Computing and Informatics, Vol. 31, 2012, 1001–1020, V 2012-Jul-16

SELF-ORGANIZING MULTI-AGENT SYSTEM FOR MANAGEMENT AND PLANNING SURVEILLANCE ROUTES

Sara Rodríguez, Dante I. Tapia, Juan F. de Paz
Javier Bajo, Juan M. Corchado

Departamento Informática y Automática
Universidad de Salamanca
Salamanca, Spain
e-mail: {srg, dante, fcofds, jbajope, corchado}@usal.es

Ajith Abraham

Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence
WA, USA
&
VSB – Technical University of Ostrava
17. listopadu 15/2172, Ostrava – Poruba, Czech Republic
e-mail: ajith.abraham@ieee.org

Communicated by Patrick Brézillon

Abstract. This paper presents the THOMAS architecture, specially designed to model open multi-agent systems, and its application in the development of a multi-agent system for managing and planning surveillance routes for security personnel. THOMAS uses agents with reasoning and planning capabilities. These agents can perform a dynamic self-organization when they detect changes in the environment. THOMAS is appropriate for developing systems in highly dynamic environments similar to the one presented in this study, as demonstrated by the results obtained after having applied the system to a case study.

Keywords: Open multi-agent systems, dynamic self-organization, planning

Mathematics Subject Classification 2010: 68T42, 68T05, 68M14, 62, 68, 68T01, 68T20, 68T27, 68T30, 68U01

1 INTRODUCTION

Current trends in software development tend to favor features such as autonomy, robustness, flexibility or adaptability. With an increase in the complexity of these applications comes the need to incorporate organizational abstractions that facilitate the design, development and maintenance of the applications. THOMAS (MeTHods, Techniques and Tools for Open Multi-Agent Systems) [4, 1] is a new architecture for open multi-agent systems composed of a set of related models that are appropriate for developing systems in highly dynamic environments.

THOMAS applies the same model, computationally speaking, to human organizations in which both software [15] and human agents adopt diverse roles and interact to achieve both individual and organizational objective and uses the virtual organizations paradigm [23]. This paradigm was conceived as a solution to manage, coordinate and control the performance of the agents [7]. Not only should the organizations be able to describe the structural composition (i.e., functions, agent groups, patterns of interaction, or relationships between roles) and the functional performance (i.e., agent tasks, plans or services), but they should also be able to describe the performance norms for the agents, the dynamic entrance and exit of the components, and the equally dynamic formation of the groups of agents. It is for this reason that the design and development of open MAS is presented as a highly complex task that creates the need to introduce architectures, similar to THOMAS, in which it is possible to regulate and establish what the agents can and cannot do.

The construction sector is one example of an organization that can be modeled through a multi-agent system. According to recent studies [14], 3 % of the workday is lost due to the lack of time control systems that can verify the actual time spent working. Implementing control systems increases productivity since workers can maximize their potential and carry out their tasks more quickly. Incidents resulting from the incorrect supervision of personnel, in particular product theft, access of non-authorized individuals, and the inadequate supervision of equipment, are common problems that increase risks and directly affect the entire process. Telepresence and remote monitoring systems are increasingly common in this type of scenario [13, 17], as they permit supervisors to observe the behavior of the workers and the state of the equipment from a distance [19].

This paper presents a multi-agent system that was developed based on the THOMAS architecture with the goal of improving the supervision of activities carried out by the person in charge of overseeing a construction job. The system monitors the construction personnel and manages the supervision automatically and in execution time. The self-organizing mechanisms provided by THOMAS allow the multi-agent system to react to changes and adapt to new situations. This paper is structured as follows: Section 2 presents the state of the art for open and adaptive MAS systems; Section 3 focuses on the THOMAS architecture; Section 4 demonstrates the development of the MAS for a specific study; finally the results and conclusions obtained are presented.

2 OPEN AND ADAPTIVE MULTI-AGENT SYSTEMS

MAS is a general software technology rooted in fundamental research issues regarding autonomy, cooperation, group formation, etc. It can be classified as open or closed, whereby the fundamental difference is that a closed MAS is created with a fixed structure and objectives, while an open system contains agents that can enter or exit the system dynamically and that have not necessarily been designed to share common objectives. This paper will place particular emphasis on open systems.

2.1 Open Multi-Agent Systems

Open systems [23] exist in dynamic operative environments in which new components can be integrated, or existing components continually abandon the system, and where the actual conditions of operation can change unpredictably. Open systems are characterized by the heterogeneity of their members, limited reliability, conflicting individual objectives, and a high probability of dissatisfaction with the specifications [12]. Numerous research jobs have appeared over the last years, claiming to offer this type of execution framework: Electronic Institutions [10], RICA-J [20], Magentis [12], SIMBA [16].

