THOMAS-MALL: A Multiagent System for Shopping and Guidance in Malls

S. Rodríguez[1], A. Fernández[2], V. Julián[3], J.M. Corchado[1], S. Ossowski[2], and V. Botti[3]

[1] Universidad de Salamanca
[2] Universidad Rey Juan Carlos de Madrid
[3] Universidad Politécnica de Valencia, (Spain)

Abstract. This article presents a case study in which the THOMAS architecture is applied in order to obtain a multi-agent system (MAS) that can provide recommendations and guidance in a shopping mall. THOMAS is made up of a group of related modules that are well-suited for developing systems in other highly volatile environments similar to a shopping mall. Because the development of this type of system is complex, it is essential to thoroughly analyze the intrinsic characteristics of typical environment applications, and to design all of the system components at a very high level of abstraction.

1 Introduction

This article presents a dependable solution for using a novel architecture in designing and building a system for guiding and advising users in a shopping mall. A shopping mall can be considered a dynamic and volatile environment in which shops change, promotions appear and disappear, and the products that are offered are continually changing. As such, a high level design with an abstract architecture is essential.

The architecture we used is THOMAS (*MeTHods, techniques and tools for Open Multi-Agent Systems*) [6][7]. THOMAS is a new architecture for open MAS and is made up of a group of related modules that are well-suited for developing systems in other highly volatile environments similar to a shopping mall. This design will use a high level of abstraction to determine which components are necessary for addressing all of the needs and characteristics of a shopping mall guidance system.

Artificial intelligence techniques have given way to new studies that allow, among other things, modeling the problem of a shopping mall in terms of agents and MAS. The shopping mall is turned into an intelligent environment where users are surrounded by these techniques, but do not need to adapt to them. One of the objectives of MAS is to build systems capable of autonomous and flexible decision-making, and that will cooperate with other systems within a "society" [5]. This "society" must consider characteristics such as distribution, continual evolution and flexibility, all of which allow the members (agents) of the society to enter and exit, to maintain a proper structural organization, and to be executed on different types of devices. All of these characteristics can be incorporated via the open MAS and virtual organization paradigm, which was conceived as a solution for the management, coordination and control of agent performance [8]. The organizations not only find

J. Cabestany et al. (Eds.): IWANN 2009, Part I, LNCS 5517, pp. 594–601, 2009.

the structural composition of agents (i.e., functions, relationships between roles) and their functional behavior (i.e., agent tasks, plans or services), but they also describe the performance rules for the agents, the dynamic entrance and exit of components, and the dynamic formation of groups of agents[3].

The goal of this study is to present a case study in which the THOMAS architecture is used to build an open MAS for guiding users through a shopping mall. We will propose an application for this architecture and will evaluate its appropriateness for developing an open MAS in a real environment. The first step of this research involves designing the components needed for addressing all the needs and characteristics of a shopping mall system. The design is based on the GORMAS (Guidelines for Organization-based Multi-Agent Systems) [1] methodology, which is specifically geared towards organizations.

This article is organized as follows: section 2 presents the principle characteristics of the architecture and methodologies used; section 3 indicates the MAS that was developed for the actual case study (the shopping mall), and highlights the characteristics provided by the type of architecture used for its development; and the final section presents some of the conclusions obtained by this research.

2 THOMAS Outline

THOMAS [6][7] is the name given to an abstract architecture for large scale, open multi-agent systems. It is based on a services oriented approach and primarily focuses on the design of virtual organizations.

The architecture is basically formed by a set of services that are modularly structured. THOMAS uses the FIPA[1] architecture, expanding its capabilities with respect to the design of the organization, while also expanding the services capacity. THOMAS has a module with the sole objective of managing organizations that have been introduced into the architecture, and incorporates a new definition of the FIPA Directory Facilitator that is capable of handling services in a much more elaborate way, following the service-oriented architecture directives.

