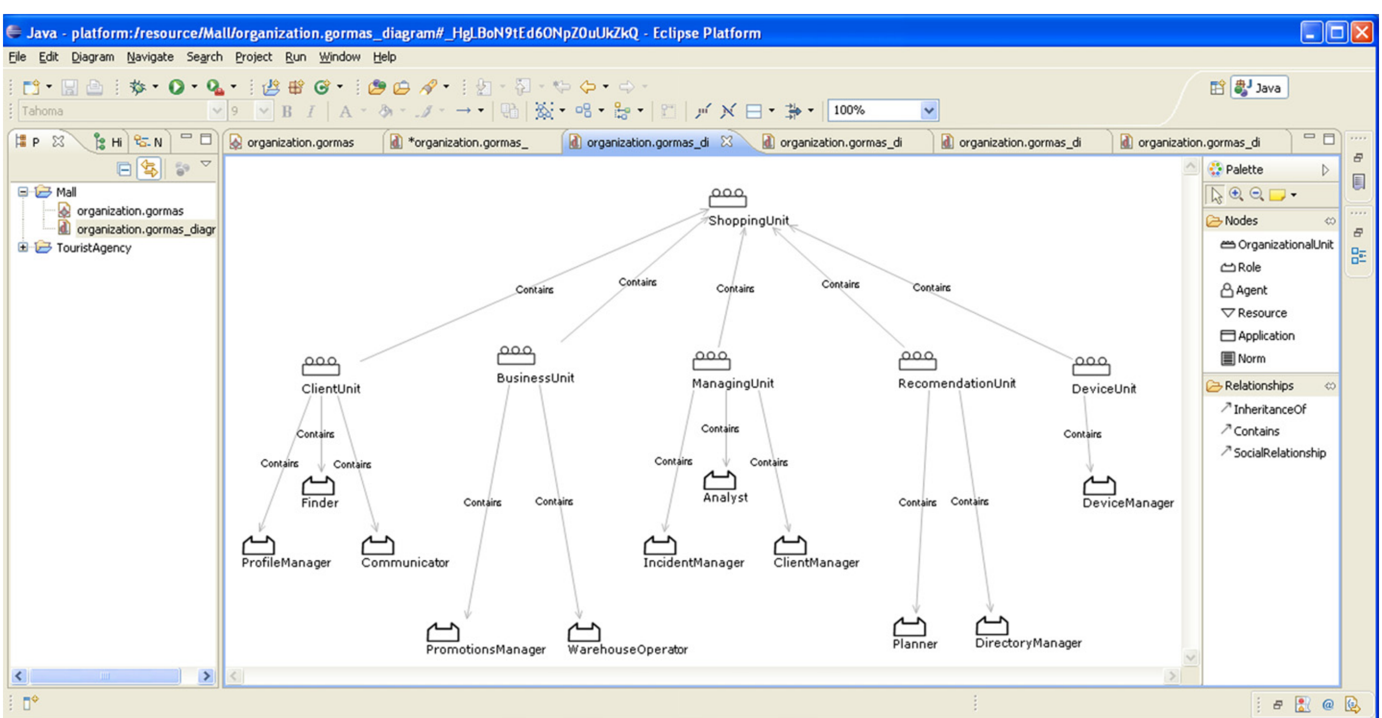

Fig. 2. EMFGORMAS toolkit.

A description of the elements of the shopping mall organization is detailed in Section 4.1. Then, an analysis and design of the selected scenario through THOMAS is explained in Section 4.2. To this end, the GORMAS methodology (Argente et al., 2008) was used. The phases proposed by GORMAS cover the analysis and design of the organizational structure, as well as the design of the dynamics of the organization. Thanks to GORMAS phases, it is specified what services the organization offers, its internal structure and what rules govern their behaviour. Phases A and B cover an analysis of the motivation of the organization and the services and tasks that can be performed. In phases C, D and E, it was analysed and selected the best alternative organization. This alternative specifies the roles, interactions and norms related to the structure of the organization. In phase F, the functionality is offered as an open system, which includes the services that should be publicized, and acquisition policies and release roles. Phases G and H are devoted to the measurement, control and reward systems.

Section 4.2 shows the roles and structural organization obtained at stages C, D, E and F of the methodology for the case study. First, the roles, units and relationships between them were identified; focusing on the structure that best should be applied in the system (phases C and D). Section 4.2.1 shows how services were reviewed, identifying all the roles that were responsible for providing and detailed in the interaction model, which identifies the services associated with each unit (phase E). Finally, thanks to the norms of the system, it was determined which services to those required by the organization, may be provided not only by agents of the system itself, but also by external agents. Norms will be implemented internally as part of system actions or work of the agents (phase F). This will be shown in Section 4.2.2. Section 4.2.3 shows an example of how all these elements are handled in THOMAS.

4.1. Elements of the shopping mall organization

The selected scenario is a guiding system for users of a shopping mall that helps them to identify bargains, offers, leisure activities, etc. An open wireless system was developed and is capable of incorporating agents that can provide useful guidance and advice services to the users not only in a shopping centre, but also in any other similar environment such as the labour market, educational system, medical care, etc. Users (clients in the mall) are able to gain access to information on shops and sales and on leisure activities (entertainment, events, attractions, etc.) by using their mobile phone or PDA. Mechanisms for route planning when a user wants to spend time in the mall are also available. Moreover, the system provides a tool for advertising personalized offers (shop owners will be able to advertise their offers to the shopping mall users), and a communication system between management, the commercial sector or shoppers.

Fig. 3 shows the multiagent based shopping mall scenario. Clients use their personal agents to consult the catalogue of the shops in the mall, to receive advice or personalized promotions, to request guidance suggestions and to locate other clients (RFID technology is required) (Vrba et al., 2008; Shekar et al., 2003).

The multiagent system provides a powerful tool for advertising personalized offers (a shop owner will be able to publicise his offers to the shopping mall users), and a communication system between management, the commercial sector or shoppers. In this way, it is possible to obtain an intelligent environment for a shopping mall based on virtual organizations composed of:

- clients or users
  - interested in one or more products or promotions;
  - interested in mechanisms for route planning when a user wants to spend time in the mall;

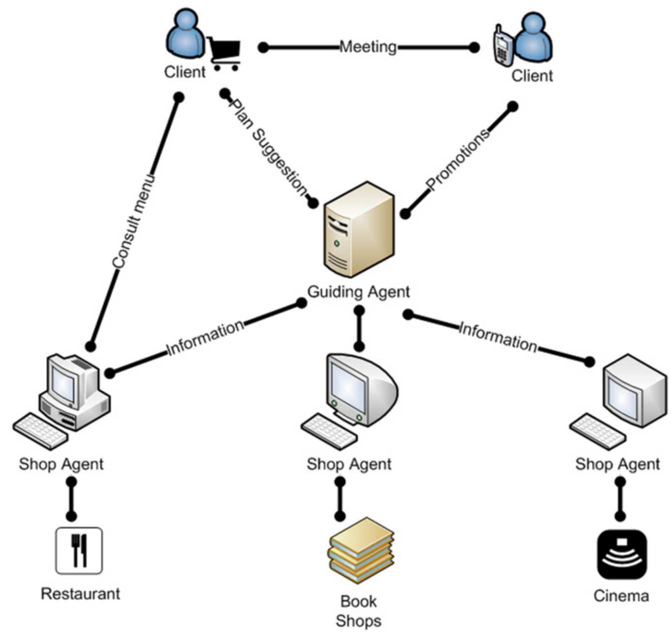

Fig. 3. Shopping mall scenario.

  - interested in receiving alerts or news;
  - interested in contacting other clients in the mall with similar profiles or interests.
- shoppers
  - interested in a tool for advertising personalized offers (a shop owner will be able to publicise his offers to the shopping mall users);
  - interested in improving certain aspects of their shops.
- directorship
  - requiring updated information to monitor and control the evolution of the mall and that can be used to obtain knowledge for decision support;
  - requiring novel solutions in terms of security or incidents resolutions and management.
- context-aware devices (radiofrequency identification—RFID, temperature sensors)
  - providing facilities of control, location and identification of individuals inside the shopping mall and serving as data sources to monitor the changing conditions of the environment.