2.2 Self-Adaptive Multi-Agent Systems

The *Agentlink Technical Forum on Self-Organisation* in MAS [6] defines self-adaptive systems as systems that change their organization without any centralized, explicit, implicit, external or internal control. In self-adaptive systems, reorganization occurs as a result of planning by an internal centralized control. According to the mechanism used, it is possible to distinguish different proposals for the development of these systems. It is possible to find research focused on changing structural aspects, such as the topology of communication between agents [24]; research focusing on system performance based on the interaction produced by the environment [18]; research focused on the ability of agents to dynamically modify their behavior according to some type of reinforcement [22]; research based on the cooperation among agents [3]; or proposals based on meta-models and reference architectures for agent organizations that can be modified and adapted according to the needs of the particular applications [9].

In order to model open and adaptive multi-agent systems, it is necessary to have an infrastructure that can utilize the concept of agent technology in the development process, apply decomposition, abstraction and organization techniques, and keep in mind each of the previous requirements. The methodological proposal applied in this research uses the THOMAS architecture to deal with decomposition and abstraction.

2.3 THOMAS

THOMAS [4, 1] is the name given to an abstract, services-oriented approach architecture that develops large scale, open multi-agent systems and is primarily oriented to the design of virtual organizations. It would be possible to define various areas of research based on the construction of the THOMAS architecture, among which the most relevant are services-oriented computing, structural organization of communities, and agent platforms. The principal components of THOMAS can be seen in Figure 1 (adapted from [1]).

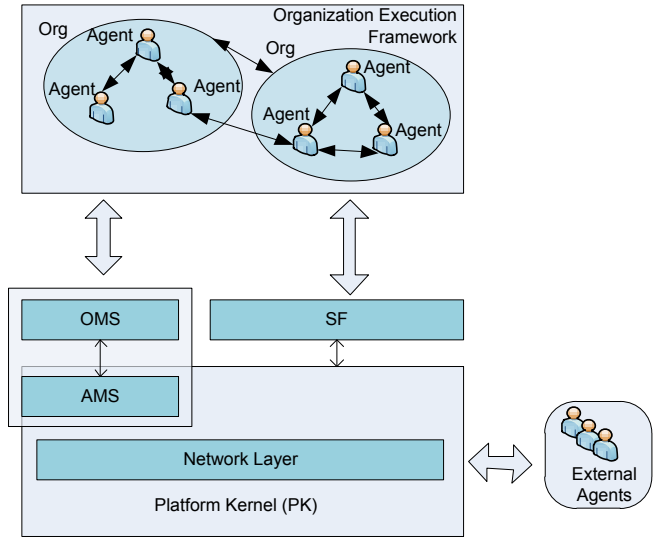

Fig. 1. THOMAS architecture

2.4 Service Facilitator (SF)

This component provides the necessary support for agents and organizations to offer and discover services. The SF primarily provides a place where autonomous entities can record the description of services as directory entries. In other words, its purpose is to function like a yellow pages manager so that it can carry out searches and determine which entities provide any particular service. The SF makes it possible to locate services according to their profile, or to meet objectives by incorporating composition and service mechanisms. The SF controls access to the THOMAS platform by using security techniques and managing authorization. A service is defined as a tuple type (*sID, goal, prof, proc, ground, ont*), where *sID* is the only existing service identifier; *goal* is the objective or aim that the service intends to

achieve, and provides the first level of abstraction for the composition of the service; *prof* is the profile of services, which it describes in terms of *IOPEs (Inputs, outputs, Preconditions and Effects)* and its non-functional attributes.

2.5 Organization Management System (OMS)

The OMS component is primarily responsible for managing the organizations and their entities. It is responsible for the life cycles of the organizations, including the specification and administration of its structural components (roles, units and norms) and its execution components (participating agents and their respective roles; active units at any given time).

- A *role* represents a position in the unit within which it is defined. It is associated with several rules of interaction that are imposed by the unit structure and the specific position it has within the unit; and with performance rules that specify its functionality (types of services that it offers and requires), limit its actions (restrictions, obligations and permission), and determine the consequences of each action (sanctions and compensations).
- A *norm* indicates the obligations, permission and restrictions for each role with respect to registering, requesting and performing services, the composition of those services and the quality of their results. It defines the restrictions that cannot be expressed in the preconditions (or postconditions) of a particular service.
- A *unit* represents groups of agents and permits recursion (units within other units). In doing so, it establishes the topological structure of the system. For example, it facilitates the representation of hierarchical, matrix or coalition structures.

In order to manage these components, OMS handles the following lists:

- *UnitList*: maintains the relationship of existing units, their directly superior units (SuperUnit), objectives and type.
- *RoleList*: maintains the relationship of existing roles within each unit.
- *NormList*: maintains the relationship of the system norms.
- *EntityPlayList*: maintains the relationship for the units in which each agent registers as a member, as well as the role that each agent assumes in the unit.

In THOMAS, a "virtual" unit is defined as representing the "world" for the system in which agents participate by default. OMS creates organizations within this "virtual" unit by registering units that can in turn be composed of more units.