THOMAS consists of three principle components: *Service Facilitator (SF), Organization Manager Service (OMS)* and *Platform Kernel (PK).*

The SF primarily provides a place where autonomous entities can register service descriptions as directory entries. The OMS component is primarily responsible for specifying and administrating its structural components (role, units and norms) and its execution components (participating agents and the roles they play, units that are active at each moment). In order to manage these components, OMS handles the following lists: *UnitList*: maintains the relationship between existing units and the immediately superior units (SuperUnit), objectives and types; *RoleList*: maintains the relationships between existing roles in each unit, which roles the unit inherits and what their attributes are (accessibility, position); *NormList*: maintains the relationship between the system rules; *EntityPlayList*: maintains the relationship between the units that register each agent as a member, as well as the role that they play in the unit. Each virtual unit in THOMAS is defined to represent the "world" for the system in

[1] http://www.fipa.org (*Foundation for Intelligent Physical Agents*)

which the agents participate by default. Additionally, the roles are defined in each unit. The roles represent the functionality that is necessary for obtaining the objective of each unit. The PK component directs the basic services on a multi-agent platform and incorporates mechanisms for transporting messages that facilitate the interaction among the various entities.

From a global perspective, the THOMAS architecture offers a total integration enabling agents to transparently offer and request services from other agents or entities, at the same time allowing external entities to interact with agents in the architecture by using the services provided.

The development of MAS is typically based on a design that focuses on each agent independently, and is geared towards each agent's structure and performance. This research presents a new focus in which the design is directed at the organizational aspects of the agents, establishing two descriptive levels: the organization and the agent [4]. The system we developed used the GORMAS [1] organizational methodology.

3 Case of Study: Tormes Shopping Mall

The case study application facilitates the interaction between the users (clients in the shopping mall), the store or sales information, and recreational activities (entertainment, events, attractions, etc.), and defines the services that can be requested or offered. We developed a wireless system capable of incorporating agents that provide orientation and recommendation functionalities to the user, and that can be applied not only in a shopping mall, but also in other similar environments such as a supermarket, an educational facility, medical or health care center, etc.[2]. The clients use the agents via their mobile devices and RFID (Radio Frequency Identification) [9]technology in order to consult the store directory, receive special offers and personalized promotions, and ask for recommendations to navigate through the mall or locate other clients. Clients can also use the mechanisms available to them to plan a particular route that allows them to better spend their time in the mall and receive personalized notices.

There are different types of agents that come into play: (i) the *User* agent, which is in charge of managing client communication, finding and identifying other user devices, and maintaining the user's profile; (ii) the *Shop* agent, which is in charge of maintaining the warehouse (i.e., product database) and the promotions that can be offered to the clients; (iii) and the *Guiding* agent, which is charge of managing user profiles, controlling communications, analyzing the promotions, managing all incidents, and most importantly, planning the best route for each user according to the available resources and the user profile. The *Guiding* agent can be considered the heart of the system, as it receives the most current information from each of the mall's stores, and interacts directly with the clients to offer personalized services.

The first step in analyzing and designing the problem is to define the following roles that will exist within the architecture: *Communicator*: in charge of managing the connections that each user makes. *Finder*: in charge of finding users with similar tastes. *Profile Manager*: in charge of creating and defining the client profile. *Promotions Manager*: in charge of suggesting promotions and offers. *Warehouse*

Operator: in charge of managing all inquiries made on the warehouse, managing updates and monitoring product shortages. *Analyst:* in charge of auditing sales information and the degree of client satisfaction. *Planner*: 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*: 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*: manages and resolves incidents, offers a client location service, and manages an alarm system. *Directory Manager*: responsible for managing the mall's store directory, including businesses, products, promotions and offers. *Device Manager*: makes it possible for the interactive elements within the environment to interact. It deals with devices that use technologies such as RFID, etc.

We have also designed an organizational structure. We will first analyze its dimensions, and then proceed to identify the structure that is best suited to apply to the system [1]. Our case study is modeled as a conglomerate (*ShoppingUnit*) made up of five units, each one dedicated to one type of functionality within the setting. The five units are: (i)*ClientUnit*, contains the roles associated with the client: Communicator, Finder, and Profile Manager; (ii)*BusinessUnit*, contains the roles associated with a business: Promotions Manager, Warehouse Operator;(iii) *ManagingUnit*, contains the roles assigned with global management tasks for the mall: Incident Manager, Client Manager, and Analyst;(iv)*RecommendationUnit*, contains the roles dealing with recommendations or suggestions made to the client: Planner and Directory Manager;(v)*DeviceUnit*, which contains the roles associated with the management of devices: Device Manager.