The identification and characterization of such functionalities after studying the requirements of the mall problem are the inputs for the analysis and design of the system and constitute the base for obtaining the roles in the system from the organizational point of view. These roles are grouped into units depending on functionality criteria. A unit represents groups of agents and establishes the topological structure of the system. It is also a recursive concept, so units can be defined within other units. It enables the representation of organizational structures like hierarchy, matrix, coalition, etc. (Argente et al., 2007). Furthermore, it indicates what the structural positions of the system are (i.e. member, supervisor, subordinate), as well as the relationships among these positions imposed by the structure. The roles are defined inside each unit and represent the functionality required to achieve the objective of the unit. The roles can have associated norms aimed at controlling the actions of the role. The OMS establishes a hierarchy of roles, so any agent that plays a specific role is enabled to request or offer services related to superior hierarchical roles, provided that organizational norms do not explicitly forbid it.

4.2. Analysis and design of the mall scenario

Once we were able to examine the elements that constitute our scenario, the motivation for our system and its objectives, we could begin to define the roles that would form part of the architecture, taking into account the advantages of the THOMAS architecture. These include the following:

- *Communicator*: Role in charge of managing the connections that each user makes.
- *Finder*: Role in charge of finding users with similar tastes.
- *Profile Manager*: Role in charge of creating and defining the client profile.
- *Promotions Manager*: Role in charge of suggesting promotions and offers.
- *Warehouse Operator*: Role in charge of managing all inquiries made on the warehouse, managing updates and monitoring product shortages.
- *Analyst*: Role in charge of auditing sales information and the degree of client satisfaction.
- *Planner*: Role that offers recommendation and guidance services to the shopping mall clients. This role is able to dynamically plan and replan in execution time. It suggests routes that clients might want to take through the mall, according to their individual preferences.
- *Client Manager*: Role in charge of managing the connections between the mall clients, managing the profiles for clients that are visiting the mall, monitoring the state of the clients and managing the notification service for the mall.
- *Incident Manager*: Role that manages and resolves incidents, offers a client location service, and manages an alarm system.
- *Directory Manager*: Role responsible for managing the mall's store directory, including businesses, products, promotions and offers.
- *Device Manager*: Role that allows the interactive elements within the environment to interact. It deals with devices that use technologies such as RFID, etc.

The structural design of the system is also carried out, following, as noted above, the methodological guide GORMAS. First the dimensions are analysed, and then the most appropriate structural organization (Argente et al., 2009) is identified.

For our case study, this process is modelled as a congregation (*ShopppingUnit*) with five units, each of which is dedicated to a type of functionality within the scenario.

These five units are:

- *ClientUnit*, contains the roles associated with the client: Communicator, Finder, and Profile Manager;
- *BusinessUnit*, contains the roles associated with a business: Promotions Manager, Warehouse Operator;
- *ManagingUnit*, contains the roles assigned with global management tasks for the mall: Incident Manager, Client Manager, and Analyst;
- *RecommendationUnit*, contains the roles dealing with recommendations or suggestions made to the client: Planner and Directory Manager;
- *DeviceUnit*, which contains the roles associated with the management of devices: Device Manager.

The diagram of the structural view of the organizational model, adapted according to the congregation pattern, is shown in Fig. 4 (the notation is explained in Argente et al., 2009).

The notation used in Fig. 4 is simple. Fig. 4 represents the system's units grouped by the functionality of their roles (*ClientUnit*, *BussinesUnit*, etc.). The roles are represented within each unit. So for example, the roles *Planner* and DirectoryManager are within RecomendationUnit. These roles may be taken by an agent of the system and may offer or request services associated with that role.

The next step in the analysis and design process consists of detailing the services for each organization unit. Fig. 5 provides an example of the internal unit structure of the ClientUnit. The basic service offered by the ClientUnit is "ManageConnection", which is provided by the agents who have assumed the Communicator role.

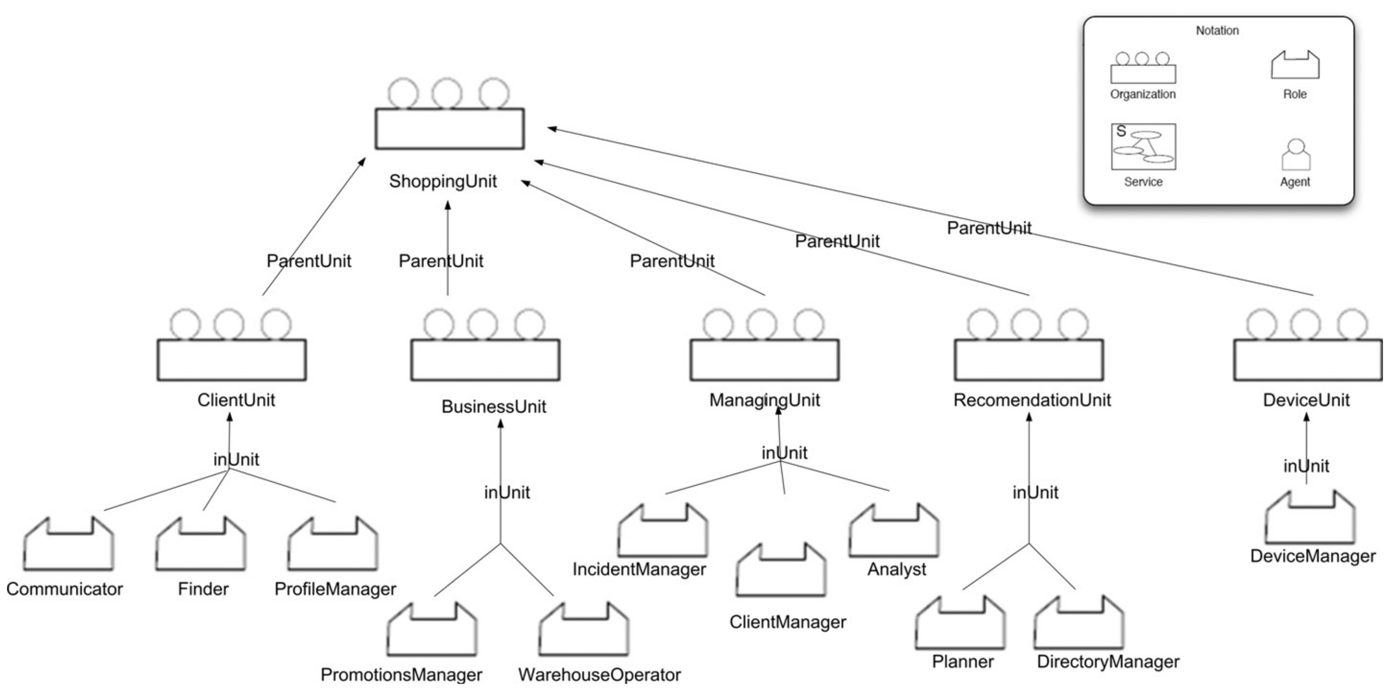

Fig. 4. Diagram of organization model: structural view.

The next subsections describe the services, norms and the planning mechanism for the case study. As the case study is very extensive, we will focus on a general description as well as on concrete and representative examples.

4.2.1. Services

A diagram representing the internal model is created for each unit (ClientUnit, BussinessUnit, ManagingUnit, RecomendationUnit and DeviceUnit). These models identify the services associated with each unit. A modelling of the functional view of the units is carried out, which allows us to identify the specific services for each domain. Then we detail as precisely as possible how each of the organizational services performs, how they interact within the environment, what interactions are established between the system entities, and how they handle the aspects related to open systems. For example, the basic service provided by the ClientUnit is "ManageConnection", which is provided by the different types of agents that assume the Communicator role. As shown in Table 2, the functionality offered by this service makes it possible for the clients to manage their connection to the system.

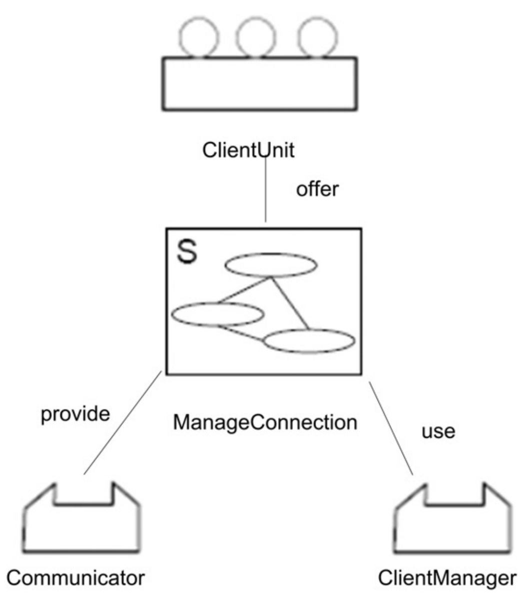

Fig. 5. Diagram of organization model: functional view of ClientUnit.