2.6 Platform Kernel (PK)

The PK component handles the basic services for a multi-agent platform. As a result, it is in charge of managing the life cycle of the agents present in the different

organizations, and allows each agent to have a communication channel (incorporating different message transport mechanisms) that facilitates the interaction between the various entities. Additionally, the PK offers a secure connection and provides applications with the necessary mechanisms for multi-device interconnectability, if so required.

The PK services that are required for the THOMAS architecture can be classified into four groups:

1. *Registration*: services necessary for adding, modifying or eliminating native platform agents;
2. *Discovery*: services that provide the functionality for obtaining information about native platform agents;
3. *Management*: services for controlling the state of activation for the native platform agents;
4. *Communication*: services allowing agents to communicate both inside and outside the platform.

The system developed in this paper used the GORMAS (Guidelines for Organization based Multi-Agent Systems) [2]. In order to highlight the advantages of the architecture and the methodologies that were carefully selected, this paper presents a case study in which the application of the architecture and methodologies were modeled in a surveillance environment. The following section details the steps that were followed in the analysis and the design used for this architecture.

3 CASE OF STUDY: MANAGEMENT AND PLANNING OF ROUTES SURVEILLANCE

A multi-agent system based on THOMAS was designed to allow scheduling and distribution of surveillance routes for security guards, and to provide better control of the activities performed by the staff responsible for overseeing industrial environments. The routes are assigned automatically and monitored in real time to ensure that the security guards complete their work shifts. The system interacts with users through a set of mobile devices (PDA's) and wireless communication technologies (Wi-Fi, GPRS and RFID). In this respect, we constructed an open system where the system agents calculate the surveillance routes according to the number of security guards available, the work shifts and the distance to be covered in the facilities. A supervisor defines the areas that must be supervised, which can be modified according the particular scenario or changes in the environment. The system is capable of re-planning the routes automatically taking the number of available security guards into consideration. It is also possible to track the activities for each worker (routes completion) over the Internet.The RFID configuration for the system presented within this paper consists of a mesh of tags distributed throughout the building. Each tag, named "control point" is related to an area that must be covered by the security guards. Each security guard carries a PDA with

a RFID reader to register the completion of each control point. The information is sent via wireless technology to a central computer where it is processed. The basis of the developed multi-agent system presented on this paper includes these features, making it possible to initiate multiple services on demand [21].

3.1 Problem Analysis and Design

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. These include the following:

- *Communicator*: in charge of carrying out the tasks that allow a user to interact with the system. Manages the connections that each user makes.
- *Finder*: in charge of finding a specific user within the system and establishing communication.
- *GuardManager*: in charge of structuring, defining and modeling the profile represented by the guard. The profile is initially established according to the data obtained for each guard, and should be continually updated according to the behavior exhibited by each guard during each shift. It is associated to the PDA for each guard and is in charge of reading the RFID devices at each control point.
- *Planner*: can offer a service for automatically and dynamically generating surveillance routes for security personnel. It will propose the optimal route for each guard to follow according to the time, number of available guards and control points.
- *IncidentManager*: is responsible for managing and providing a solution for each kind of incident that might arise during a security round. It also provides a location service for the guards, and an alarm system management.
- *InformationManager*: is responsible for managing all the information generated by the system (incident, date, hour, control points that are marked or omitted, etc.)
- *DeviceManager*: can interact with the interactive elements within the environment. It deals with devices that use technologies such as RFID, Zigbee, etc.
- *PointControlManager*: this role makes it possible to use monitoring services at the control points through which security personnel must pass.

The structural design of the system is also carried out. First the dimensions are analyzed, and then the most appropriate structural organization [2] is identified. For our case study, this process is modeled as a congregation (SurveillanceUnit) with four units, each of which is dedicated to a type of functionality within the scenario. These four units are:

- *GuardUnit*, which includes the roles associated to a system user: Communicator, Finder and GuardManager.

- *ManagingUnit*, which includes the roles assigned with global management tasks for the management system and route planning: IncidentManager, InformationManager and PointControl Manager.
- *RoutesUnit*, which includes roles related to planning routes for the guards: Planner.
- *DeviceUnit*, which includes the roles associated with device management: DeviceManager.

The diagram of the structural view of the organizational model, adapted according to the congregation pattern, is shown in Figure 2 (the notation is explained in [2]).

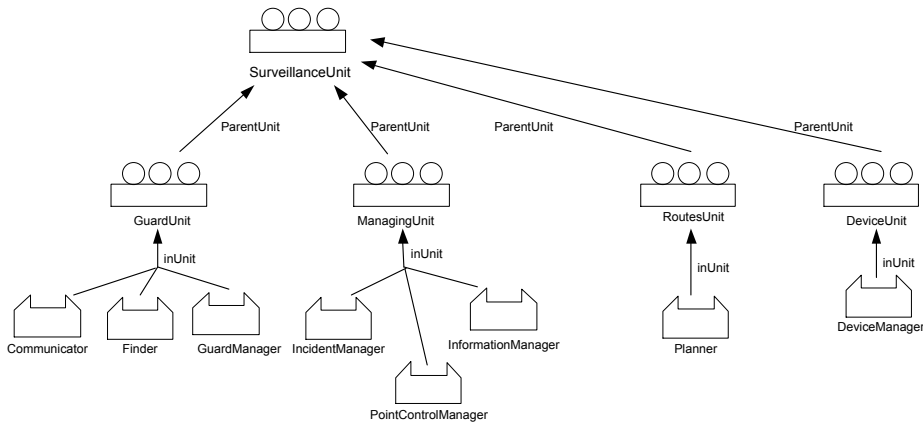

Fig. 2. Diagram of the organization model for the system: structural view

The next step in the analysis and design process consists of detailing the services for each organization unit.