The diagram in Figure 1c provides a structural view of the organizational model, which is adapted according to a conglomerate pattern. Different services are provided within each unit of the organization. The following section defines the services offered by the units, and uses an example to detail each one and how it has been modeled and described in the architecture. The type of role, the inputs and outputs, and a summary of the functionality for each unit are all explained. Figure 1b shows the internal model of the *ClientUnit*. The internal structure for *ShoppingUnit* and the remaining units was modeled in the same way.

a)

Service Specification

Name: gestionarConexión
Description: Gestiona la Conexión de un cliente
Supplied by: ST
Required by:
ClientRole: GestorDeClientes
ProviderRole: Comunicador

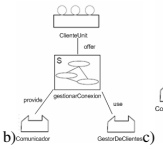
Input Parameters

Name	Description	Mand.	Type	Value Range	Default
requestTime	Tiempo de conexión	Si	date		
connexionsData	Datos de conexión	Si	string		
operacion	Tipo de operación sobre la conexión	Si	string		

Output Parameters

Name	Description	Mand.	Type	Value Range	Default
conexión	Conexión establecida entre dispositivos	Si	objectIon		

b)



c)

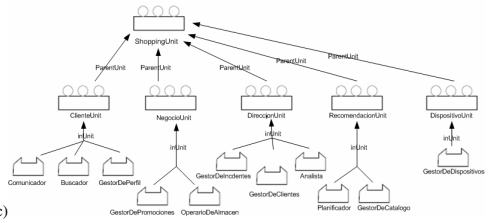


Fig. 1. a) *ManageConnection* service in *ClientUnit* b) Diagram of organization model: functional view of *ClientUnit* c) Diagram of organization model: structural view

One side of the diagram models the functional views of the units, which allows us to identify the services specific to each domain, while the other side precisely details the behavior of the organization services, how they interact with the environment, which interactions are established between the system entities, and how they approach the aspects of an open system. The next step is to define the rules in order to establish the control and management of the services. For example, the basic service provided by *ClientUnit* will be *ManageConnection*, which is provided by the agents that take on the role of Communicator. The functionalities offered by this service will allow the clients to control their connection to the system.

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).

The type of services offered is controlled by the system according to the established norms [7]. The internal functionality of the services is responsible for the agents that are offered, but the system is what specifies the agent profiles, as well as the rules to follow for ordering requests or offering results. In this way, when faced with illicit or improper client performance, the system can act to impose sanctions. The OMS will internally save the list of norms that define the role involved, the content of the norms, and the roles in charge of ensuring that the norm is met. We have defined a set of norms in our system for controlling the performance within each unit. This way, for example, an agent within *ClientUnit* that acts like *Communicator* is required to register a service as *manageConnection*. If it does not abide by these norms, it will be punished and expelled from the unit. The punishment is logical given that if the agent does not establish a connection within the allocated time, it cannot perform any of the other system tasks. *OBLIGED Communicator REGISTER manageConnection(?requestTime, ?connectionData, ?operation) BEFORE deadline SANCTION (OBLIGED OMS SERVE Expulse (?agentID Comunicator ClienteUnit))*

Similarly, we have defined a complete set of norms that will control all of the system performances.

3.1 Example of Service Planning with THOMAS

The system considers the client objectives, the available time, and financial limitations, and proposes the optimal route according to the client's profile. The planning model we propose was integrated within a previously developed MAS [2]. We will see the series of steps that are taken within the system when a planning route is requested, and how THOMAS generates the system configuration that will give way to the plan. The first thing is to define the structural components of the organization, that is, the units that will be involved (which are initially empty), the system roles and norms. The indicated service requirements will be registered in the SF. To do so, either the basic OMS services for registering structural components