Similarly, within the BusinessUnit there are roles associated with a particular business and as a result, the services offered will be related to the corresponding promotions, products and sales (e.g., SendPromotion or RetrievePromotion). The services related to ManagingUnit involve the overall management tasks within a shopping mall (e.g., system incidents, data analysis, surveys, client management, notices, etc.). RecommendationUnit is comprised of services that request recommendations or suggestions based on user preferences and certain restrictions (time, money, etc.). It also includes planning and replanning the route that the user will follow based on the suggested recommendations, and determines the validity and value of the proposed routes. The DeviceUnit services deal with the sensors embedded in the physical system (RFID).

4.2.2. Norms

Upon completing the modelling of the functional views of the units, which allows us to identify the services particular to each domain and to detail the performance of the organizational services as precisely as possible, the next step is to define the norms in order to establish the control and management of the services. The type of services that can be offered is controlled by the system through the norms. Each possible type of performance within the system is controlled by the norms as defined by the following syntax (according to the normative language defined in (Criado et al., 2009)):

```
<norm>::=<deontic_concept>(<action>
[<temporal_situation>]
[IF <if_condition>]) [SANCTION(<state>)]
[REWARD(<state>)]
```

where <deontic_concept>::=OBLIGED|FORBIDDEN|PERMITTED and the action will be a dialogue action (send message) or an action to request, provide or register a service, (REQUEST|SERVE|REGISTER <service>). With <if_condition> it will be possible to indicate the results of the functions or services.

The internal functionality of the services is the responsibility of the agents that offer them, although it is the system that specifies the service profiles as well as the norms regarding the order of requests or the results that can be offered. This allows the system to respond with sanctions against any illicit or inappropriate behaviour on behalf of the user. The OMS will internally store the list of norms (classified according to type) that define the

Table 2
ManageConnection service in ClientUnit.

Service specification
Name: ManageConnection
Description: Manage client connection
Supplied by: SF
Required by:
ClientRole: ClientManager
ProviderRole: Comunnicator

Name	Description	Mand.	Type	Value range	Default
Input parameters					
Requesttime	Connection time	Yes	Date		
Connectiondata	Connection Data	Yes	String		
Operation	Kind Connection	Yes	String		
Output parameters					
Connection	Connection established	Yes	Connection		
Precondition					
–					
Postcondition					
–					

affected role, the content of the norm and the roles in charge of controlling whether the norm is followed. The OMS will also keep a list of existing roles for the units and their relationships, as well as the role they assume, i.e., which role is assumed for each entity and in which unit at any given time. Upon the initialization of the system, the OMS has the structural information of the system (roles, units, relationship between units and roles, relationship between roles and agents), so it can create the complete structure of the organization of agents.

A set of norms was defined in our system in order to control the performance within each of the units. This set of norms is managed by the OMS due to the fact that this performance is carried out by means of the services offered by the OMS. As it has been mentioned, further control of the norms regulating the system will be carried out by the agents playing the proper roles (the ones in charge of controlling such norms) and techniques coded by their designer (agents that may be developed in any toolkit but that should communicate in FIPA-ACL and offer/use services in THOMAS). This way, for example, an agent acting as Communicator within the ClientUnit is required to register ManageConnection services, and the OMS is in charge of controlling it. If it does not abide by this norm, it will be sanctioned and banished from the unit. The sanction is logical since if there is no connection established within a set amount of time, none of the other system tasks can be carried out:

```
OBLIGED Communicator REGISTER manageConenction
(?requestTime, ?connexionData, ?operacion)
BEFORE deadline SANCTION (OBLIGED OMS SERVE
Expulse (?agentID Communicator ClientUnit))
```

Likewise, we defined a complete set of norms that control all of the system performances.

4.2.3. Example of service organizing with THOMAS

One of the primary capabilities of the MAS proposed in this paper is the capability of reorganization. This is a very common feature in the organizations, but has not been addressed by the existing approaches and requires novel solutions. In this way, THOMAS incorporates a planning-based approach in order to provide a reorganization capability. We have developed a planning service that has been integrated within the multiagent system (Bajo et al., 2009) and modelled through THOMAS. It involves a self-adapting mechanism that facilitates the automatic assignment of tasks within the multiagent system. This section demonstrates the sequence of tasks that are executed within the system when a planning service is requested, and how THOMAS can generate the organization configuration and ensure the planning takes place. We will demonstrate the manner in which the architecture permits the reorganization of the MAS when new agents are introduced. The system evolves by replanning the guiding routes that clients can follow. The system considers the client objectives, the available time, and financial limitations, and proposes the optimal route according to the client's profile.

The first step is to define the structural components of the organization, i.e., the units that will be used (which are initially empty), the system roles and the norms. The requirements for the indicated services will be registered in the SF, thus establishing their respective profile (structure for the entrances/exists, preconditions/postconditions that should be met). To accomplish this, it is possible to either call on the basic services from the OMS registry for the structural components, or use the API that directly executes the same functionality. As a result, a congregation type *ShoppingUnit* is created, which represents the organization whose objective is to control the mall that will be monitored. There are five basic internal units, *ClientUnit*, *BusinessUnit*, *ManagingUnit*, *RecommentationUnit* and *DeviceUnit*, each one dedicated to the functionalities that have been previously noted. The list of system units will remain registered in the UnitList of the OMS, as shown in Fig. 6. The roles for each unit are defined, and each of their attributes indicated (visibility, position and heritage). Because it is possible for the same name of a role to appear in different units, the name of the unit should also be mentioned. This information is stored in the OMS RoleList.

The SF will list the services that are needed for the functionality of the system. The basic services are those that are essential (as defined by the norms) when the units are being created. The SF will keep a registry of the services offered by each entity, the action taken by the service, the type of role that can request (client role) and offer (client provider) the service, and its profile. These services are only assigned one profile, i.e., one structure of entrances/exits, preconditions/postconditions that they must carry out. However, since it is not yet known which role will perform the service, they cannot have a process or grounding assigned to them. That is, there is still no agent within the system that is taking this role and is implementing this service in some way (it has not an assigned grounding). As a result, the EntityPlayList for the OMS will still be empty. The agents have not yet begun to "play" within the system.

At this point, external agents can request the current list of services and decide whether to enter and form part of the organization, and with which type of role. In the following example, two clients will use their mobile device to send a request to the system in order to find the most optimal route for them to follow so they can perform their conditions (time, money, etc.). To do so, there must be:

- Agents C1 and C2: represent the clients that would like to receive a planning route in the mall; they will enter the system.
- Co1, Pe1 and Pl1 acting as agents that will carry out the roles of *Communicator*, *ProfileManager* and *Planner*, respectively, and can offer and/or request services from others with whom they are associated according to the SF.

All of these agents are first initiated on the THOMAS platform and associated to the virtual "world" organization. As such, in the OMS they will play the "member" role in the "world" organization. Asking the SF about which services exist in the system will generate the following answer:

```
ClientUnit Requires ManageConnection ClientRo-
le=ClientManager; ProvRole=Communicator;
```

The *ClientUnit*, *BusinessUnit*, *ManagingUnit*, *RecommentationUnit* and *DeviceUnit*, will already be visible in the "world" with a series of available services for the agents that wish to perform these tasks according to the roles they assume (Fig. 7). The service has no assigned "grounding", which cannot therefore be requested. That is, there is still no agent has taken the role can perform this service, that is able to implement it in some way. However it can have a functionality added and thus take on the role of *Communicator*. The Co1 agent would like to offer this functionality, and so requests the *Communicator* role from the *ClientUnit*: `AcquireRole(ClientUnit, Communicator)`.

If all goes well, the OMS will register that Co1 is assuming the role of *Communicator* in *ClientUnit* within the Entity Play List. As we can see in Table 3, the Entity Play List shows the roles adopted by agents within THOMAS. The Co1 agent has taken all the normal steps for acquiring a role within THOMAS.