3.1.1 Services

A diagram representing the internal model is created for each unit (*GuardUnit, ManagingUnit, RoutesUnit, DeviceUnit*). These models identify the services associated with each unit. A modeling 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 GuardUnit is *manageConnection*, which is provided by the different types of agents that assume the Communicator role. As

shown in Table 1, the functionality offered by this service makes it possible for the clients to manage their connection to the system.

Service Specification					
Name: manageConnection **Description**: Manage Client Connection **Supplied by**: SF **Required by**: **ClientRole**: GuardManager **ProviderRole**:Comunnicator					
Input Parameters					
Name	**Description**	**Mand.**	**Type**	**Value Range**	**Default**
requestTime	Connection time	Yes	date		
connectionData	Connection Data	Yes	string		
operation	Kind Conecction	Yes	string		
Output Parameters					
Name	**Description**	**Mand.**	**Type**	**Value Range**	**Default**
connection	Connection established	Yes	connection		
Precondition					
–					
Postcondition					
–					

Table 1. *manageConnection* service in *GuardUnit*

3.1.2 Norms

Upon completing the modeling 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. Each possible type of performance of the system is controlled by the norms defined by the following syntax:

```
<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.

A set of norms was defined in our system in order to control the performance within each of the units. This way, for example, an agent acting as *Communicator* within the GuardUnit is required to register *manageConnection* services. 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 GuardUnit))
```

3.1.3 Example of Self-Organization with THOMAS

One of the primary capabilities of the multi-agent system proposed in this paper is the capability of self-organization. We have developed a planning service that has been integrated within the multi-agent system developed [21, 5] and modeled through THOMAS. It involves a self-adapting mechanism that facilitates the automatic assignment of tasks within the multi-agent system. To accomplish this, a case-based reasoning paradigm was used so that a new problem can be resolved based on similar past experiences [5, 21]. The following section demonstrates the sequence of tasks that are executed within the system when a planning service is requested, and how THOMAS can generate the system configuration and ensure the planning takes place. The system evolves by replanning the routes that the guards should follow.

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). As a result, a congregation type *SurveillanceUnit* is created, which represents the organization whose objective it is to control the environment under construction that will be monitored. There are four basic internal units, *GuardUnit, ManagingUnit, RoutesUnit* and *DeviceUnit*, each one dedicated to the functionalities that have been previously noted. The list of system units will remain registerd in the UnitList of the OMS, as shown in Figure 3.

The roles for each unit are defined, and each of their attributes indicated (visibility, position, and heritage), and the name of the unit should also be mentioned. This information will be registered in the OMS *RoleList*. The SF will list the services that are needed for the functionality of the system. The basic services are