Table 1. SF: Basic services

Service Facilitator					
Entity	**Action**	**Service**	**ClientRole**	**ProvRole**	**Profile**
ClientUnit	Requires	manageConnection	ClientManager	Communicator	ClienteSP
DeviceUnit	Requires	locate	Communicator/IncidentManager	DeviceManager	DispositivoSP
…	…	…	…	…	…

will be used, or the API will directly execute the same functionality. This way, a community type *ShoppingUnit* will be created, representing the organization, whose purpose is to control the shopping mall. It has five internal unit planes: *ClientUnit*, *BusinessUnit*, *ManagingUnit*, *RecommentationUnit*, and *DeviceUnit*, each of which is dedicated to the functionalities we have previously seen. Each unit defines the existing roles, indicating their attributes (visibility, position, etc) and who they inherit them from.

The SF will announce basic services that are required for the overall system functionality. The basic services indicate which services are required (according to the defined norms) when creating the units.

From this moment on, the external agents can request the list of existing services and decide whether or not to enter and form part of the organization and with which roles. In our case we have clients that use their mobile device to send a request to the system so that it can inform them on the optimal route to take within the shopping mall. In order to carry out this function, we have, for example *Co1*, *Pe1* and *Pl1* acting as agents that will carry out the roles of *Communicator*, *ProfileManager* and *Planner* respectively. Agents *C1* and *C2* represent the clients that would like to receive a planning route.

Initially, all the agents head towards the THOMAS platform and are associated with the virtual "world" organization. As such, the OMS will play the *member* role in the "world" organization. When SF is asked about existing services in the system, the following response is obtained: `ClientUnit Requires manageConnection ClientRole =ClientManager;ProvRole=Comunicator;`

Because the service doesn't have an assigned *grounding*, it cannot be requested. But a functionality can be added, thus obtaining the *Communicator* role.

The *Co1* agent wants to offer that functionality, for which it requests receiving the *Communicator* role for the *ClientUnit*: `AcquireRole(ClientUnit, Communicator)`

If all goes well, the OMS will register *Co1* in the role of *Communicator* in *ClientUnit* within the *Entity Play List*. This list shows the roles that the different agents assume within THOMAS.

The *Co1* agent has carried out all of the regular steps for acquiring a role within THOMAS. This process is illustrated in Figure 2a where 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 Communicator InformationServiceProfile corresponds to the profile of the *manage Connection* service implemented by *Co1*. 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 the virtual organization, thus *Co1* is registered as a *Communicator*.

There will be another inquiry regarding which services exist within the units. *DirectionUnit*, *RecommendationUnit* and *DeviceUnit* will return the services that are necessary for planning. The SF will again return a list (similar to Table1).

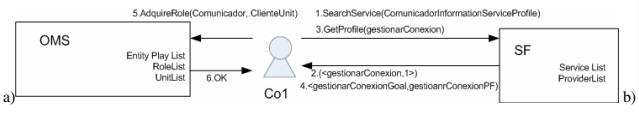
Based on the profiles, we will determine that *Co1* is interested in acquiring the role of *DeviceManager* since in this case it wants to interact with the elements within the environment. *Co1* will use this role to act as intermediary to process the signals that come from the client devices and make them comprehensible to the system. It will allow the order requested by the client from a mobile device to be understood and executed by the specific device that is the object of the request. (`AcquireRole(DeviceUnit, DeviceManager))`.

The agent will now be registered as a member of *DeviceUnit* with the role of *DeviceManager*. This role will require the agent to register the Locate service, associating it with the *process* and *grounding* that it considers to most useful. If this is not done within the allocated time, the agent will be expelled. The actual norm is as follows: `OBLIGED DeviceManager REGISTER Locate(?route) BEFORE deadline SANCTION (OBLIGED OMS SERVE Expulse (?agentID DeviceManager DeviceUnit))`

The agent will be informed of the norm upon carrying out the "AcquireRole", so that it can take it into consideration if it is a normative agent (otherwise ignore it). To avoid external agents assuming the role of *DeviceManager*, the agent registers a new incompatibility norm in the system. This norm will make it impossible for other agents to take on the same role: `RegisterNorm ("norm1", "FORBIDDEN Member REQUEST AcquireRole Message(CONTENT(role 'DeviceManager))")`