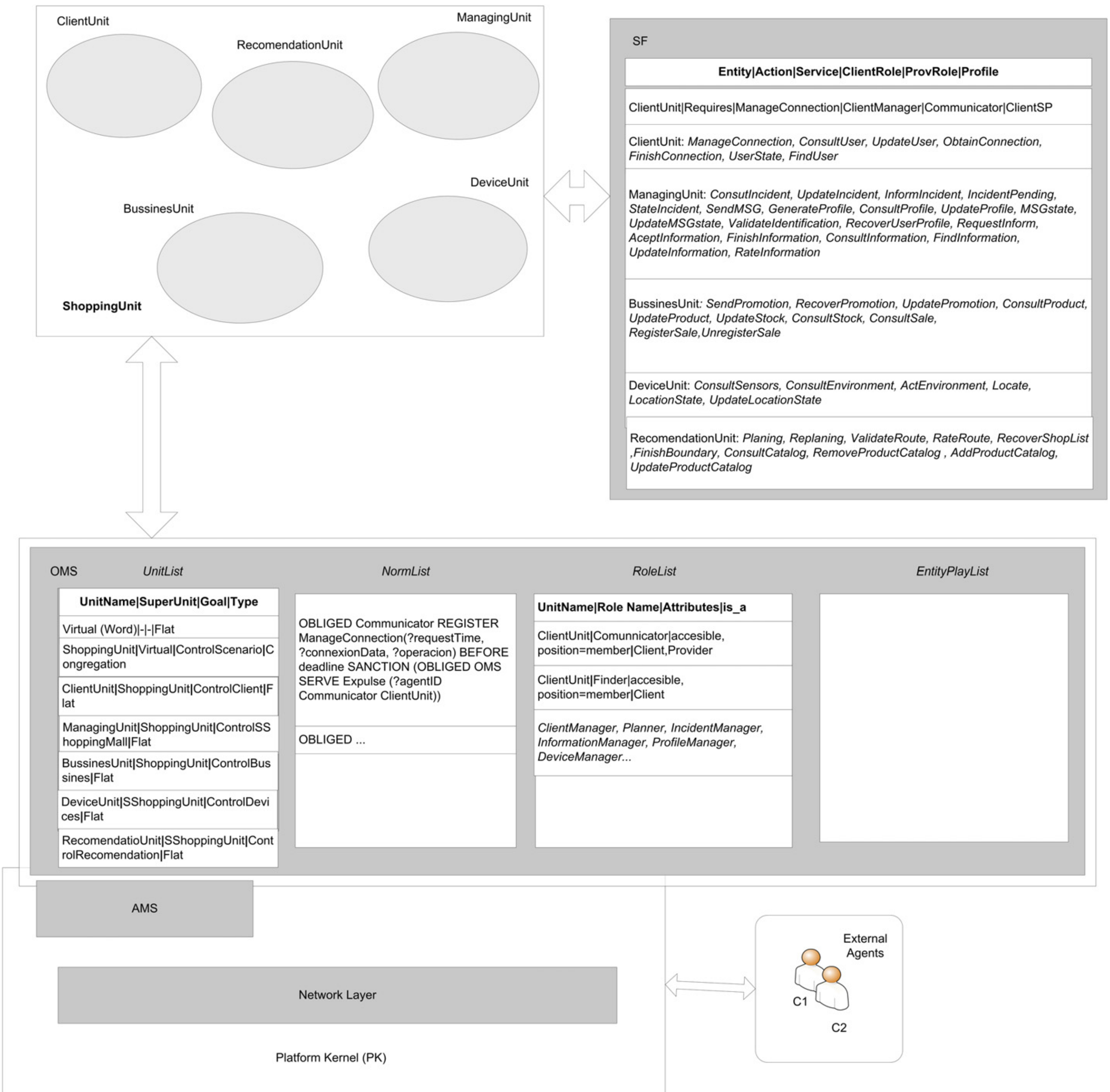

Fig. 6. Architecture of the initial system with an empty framework.

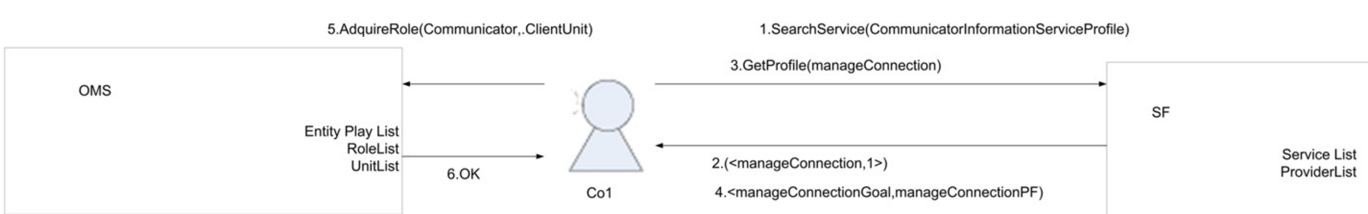

Fig. 7. Agent Co registering.

Fig. 7 illustrates these steps. Once Co1 has been registered as a member of the THOMAS platform; it asks SF which defined services have a profile similar to its own "communicator information service". This request is carried out using the SF SearchService (message 1), in which CommunicatorInformationServiceProfile corresponds to the profile of the "ManageConnection" service implemented by Co1.

Table 3
Entity play list.

Entity	Unit	Role
Pl	World	Member
Co1	ClientUnit	Communicator

The SF returns service identifiers that satisfy these search requirements together with a ranking value for each service (message 2). Ranking value indicates the degree of suitability between a service and a specified service purpose. Then Co1 executes GetProfile (message 3) in order to obtain detailed information about the *ManageConnection* service. Service outputs are "service goal" and "profile" (message 4). The *ManageConnection* profile specifies that service providers have to play a Communicator role within *ClientUnit*. Thus, Co1 requests the AcquireRole service from the OMS in order to acquire this provider role (message 5). AcquireRole service is carried out successfully (message 6), because *ClientUnit* is accessible from virtual organization, thus Co1 is registered as a *Communicator*.

Within each unit there will be another request to know which services exist. *BusinessUnit*, *ManagingUnit*, *RecommentationUnit*, and *DeviceUnit* will return the services that are necessary for planning. The profiles function will determine that Co1 is also interested in assuming the *DeviceManager* role since in this case it wants to interact with elements within the environment. Co1 will use this role to act as intermediary to process the signals that come from the user's devices and make them comprehensible within the system. It will allow the order requested by a user through a device to be understood and executed by the specific device that is the object of the order: AdquireRole(DeviceUnit, DeviceManager). It will now be registered as a member of DeviceUnit in the role of DeviceManager. This role will require the agent to register the Locate service and associate the process and grounding that it considers most suitable. If this is not accomplished within a determined amount of time, it will be banished. The norm specifically is:

```
OBLIGED DeviceManager REGISTER Locate(?route)
BEFORE deadline SANCTION (OBLIGED OMS SERVE Expulse
(?agentID DeviceManager DeviceUnit))
```

The agent will be informed of this norm upon performing the AcquireRole, so that it can reason out the norm if it is a norm agent (or ignore it otherwise). To keep other external agents from assuming the DeviceManager role, the agent will register a new incompatibility norm within the system. This norm makes it impossible for other agents to assume the same role.

```
RegisterNorm("norma1", "FORBIDDEN Member REQUEST
AcquireRole Message(CONTENT(role 'DeviceManager'))")
```

Agents C1 and C2 will act in a similar fashion. They will request to acquire the ClientManager role in order to access the basic services: FindClient, GenerateProfile, ConsultProfile, UpdateProfile, MSGState and UpdateMSGState. The Entity Play List and the units will end up as shown in Fig. 8.

Within the execution framework, we have the congregation type of *ShoppingUnit* that represents the organization, while internally we have the four basic units *ClientUnit*, *ManagingUnit*, *RecomendationUnit* and *BussinesUnit y DeviceUnit*. The list containing the system units is registered in the OMS *UnitList*, the norms are registered with the *NormList* and the roles with the *RoleList*. Both the *NormList* and the *EntityPlayList* will continue to be updated dynamically to register the performance, norms and agent roles within the system.