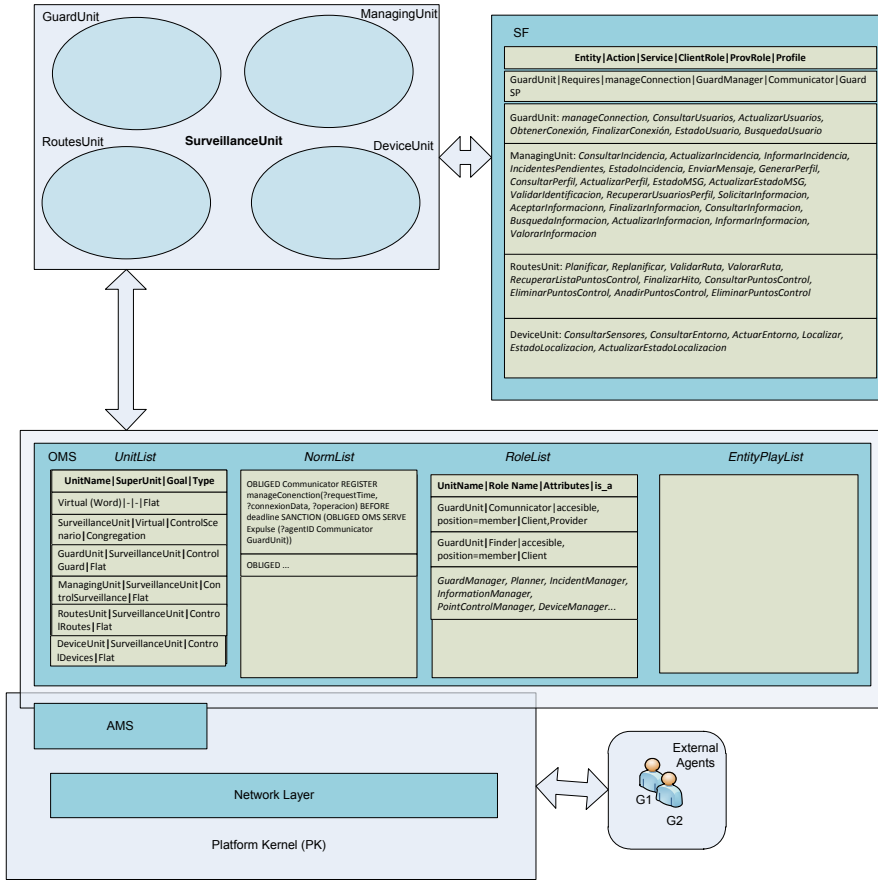

Fig. 3. Architecture of the initial system with an empty framework

those which are essential (as defined by the norms) when the units are being created. 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 guards will use their mobile device to send a request in order to find the most optimal route for them to follow so they can perform their security rounds according to existing conditions (time, control points, etc.).

- Agents *G1* and *G2*: represent the security guards that wish to obtain their route; they will enter the system.

- *Co*, *Pl* and *Pc*: agents that plan to assume the *Communicator, Planner* and *PointControlManager* roles 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, the OMS will play the "member" role in the "world" organization. Asking the SF about which services exist in the system will generate the following answer:

```
GuardUnit Requires manageConnection ClientRole=GuardManager;
ProvRole=Communicator;
```