The *Pe1* and *Pl1* agents will act in a similar fashion, registering at the end for the corresponding units *ProfileManager* and *Planner*. They too will be required to register the services as indicated by the defined norms. (*GenerateProfile*, *ConsultProfile*, *UpdateProfile*, *MSState, UpdateMSGState, Replan, ValidateRoute, ValueRoute, ShopListRecovery*) Each one is required for generating the optimal route for the user to follow. The *C1* and *C2* agents will request acquiring the *ClientManager* role in order to access the basic services: *FindClient, GenerateProfile, ConsultProfile, UpdateProfile, MSGState, and UpdateMSGState*.

The agents will also consider whether to acquire other system roles that might be necessary for the required functionality. *C1* can request existing services from the SF, and will receive a list with all the agents that offer their services. In this case, for example, *C1* could be interested in offering the *SendPromotion* service as a suggesting sent to the user. These services are offered from the *BusinessUnit*, for which it is necessary to acquire the role of *ProfileManager* `(AcquireRole, (BusinessUnit, ProfileManager))`. The *Entity Play List* would end up as shown in figure 2b.



Entity Play List		
Entity	**Unit**	**Rol**
Co1	ClienteUnit	Comunicador
Co1	DispositivoUnit	GestorDeDispositivos
Pl1	RecomendacionUnit	Planificador
C1	DireccionUnit	GestorDeCliente
C2	DireccionUnit	GestorDeCliente
C1	NegocioUnit	GestorDePerfil

Fig. 2. a)Agent Co1 registering,b)Entity Play List

4 Conclusions

An important issue in the development of real open multi-agent systems 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 MAS 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 organization methods. We propose a methodology that incorporates decomposition and abstraction via the THOMAS architecture for a dynamic MAS environment. This architecture has allowed us to directly model the organization of a shopping center according to a previous basic analysis, to dynamically and openly define the agent roles, functionalities and restrictions, and to obtain beforehand the service management capabilities (discovery, directory, etc.) within the platform. THOMAS provides us with the level of abstraction necessary for the development of our system, and the set of tools that facilitate its development. In THOMAS architecture, agents can transparently offer and invoke services 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 as an illustration of not only the usage of THOMAS components and services, but also of the dynamics of the applications to be developed with this architecture. In this way, examples of THOMAS service calls have been shown through several scenarios, along with the evolution of different dynamic virtual organizations.

References

1. Argente, E., Julian, V., Botti, V.: MAS Modelling based on Organizations. In: 9th Int. Workshop on Agent Oriented Software Engineering, pp. 1–12 (2008)
2. Bajo, J., Corchado, J.M., De Paz, Y., De Paz, J.F., Rodríguez, S., Martín, Q., Abraham, A.: SHOMAS: Intelligent Guidance and Suggestions in Shopping Centres. Applied Soft Computing (2008) ISSN 1568-4946
3. Bernon, C., Cossentino, M., Pavón, J.: Agent Oriented Software Engineering. The Knowledge Engineering Review 20(2), 99–116 (2005)
4. Boissier, O.: Organization Oriented Programming, from closed to open organizations. In: Engineering Societies in the Agents World VI. Springer, Ireland (2007)
5. Carbó, J., Molina, J.M., Dávila, J.: Fuzzy Referral based Cooperation in Social Networks of Agents. AI Communications 18(1), 1–13 (2005)
6. Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., Botti, V.: Service Oriented MAS: An open architecture. In: Actas del AAMAS 2009 (in press, 2009)
7. GTI-IA. An Abstract Architecture for Virtual Organizations: The THOMAS project, `http://www.fipa.org/docs/THOMASarchitecture.pdf`
8. Dignum, V., Dignum, F.: A landscape of agent systems for the real world. Technical report 44-cs-2006- 061, Institute of Information and Computing Sciences, Utrecht (2006)
9. Shekar, S., Nair, P., Helal, A.: iGrocer- A Ubiquitous and Pervasive Smart Grocery Shopping System. In: Proc. ACM SAC, pp. 645–652 (2003)