At this point, the participating agents within THOMAS have requested and acquired the services and roles necessary to carry out the planning needed for each of the users. Each of the behavioural performance norms that were established within the THOMAS organizational structure were defined and respected when the units were initially empty, as well as when they were dynamically generated by each of the agents. Once the agents are in the system, the organization can evolve according to the plans that are carried out. The steps for planning a route for one of the agents, C1, can be summarized as follows:

- Once inside the THOMAS platform, C1 obtains a connection using the *obtainConnection* service provided by the *Communicator* role that has acquired the Co1 agent already immersed within the system.
- C1 locates the user's mobile device within the system acting as a client through the *Locate* service provided by Co1.
- Likewise, C1 will generate its profile using the *GenerateProfile* service and will be subsequently used by the *ProfileManager* Pl1 agent to know the data for that client.
- C1 uses the *Planning* service to request the *Planner* P1 agent to recommend a route based on the restrictions within its own user profile.
- At the same time P1 acts on Pl1 to use the *RecoverShopLists* service and obtain the points required to generate the route.
- P1 will provide the route to C1 who in turn will validate the route using the *ValidateRoute* service obtained from P1.
- C1 will update the profile data with the data obtained from the system using the *UpdateProfile* and *UpdateMSGState* services.
- At this time the path to follow is shown on the user's device.

5. Evaluation results

This section presents the results obtained after testing the THOMAS architecture in a real scenario. Initially, the organizational skills of THOMAS were evaluated. In order to do so, the options available in THOMAS to create organizational models were compared to other existing alternative architectures. After this, the improvements provided by the virtual organizations paradigm in the mall scenario are discussed.

5.1. Organizational models

The THOMAS architecture can be compared to other options currently available that can be used to create organizational models, platforms and agent architectures. New organizational-oriented platforms or middleware should consider aspects like organizational structure (group support, topology, role hierarchy, interactions and social norms) and organizational dynamics (i.e. agents joining into the organization, role enactment, agents' life cycle, behaviour control, adaptation).

A comparison of those aspects are considered by different platforms (including Jade as the most well-known platform) is shown in Table 4. As can be seen, THOMAS supports the most important aspects for the implementation and execution of organizational-oriented systems. These aspects, that are not directly supported by Jade, have been used in the design and subsequent implementation of the new version of the system. Besides, THOMAS incorporates certain organizational features that are not supported by Madkit, Jack Teams, Ameli or S-Moise+. The availability of these organizational characteristics makes THOMAS unique in its conception and has provided many profits,

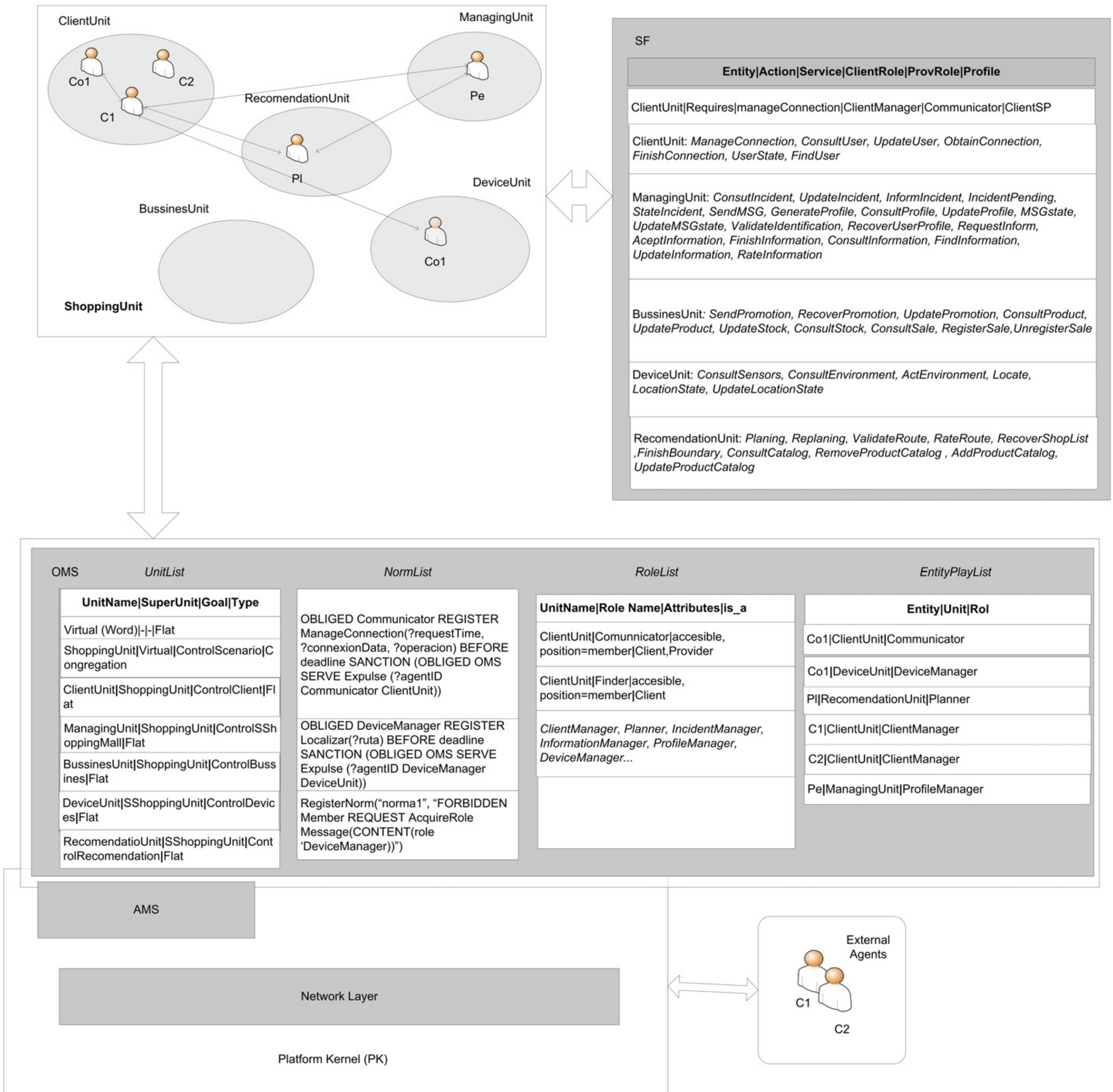

Fig. 8. System architecture with execution framework.

as demonstrated in the new version of the system presented in the case study.

However, in order to evaluate the impact of the new and easier method to develop MAS using an organizational paradigm, it is necessary to revise the behaviour of the multiagent system in terms of performance. A prototype was constructed in the mall scenario based on THOMAS that could be compared to the previous existing system (Bajo et al., 2009) and the improvements obtained could be quantified, as can be seen in the results shown in the next subsection (Figs. 10 and 11). The initial prototype was modelled as a fixed society of agents: the CBP guiding agent, Shop agents situated in each shop and User agents situated in the user mobile device (Bajo et al., 2009). The MAS shown in Fig. 2 is not open and the re-organizational abilities are limited, since the roles and norms cannot be dynamically adapted. As can be seen in Table 5, the system proposed in this paper provides several functional, taxonomic, normative, dynamics and adaptation properties. The organizational properties are a key factor in an architecture of this kind, but the capacity for dynamic adaption in execution time, not only for routes (functional adaption), but also for the structure of the organization, can be considered as a differential characteristic of the THOMAS architecture.

The initial system (v1) was designed using a combination of the Gaia and AUML design methodologies that do not provide mechanisms for modelling using an organizational paradigm. The new version of the system (v2) was designed using organizational oriented methodologies, and more concretely the GORMAS methodology (Argente et al., 2009) supported by the THOMAS

Table 4
Comparison of organizational features considered in different platforms.

Organizational features	Jade	Ameli	MadKit[a]	Jack Teams	S-Moise+	THOMAS
Agent Model	✓		✓	✓		
Estructure						
Group			✓	✓	✓	✓
Topology					✓	✓
Roles		✓	✓	✓	✓	✓
Interactions	✓	✓	✓	✓	✓	✓
Norms		✓			✓	✓
Dynamics						
Agent joining		✓			✓	✓
Role enactment		✓	✓		✓	✓
Behaviour control	✓	✓		✓	✓	✓
Org. joining						✓
Adaptation					✓	✓

[a] Madkit: www.madkit.org

Table 5
Comparison of previous and current system.