The *GuardUnit, ManagingUnit, RoutesUnit* 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. *ManagingUnit, RoutesUnit* and *DeviceUnit* will return the services that are necessary for planning. The profiles function will determine that Co is also interested in assuming the *DeviceManager* role since in this case it wants to interact with elements within the environment. Co 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 Localizar(?ruta) 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))")
```

The *Entity Play List* and the units will end up as shown in the Figure 4. 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 security guards. Once the agents are in the system, the organization can evolve according to the plans that are carried out.

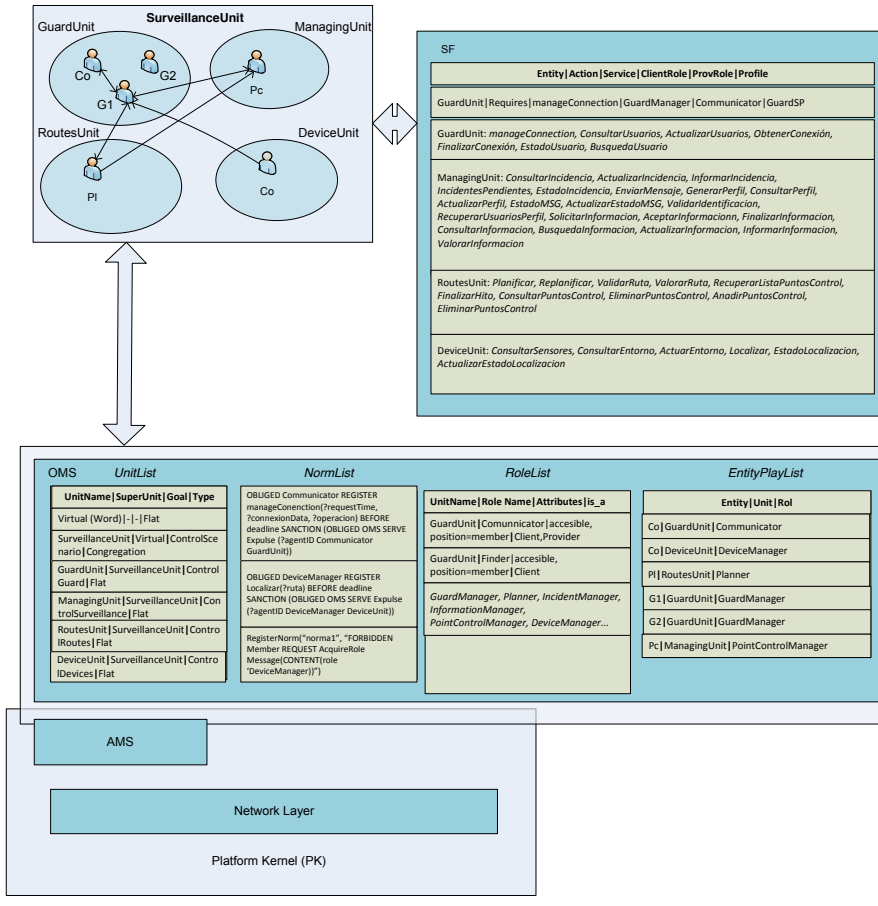

Fig. 4. System architecture with execution framework

The steps that must be followed for planning a route for one of the agents, G1, can be summarized as follows:

- Once inside the THOMAS platform, G1 obtains a connection using the *obtainConnection* service provided by the *Communicator* role that has acquired the Co agent already immersed within the system.
- G1 locates the user's mobile device within the system acting as a client through the *Locate* service provided by Co.
- Likewise, G1 generates its profile using the *GenerateProfile* service and is used by the *PointControlManager* PC agent to know the data for the guard.
- G1 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 Pc to use the *RecoverListPoints* service and obtain the control required to generate the route.
- P1 will provide the route to G1 which in turn will validate the route using the *ValidateRoute* service obtained from P1.
- G1 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 guard's device.

4 RESULTS AND CONCLUSSIONS

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 for these kinds of systems. Traditional multi-agent system development methodologies are not suitable for developing open multi-agent systems because they assume a fixed number of agents that are specified during the system analysis phase. The proposed methodology presented in this paper manages these techniques by using the THOMAS architecture for a multi-agent system in a dynamic environment.

The THOMAS architecture can be compared with other options currently available that can create organizational models, platforms and agent architectures. In our case, the use of THOMAS allowed us to dynamically model and develop concepts for a route planning system, something we could not have been able to achieve with other platforms comparable to THOMAS such as [11, 20, 8, 9], as mentioned in this paper. Specifically, our proposal allowed us to directly model the organization for a security environment 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.

Several tests have been comparing the overall performance of the system with respect to its previous version, the latter having used THOMAS. The tests consisted of a set of requests delivered to the planning mechanism of services which in turn had to generate paths for each security guard. The system presented in this paper was implemented and tested in controlled environments. With each route generator simulation, we noted that the system improved significantly, both the actual case study as well as the architecture agents at the organizational and resource level. Several data have been obtained from these tests, notably the average time to accomplish the plans, the number of crashed agents, and the number of crashed services.

Figure 5 illustrates the improvement that was achieved with regards to the system response times, due primarily to the structural change. 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 surveillance routes, to having an organizational and dynamic agent-based structure

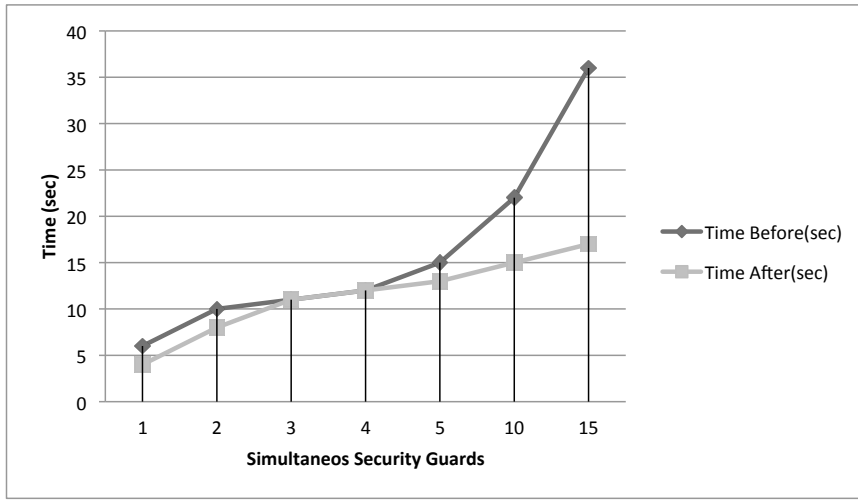

Fig. 5. Time needed for both systems to generate paths for a group of security guards

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 surveillance routes. This reduction was shown to optimize the workday for the security guards, increasing their productivity by 7 % by dynamically assigning the control points, which minimized the area to cover during the surveillance routes.

In addition to having a non-centralized organization, we were able to achieve an improvement in the robustness and scalability of the system. Figure 6 illustrates the results obtained with regards to the number of agents with errors in the system. 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. Additionally, the agents with errors in the new system can be instancized again and assume their role within the organization once more, because of the self-organization 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.

We can conclude that THOMAS was able to provide us with the necessary level of abstraction for developing our system, and the set of tools for facilitating its development. In 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

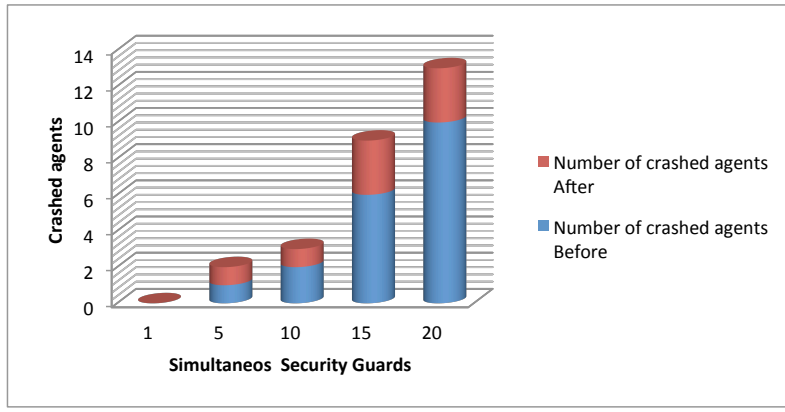

Fig. 6. Number of agents and services crashed for both versions of the system

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.

REFERENCES

[1] Argente, E.—Botti, V.—Carrascosa, C.—Giret, A.—Julian, V.—Rebollo, M.: An Abstract Architecture for Virtual Organizations: The THOMAS project DSIC-II/13/08 (to appear).

[2] Argente, E.—Julian, V.—Botti, V.: MAS Modelling Based on Organizations. Proceedings of 9th Int. Workshop on Agent Oriented Software Engineering – AOSE08, Estoril, Portugal, March 2009, pp. 16–30.

[3] Capera, D.—Georgé, J. P.—Gleizes, M. P.—Glize, P.: Emergence of Organisations, Emergence of Functions. Proceedings of Symposium on Adaptive Agents and Multi-Agent Systems, Wales 2003, pp. 103–108.

[4] Carrascosa, C.—Giret, A.—Julian, V.—Rebollo, M.—Argente, E.—Botti, V.: Service Oriented MAS: An Open Architecture. Proceedings of 8th International Conference on Autonomous Agents and Multiagent Systems – AAMAS 2009, Budapest, Hungary, May 2009, pp. 1291–1292.

[5] Corchado, J. M.—Gonzalez-Bedia, M.—de Paz, Y.—Bajo, J.—de Paz, J. F.: Replanning Mechanism for Deliberative Agents in Dynamic Changing Environments. Computational Intelligence, Vol. 24, 2008, No. 2, pp. 77–107.

[6] di Marzo Serugendo, G.—Gleizes, M.—Karageorgos, A.: AgentLink First Technical Forum Group Self-Organisation in Multi-Agent Systems. AgentLink Newsletter, Vol. 16, 2004, pp. 23–24.

[7] 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 University 2006.

[8] Dignum, V.—Vazquez-Salceda, J.—Dignum, F.: OMNI: Introducing Social Structure, Norms and Ontologies into Agent Organizations. Proceeding of 3rd International Workshop on Programming Multi-Agent Systems, Utretch, The Netherlands 2005, pp. 181–198.

[9] Escriva, M.—Palanca, J.—Aranda,G.—García, A.—Julian, V.—Botti, V.: A Jabber-Based Multi-Agent System Platform. Proceeding of 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, May, 2006, pp. 1282–1284.

[10] Esteva, M.—Rodríguez, J.—Sierra, C.—García, P.—Arcos, J.: On the Formal Specification of Electronic Institutions. Agent Mediated Electronic Commerce 1991 (2001), pp. 126–147.

[11] Giorgini, P.—Kolp, M.—Mylopoulos, J.: Multi-Agent Architectures as Organizational Structures. Journal of AAMAS, Vol. 13, 2003, No. 1, pp. 3–25.

[12] Giret, A.—Botti, V.—Valero, S..: MAS Methodology for HMS. Proceedings of 2nd International Conference on Applications of Holonic and Multi-Agent Systems – HoloMAS 2005, Copenhagen, Denmark, August 2005, pp. 39–49.

[13] Heikkil, T.—Kollingbaum, M.—Valckenaers, P.—Bluemink, G. J.: An Agent Architecture for Manufacturing Control: manAge. Computers in Industry, Vol. 46, 2001, pp. 315–331.

[14] Inology web site. Availaible on: `http://www.controldetiempos.com/`.

[15] Jennings, N. R.—Sycara, K.—Wooldridge, M.: A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, Vol. 1, 1998, pp. 275–306.

[16] Julián, V.—Carrascosa, C.—Rebollo, M.—Soler, J.—Botti,V.: SIMBA: An Approach for Real-Time Multi-Agent Systems. Proceeding of Catalonian Conference on Topics inArtificial Intelligence. Castellon, Spain, Octorber 2002, pp. 282–293.

[17] Lim, M. K.—Zhang, Z.: A Multi-Agent Based Manufacturing Control. Strategy for Responsive Manufacturing. Journal of Materials Processing Technology, Vol. 139, 2003, No. 1-3, pp. 379–384.

[18] Reitbauer, A.—Battino, A.—Karageorgos, A.—Mehandjiev, P.—Valckenaers, P.—Saint-Germain, B.: The MaBE Middleware: Extending Multi-Agent Systems to Enable Open Business Collaboration. Proceedings of 6th International Conf. on Information Techonology for Balanced Automation Systems in Manufacturing and Services – BASYS '04, Vienna, Austria, September 2004, pp. 1–10.

[19] SEOPAN web site. Availaible on: `http://www.seopan.es/`.

[20] Serrano, J. M.—Ossowski, S.: On the Impact of Agent Communication Languages on the Implementation of Agent Systems. Proceeding of Cooperative Information Agents VIII, Erfurt, Germany, September 2004, pp. 92–106.

[21] Tapia, D. I.—de Paz, J. F.—Rodríguez, S.—Corchado, J. M.: Multi-Agent System for Management and Monitoring of Surveillance Routes. IEEE Latin American Transactions, Vol. 6, 2009, No. 6, pp. 494–499.

[22] Weyns, D.—Schelfthout, K.—Holvoet, T.—Glorieux, O.: Role Based Model for Adaptive Agents. Proceedings of 4th Symposium on Adaptive Agents and Multiagent Systems – AISB '04, Leeds, UK 2004, pp. 75–86.

[23] Zambonelli, F.—Jennings, N.—Wooldridge, M.: Developing Multiagent Systems: The GAIA Methodology. ACM Transactions on Software Engineering and Methodology, Vol. 12, 2003, pp. 317–370.

[24] Zambonelli, F.—Gleizes, M. P.—Mamei, M.—Tolksdorf, R.: Spray Computers: Frontiers of Self-Organisation for Pervasive Computing. Proceeding of 1st International Conference on Autonomic Computing, Modena, Italy, June 2003, pp. 97–402.