Properties	System v1	System v2
Functional		
BDI Model	✓	✓
Taxonomic		
Group		✓
Topology		✓
Roles		✓
Interactions	✓	✓
Normative		
Norms		✓
Dynamics		
Agent joining	✓	✓
Role enactment		✓
Behaviour control	✓	✓
Org. joining		✓
Adaptation		
Taxonomic		✓
Normative		✓
Functional	✓	✓

architecture. The phases proposed by GORMAS cover the analysis and design of the organizational structure, as well as the design of the dynamics of the organization. The design of the dynamics of the architecture is a very important characteristic, since one of the substantial improvements of the system in the second prototype is the ability for dynamic adaption at execution time. Fig. 8 makes use of the GORMAS notation to illustrate an example of change in the taxonomic dimension of the organization, when a *BusinessUnit* is transformed in a federation. A federation is a group of agents with a broker that represents the group and is responsible of receiving the messages of its group and sending messages to another brokers. The mall can be modelled as an organization with three main business (entertainment, food and clothes), which functionalities are defined attending at the services provided. At any given moment, the organizational structure of the system can be re-adapted transforming the *BussinesUnit* in a federation. In this way, every *BussinesUnit* can be treated in the organization with its own norms. In the federation there exists an initial unit of the Team type, which integrates the different federations that are implemented following a hierarchy. The *BussinesUnit*, once acquires a federation structure, can be modelled as a team, where the member can communicate each others to achieve the common functional objectives.

In the case study presented in this paper, the common objectives can be clearly identified, as the business are aimed at attracting as clients as possible, improving the quality of the services and their overall productivity. The supervisor role appears in the system, and is adopted by the internal agents of the system. This role receives requests and stores information about the members (business). As such, the supervisor acts as an interface between every business and those agents in the mall playing the *PromotionsManager* and *WarehouseOperator* roles.

THOMAS offers the necessary functional facilities to complete this process in an easier manner, with the definition and specification of the new units and roles in the organization. This fact was not possible in the initial version of the system, and to distinguish the functional characteristics of these agent types it would be necessary to define a new agent type, with its own functionalities and requirements. Furthermore, the overall system should be modified in order to take into account the characteristics of the new agent type in a global manner. THOMAS provides a novel mechanism to dynamically redefine and reorganize the functional, structural and normative characteristics of the system, and this can be achieved in a simple way, using the registration or deregistration services of the architecture for roles, norms and units. As can be seen in Fig. 9, the self-adaptation capability is a key factor in the system not only to react to the changes in the environment, but also to improve the overall functioning of the organization.

One of the main differences regarding the initial system is that the functionalities of the agents are not inside their structure (Bajo et al., 2009) but modelled as services (offered or demanded), and entities (agents) providing or consuming these services. This new perspective provides the system with a higher ability to recover from errors and a better flexibility to change their behaviour at execution time. Next subsection presents the empirical results obtained in the mall scenario aimed at quantitatively evaluate the proposed approach.

5.2. Results in the mall scenario

Several tests have been carried out in order to compare the overall performance of the shopping mall multiagent system regarding its previous version, the latter having used THOMAS. The tests were basically aimed at comparing response times and number of crashes in both systems, and consisted of a set of requests delivered to the services planning mechanism available in the mall for generating guidance suggestions. The performance of the new prototype improved progressively as the system learnt from the past experiences acquired during the simulation. The planning mechanism uses a case-based reasoning (CBR) engine (Aamodt and Plaza, 1994; Corchado et al., 2004, 2008), that allows it to make use of past experiences to find the best plans to achieve goals. The purpose of CBR is to solve new problems by adapting

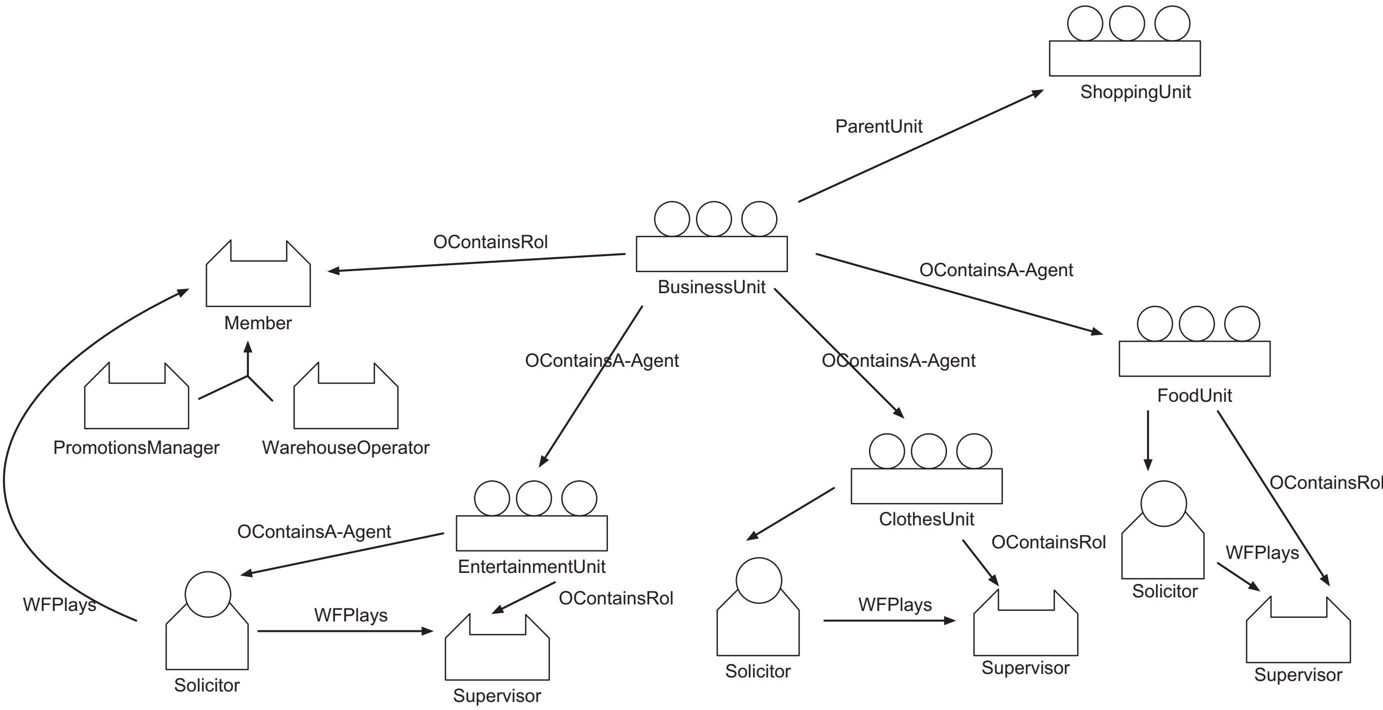

Fig. 9. Business unit structural view.

solutions that have been used to solve similar problems in the past. The CBR system performs a reasoning cycle that consists of four sequential phases: retrieve, reuse, revise and retain (Aamodt and Plaza, 1994). Each of these activities can be automated, which implies that the whole reasoning process can be automated to a certain extent (Corchado and Lee, 2001). According to this, agents implemented using CBR systems could reason autonomously and therefore adapt themselves to environmental changes.

We noted a significant improvement of the behaviour of the architecture agents at the organizational and resource level. The results obtained from these tests demonstrate a notably improvement in the average time required to accomplish the plans, the number of crashed agents, and the number of crashed services.

Fig. 10 illustrates the improvement that was achieved with regards to the system response times, due primarily to the structural change. As can be seen in Fig. 10, as the number of simultaneous agents increases, the performance of the previous prototype in terms of response time decreases, while the response time for the THOMAS-based prototype remains stable. In other words, the system went from having a mainly centralized internal composition in which a coordinating agent was in charge of directing the primary tasks for generating guiding routes, to having an organizational and dynamic agent-based structure in which each specific role or service can be acquired by independent agents. As a result it is possible to distribute computationally complex tasks such as planning into services, thus reducing the response time for changes in the environment that provoke a replanning of guiding routes. This reduction was shown to optimize the workday for the clients in the mall, increasing their time of productivity by 7% by dynamically assigning the points, which minimized the area and the time to cover during the guiding routes.

In addition to having a non-centralized organization, we were able to achieve an improvement in the robustness and scalability of the system. One of the main problems of the initial prototype was the number of agent's crashes, since the agents have a high computational load. Fig. 11 illustrates the results obtained with regards to the number of agents with errors in both systems.