Sara Rodríguez pursued her studies of Ph. D. in this University. She obtained her Technical Engineering in Systems Computer Sciences degree in 2004, and Engineering in Computer Sciences degree in 2007 from the University of Salamanca. She is Assistant Professor at the University of Salamanca and researcher in the BISITE research group (`http://bisite.usal.es`). She has participated as a co-author of papers published in recognized international conferences and symposiums.

Dante I. Tapia received his Ph. D. in Computer Science from the University of Salamanca (Spain) in 2009. At present, he is a researcher aint the BISITE Research Group of the University of Salamanca (Spain). He obtained an Engineering in Computer Sciences degree in 2001 and an M. Sc. in Telematics from the University of Colima (Mexico) in 2004. He has been involved in the development of automated systems in the Faculty of Telematics at the University of Colima and deeply collaborating with the Government of the State, where he obtained a scholarship to complete his academic formation. He has also been a co-author of papers published in recognized workshops and symposiums.

Juan Francisco de Paz received his Ph. D. in Computer Science from the University of Salamanca (Spain) in 2010. He is Assistant Professor at the University of Salamanca and researcher in the BISITE research group (`http://bisite.usal.es`). He obtained a Technical Engineering in Systems Computer Sciences degree in 2003, Engineering in Computer Sciences degree in 2005 from the University of Salamanca and Statistic degree in 2007 from the same University. He has been co-author of published papers in several journals, workshops and symposiums.

Javier Bajo received his Ph. D. in Computer Science and Artificial Intelligence from the University of Salamanca in 2007. At present, he is Director of the Data Processing Center and Associate Professor at the Pontifical University of Salamanca (Spain) and researcher in the BISITE research group (`http://bisite.usal.es`) at the University of Salamanca (Spain). He obtained his Information Technology degree from the University of Valladolid (Spain) in 2001 and Engineering in Computer Sciences degree from the Pontifical University of Salamanca in 2003. He has been a member of the organizing and scientific committee of several international symposiums such as CAEPIA, IDEAL, HAIS, etc. and is co-author of more than 170 papers published in recognized journals, workshops and symposiums.

Juan M. Corchado received his Ph. D. in Computer Science from the University of Salamanca in 1998 and Ph. D. in Artificial Intelligence from the University of Paisley, Glasgow (UK) in 2000. At present, he is Dean at the Faculty of Computer Sciences, Associate Professor, Director of the Intelligent Information System Group (`http://bisite.usal.es`) and Director of the MSc programs in Computer Science at the University of Salamanca (Spain). Previously, he was sub-director of the Computer Science School at the University of Vigo (Spain, 1999–2000) and a researcher at the University of Paisley (UK, 1995–1998). He has been a research collaborator with the Plymouth Marine Laboratory (UK) since 1993. He has led several artificial intelligence research projects sponsored by Spanish and European public and private sector institutions and has supervised seven Ph.D. students. He is the co-author of over 230 books, book chapters, journal papers, technical reports, etc.

Ajith Abraham received the Ph. D. degree in Computer Science from Monash University, Melbourne, Australia. He is currently the Director of Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, USA, which has members from more than 85 countries. He serves/has served the editorial board of over 50 international journals and has also guest edited 40 special issues on various topics. He has authored/co-authored more than 800 publications, and some of the works have also won best paper awards at international conferences. His research and development experience includes more than 22 years in the industry and academia. He works in a multidisciplinary environment involving machine intelligence, network security, various aspects of networks, e-commerce, Web intelligence, Web services, computational grids, data mining, and their applications to various real-world problems. He has given more than 50 plenary

lectures and conference tutorials in these areas. He is the Chair of IEEE Systems Man and Cybernetics Society Technical Committee on Soft Computing and also a Distinguished Speaker of IEEE Computer Society representing Europe. He is a Senior Member of the IEEE, the IEEE Computer Society, the Institution of Engineering and Technology (U.K.) and the Institution of Engineers Australia (Australia), etc. He is actively involved in the Hybrid Intelligent Systems (HIS); Intelligent Systems Design and Applications (ISDA); Information Assurance and Security (IAS); and Next Generation Web Services Practices (NWeSP) series of international conferences, in addition to other conferences. More information at `http://www.softcomputing.net`.