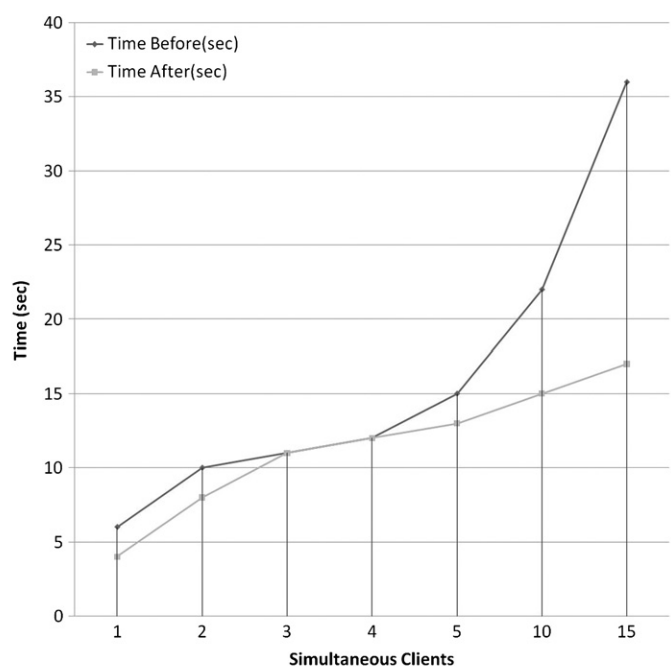

Fig. 10. Time needed for both systems to generate guidance suggestions for a group of clients.

As shown, the rate of agents with errors is reduced by an average of 21%. This reduction is essentially due to the distribution of computationally complex system tasks. The THOMAS architecture allows the agents to integrate within web services, and this fact causes a reduction of the agent's computational load.

Additionally, the agents with errors in the new system can be instantiated again and assume their role within the organization once more, because of the reorganization feature provided by THOMAS. These data demonstrate that this approach provides a higher ability to recover from errors and a good ability for self-organizing.

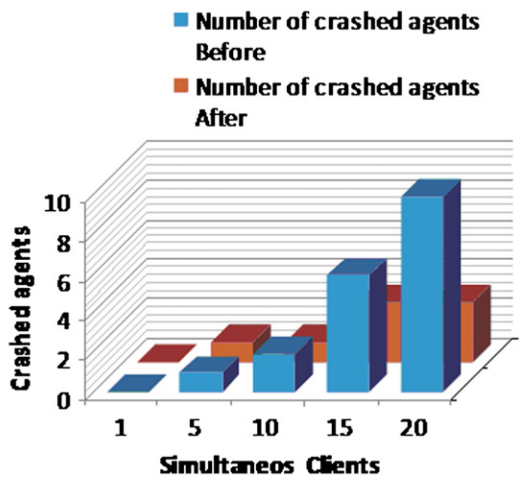

Fig. 11. Number of agents and services crashed for both versions of the system.

6. Conclusions

An important issue in the development of real open MAS aimed at modelling virtual organizations is to provide developers with methods, tools and appropriate architectures which support all of the requirements of these kinds of systems. Traditional MAS development methodologies are not suitable for developing open agent-based virtual organizations because they assume a fixed number of agents that are specified during the system analysis phase. It then becomes necessary to have an infrastructure that can use the concept of agent technology in the development process, and apply decomposition, abstraction and reorganization methods. Based on this and taking into account the virtual organization's paradigm, we identified a set of requirements for the construction of complex software open systems that address this kind of problems and have come up with a set of requirements for a multiagent architecture aimed at notoriously open environments: THOMAS. In presenting this architecture, we have considered each of these points in the process of integrating our previous study (Bajo et al., 2009).

The hypothesis in the present study involved applying autonomous capabilities to a virtual organization, thus allowing a dynamic response when facing the expected changing situations produced by the adaption and/or evolution of the organization. As a result, the organization would be capable of detecting potential situations of interest, such as functioning errors, and be able to manage them by maximizing its flexibility and adaptive ability. The adaption of the organization involves, among other aspects, its norms, agreements, obligations and topological structure. The multiagent system presented in this paper manages these techniques by using the THOMAS architecture in a dynamic environment, and provides self-adaptation capabilities. In this sense, our proposal adapts previous studies in the area of MAS (Bajo et al., 2009) to our domain, specifically with the paradigm developed in the THOMAS architecture. In this study, the use of THOMAS allowed us to dynamically model and develop concepts for a route planning system. Specifically, our proposal allowed us to directly model the organization for a shopping mall according to a previous basic analysis, to define agent roles, functionalities and restrictions in a dynamic and open manner, and to add service management capabilities (discovery, directory, etc.) within the platform beforehand.

We can conclude that THOMAS was able to provide the necessary level of abstraction for developing our system, and the set of tools for facilitating its development. In the THOMAS architecture, agents can offer and invoke services in a transparent way from other agents, virtual organizations or entities. Additionally, external entities can interact with agents through the use of the services offered. A case-study example was employed to illustrate not only the usage of THOMAS components and services, but also the dynamics of the applications to be developed with such architecture. Examples of THOMAS service calls were shown through the use of several scenarios, along with the evolution of different dynamic virtual organizations.

Acknowledgements

This work has been supported by the Spanish Ministry of Industry, Tourism and Commerce, project MCYT—TIN 2009-13839.

References

Aamodt, A., Plaza, E., 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7, 39–59.
Aldewereld, H., Dignum, F., García-Camino, A., Noriega, P., Antonio Rodríguez-Aguilar, J., Sierra, C., 2006. Operationalisation of norms for electronic institutions. In: AAMAS 2006 and ECAI 2006 International Workshops, COIN.
Ali, R., Bryl, V., Cabri, G., Cossentino, M., Dalpiaz, F., Giorgini, P., Molesini, A., Omicini, A., Puviani, M., Seidita, V., 2008. MEnSA Project—Methodologies for the Engineering of Complex Software Systems: Agent-based Approach. Technical Report 1.2, UniTn.
Arcos, J.L, Esteva, M., Noriega, P., Rodriguez-Aguilar, J.A., Sierra, C., 2005. Engineering open environments with electronic institutions. Engineering Applications of Artificial Intelligence 18 (2), 191–204.
Argente, E., Giret, A., Valero, S., Julian, V., Botti, V., 2004. Survey of MAS methods and platforms focusing on organizational concepts. In: Vitria, J., Radeva, p., Aguilo, I (Eds.), Frontiers in Artificial Intelligence and Applications, pp. 309–316.
Agent-Oriented-Software. JACK intelligent agents: JACK teams manual. Release 4.1, 2004.
Argente, E., Julian, V., Botti, V., 2006. Multi-agent system development based on organizations. Electronic Notes in Theoretical Computer Science 150, 55–71.
Argente, E., Palanca, J., Aranda, G., Julian, V., et al., 2007. Supporting agent organizations. In: Proceedings of the CEEMAS'07, pp. 236–245.
Argente, E., Julian, V., Botti., V., 2008. MAS modelling based on organizations. In: Luck, Michael, Gomez-Sanz, Jorge J. (Eds.), Agent Oriented Software Engineering IX LNCS, vol. 5386, pp. 16–30.
Argente, E., Botti, V., Julian, V., 2009. GORMAS: an organizational-oriented methodological guideline for open MAS. In: 10th International WS on Agent-Oriented Software Engineering, pp. 85–96.
Bajo, J., Corchado, J.M., De Paz, Y., De Paz, J.F., Rodríguez, S., Martín, Q., Abraham., A., 2009. SHOMAS: intelligent guidance and suggestions in shopping centres. Applied Soft Computing.
Baumer, G., Breugst, M., Choy, S., Magedanz, T., 2000. Grasshopper: A Universal Agent Platform Based on OMG MASIF and FIPA Standars. Agents Technology in Europe.
Boella, G., Hulstijn, J., van der Torre, L., 2005. Virtual Organizations as Normative Multiagent Systems. HICSS IEEE Computer Society.
Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos., J., 2004. Tropos: an agent-oriented software development methodology. AAMAS 8 (3), 203–236 ⟨doi:http://dx.doi.org/10.1023/B:AGNT.0000018806.20944.ef⟩.
Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., Botti, V., 2009. Service oriented MAS: an open architecture. In: Decker, Sichman, Sierra, Castelfranchi (Eds.), Proceedings of the 8th International Conference on AAMAS, pp. 1291–1292.
Corchado, J.M., Lee, B., 2001. A hybrid case-based model for forecasting. Applied Artificial Intelligence 15 (2), 105–127.
Corchado, J.M., Pavón, J., Corchado, E., Castillo, L.F., 2004. Development of CBR-BDI agents: a tourist guide application. In: Proceedings of the ECCBR'04, Lecture Notes in Artificial Intelligence, vol. 3155, Springer, Berlin, pp. 547–559. .
Corchado, J.M., Gonzalez-Bedia, M., De Paz, Y., Bajo, J., De Paz, J.F., 2008. Replanning mechanism for deliberative agents in dynamic changing environments. Computational Intelligence 24 (2), 77–107.
Cossentino, M., 2005. From requirements to code with the Passi methodology. Agent Oriented Methodologies IV, 79–106.
Criado, N., Argente, E., Julian, V., Botti, V., 2007. Organizational services for Spade agent platform. In: Proceedings of the 6th International Workshop on PAAMS.
Criado, N., Argente, E., Julian, V., Botti, V., 2009. Designing virtual organizations. In: 7th International Conference on PAAMS2009, vol. 55, pp. 440–449.
Criado, N., Julian, V., Botti, V., Argente, E., 2009 A norm-based organization management system. AAMAS Workshop COIN, pp. 1–16.
DeLoach, S., 2009. Multi-agent systems: semantics and dynamics of organizational models, IGI Global, Ch. Organizational Model for Adaptive Complex Systems, 1–26.

Dignum, V., 2003. A Model for Organizational Interaction: Based on Agents, Founded in Logic. Ph.D. Thesis, Utrecht University.
Dignum, V., Dignum, F., 2007. A logic for agent organization. In: Proceedings of the FAMAS@Agents'007.
Dignum, V., Vazquez-Salceda, J., Dignum, F., 2005. Omni: introducing social structure, norms and ontologies into agent organizations. Lecture Notes LNAI 3346, Springer.
Escriva, M., Palanca, J., Aranda, G., García-Fornes, A., Julian, V., Botti, V., 2006. A Jabber-based multi-agent system platform. In: Proceedings of the AAMAS06, pp. 1282–1284.
Esteva, M., Rodriguez-Aguilar, J., Sierra, C., Arcos, J., Garcia, P., 2001. On the Formal Specification of Electronic Institutions. Springer-Verlag. LNAI, pp. 126–147.
Esteva, M., Rosell, B., Rodriguez, J. A., Arcos, J. L., 2004. AMELI: an agent-based middleware for electronic institutions. In: Proceedings of the AAMAS04, pp. 236–243.
Ferber, J., Gutknecht, O., 1998. A meta-model for the analysis and design of organizations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems, IEEE Computer Society, pp. 128–135.
Ferber, J., Gutknecht, O., Michel., F., 2004. From agents to organizations: an organizational view of multi-agent systems. In: Giorgini, P., Muller, J., Odell, J. (Eds.), Agent-Oriented Software Engineering VI, vol. LNCS 2935. Springer-Verlag, pp. 214–230.
Foster, I., Kesselman, C., Tuecke., S., 2001. The anatomy of the grid: enabling scalable virtual organizations. International Journal of High Performance Computing and Applications 15 (3), 200–222.
Garcia, E., Argente, E., Giret, A., 2009. A modeling tool for service-oriented Open Multiagent Systems. The 12th International PRIMA 2009, vol. 5925, pp. 345–360.
Gâteau, B., Boissier, O., Khadraoui, D., Dubois, E. Controlling an interactive game with a multi-agent based normative organisational model. In: AAMAS 2006 and ECAI 2006 International Workshops, COIN, 2006.
Giret, A., Julian, V., Rebollo, M., Argente, E., Carrascosa, C., Botti, V. 2009. An open architecture for service-oriented virtual organizations. Seventh international Workshop on Programming Multi-Agent Systems, PROMAS 2009, pp. 23–33.
Greenwood, D., Calisti, M., 2004. Engineering web service—agent integration. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1918–1925.
Gutknecht, O., Ferber, J., 1997. Madkit: Organizing Heterogeneity with Groups in a Platform for Multiple Multi-agent Systems. Technical Report 97188 LIRMM.
Horling, B., Lesser, V., 2004. A survey of multiagent organizational paradigms. The Knowledge Engineering Review 19, 281–316.
Horling, B., Lesser, V., 2005. Using ODML to model multi-agent organizations. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology.
Howden, N., Rönnquist, R., Hodgson, A., Lucas, A., 2001. JACK intelligent agents—summary of an agent infrastructure. In: Proceedings of the 5th ACM International Conference on Autonomous Agents.
Hubner, J., Sichman, J., Boissier, O., 2002. A model for the structural, functional, and deontic specification of organizations in mulitagent systems, In: G. Bittencourt, G. Ramalho (Eds.), 16th Brazilian Symposium on Artificial Intelligence (SBIA02), vol. 2507 of LNAI, Springer, pp. 118–128.
Hubner, J., Sichman, J., Boissier, O.. 2006. S-Moise+: a middleware for developing organised multi-agent systems. In: Proceedings of the Workshop on Organizations in MAS, from Organizations to Organization Oriented Programming in MAS, LNCS, vol. 3913, pp. 64–78.
Jennings, N.R., Wooldridge, M., 2002. Agent-oriented Software Engineering. Handbook of Agent Technology.
Lopez, F., Luck, M., d'Inverno., M., 2006. A normative framework for agent-based systems. Computational and Mathematical Organization Theory 12, 227–250.
Luck, M., McBurney, P., 2008. Computing as interaction: agent and agreement technologies. In: IEEE SMC Conference on Distributed Human-Machine Systems, pp. 1–6.
Luck, M., McBurney, P., Shehory, O., Willmott., S., 2005. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink.
Molesini, A., Omicini, A., Denti, E., Ricci, A., 2006. Soda. A roadmap to artefacts. Engineering Societies in the Agents World VI LNAI 3963, 49–62.
Nguyen, T., Kowalczyk, R., 2005. Ws2jade: Integrating Web Service with Jade Agents. Technical Report SUTICT-TR2005.03, Centre for Intelligent Agents and Multi-Agent Systems, Swinburne University of Technology.
Noriega, P., Sierra, C., 2002. Electronic Institutions: Future Trends and Challenges. CIA, 14–17.
Nwana, S., Ndumu, D.T., Lee, L.C., Collis, J.C., 1999. ZEUS: a toolkit and approach for building distributed multi-agent systems. Agents 1999, 360–361.
Odell, J., Nodine, M., Levy, R., 2005. A metamodel for agents, roles, and groups. In: James Odell, J.M., Giorgini, P. (Eds.), Agent-Oriented Software Engineering (AOSE) V, Lecture Notes in Computer Science. Springer.
Pavon, J., Gomez-Sanz, J., Fuentes., R., 2005. The INGENIAS Methodology and Tools. Idea Group Publishing article IX, pp. 236–276.
Serrano, J.M., Ossowski, S., Fernández, A., 2004. On the impact of agent communication languages on the implementation of agent systems. In: Klusch, Ossowski, Kargupta, Unland (Eds.), Cooperative Information Agents VIII. Springer.
Shekar, S., Nair, P., Helal, A., 2003. iGrocer—A ubiquitous and pervasive smart grocery shopping system. In: Proceedings of the ACM SAC, ACM Press, pp. 645–652.
Soler, J., Julián, V., Rebollo, M., Carrascosa, C., Botti, V., 2002. Towards a real-time multi-agent system architecture. In: Proceedings of the Challenge 2002, AAMAS 2002, Bolonia.
Vrba, P., Macurek, F., Marik, V., 2008. Using radio frequency identification in agent-based control systems for industrial applications. Engineering Applications of Artificial Intelligence 21 (3), 331–342.
Zambonelli, F., Parunak, H., 2002. From design to intention: signs of a revolution. In: Proceedings of the 1st International Joint Conference on AAMAS, pp. 455–456.
Zambonelli, F., Jennings, N.R., Wooldridge, M., 2003. Developing multiagent systems: the gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM) 12 (3), 317–370.