The THOMAS architecture: A case study in Home Care Scenarios.

Fraile Nieto, J.A.[1], Rodríguez, S. [2], Bajo, J.[1] and Corchado, J.M.[2]

[1]Pontifical University of Salamanca, c/ Compañía 5, 37002 Salamanca, Spain
[2]University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain
{jafraileni, jbajope}@upsa.es, {srg, corchado}@usal.es

Abstract. Nowadays, the need for architectures and computational models for large scale open multi-agent systems is considered a key issue for the success of agent technology in real world scenarios. The main goal of this paper is to describe a case study in Home Care applying an abstract architecture and a computational model for large scale open multi-agent systems based on a service-oriented approach. The architecture we used is THOMAS. THOMAS is specifically addresses to design organizational structures for multiagent systems, in this case, a Home Care system. The paper presents services example for the management of a home dependent environment, which demonstrates the new features of the proposal.

Keywords: Dependent environments, Abstract Arquitectures, Multiagent Systems, Home Care, Organizations, Services

1 Introduction

The continuous growth of the dependency people sector has dramatically increased the need for new home care solutions [1] [7]. Besides, the commitments that have been acquired to meet the needs of this sector, suggest that it is necessary to modernize the current systems. Home Care is one of the objectives of the pervasive computing, and dependent people require new solutions that can make use of the technological advances to provide novel and fundamental services [2]. The vision of the pervasive computing allows improving the quality, access, equity and continuity of health care [2]. In this sense, the intelligent environments can improve health care services and can have a high social impact, especially in the home care services for dependent chronic patients [3]. Home Care requires effective communication as well as distributed problem solving [3].

Multi-agent systems [4], [15], and intelligent devices-based architectures have been recently explored as supervisor systems for health care scenarios [2] for elderly people and for Alzheimer patients [7]. These systems allow providing constant care in the daily life of dependent patients [5], predicting potentially dangerous situations and facilitating a cognitive and physical support for the dependent patient [3].

The goal of work is to present a case study in which the THOMAS (MeTHods, Techniques and Tools for Open Multi-*A*gent Systems) [6] [9] architecture is used to build an open MAS for supervising and monitoring dependent patients at home. 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 volatile environments. THOMAS provides a high level of abstraction to determine which components are necessary for addressing all of the needs and characteristics of a home care environment. The multi-agent system developed offers a series of functionalities including automatic reasoning and planning mechanism for scheduling the medical staff working day, an alert system, a location and tracking system and an identification system. The medical staffs has been provided with PDAs and mobile phones, as well as with Java Card tags, and the home environments have been equipped with presence detection sensors, access control mechanisms, door opening devices and video cameras. The multi-agent system monitors the daily routine of the patient and detects dangerous situations. If any anomalous situation is detected, alert system is used to obtain medical assistance.

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" . 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 are incorporated in THOMAS via the open MAS and virtual organization paradigm, which was conceived as a solution for the management, coordination and control of agent performance. The organizations not only find the structural composition of agents (i.e., functions, relationships between roles) and their functional behaviour (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.

The rest of the paper is structured as follows: section 2 provides an analysis of related studies; section 3 presents the proposed architecture model; section 4 shows an example of an implementation, highlighting the new possibilities provided by this type of architecture and specifically presents an approach for a home care management; finally, some conclusions of work are shown in section 5.

2 Related works

Agents and multi-agent systems in dependency environments are becoming a reality, especially in health care. Most agents-based applications are related to the use of this technology in the monitoring of patients, treatment supervision and data mining. Lanzola present a methodology [10] that facilitates the development of interoperable intelligent software agents for medical applications, and propose a generic computational model for implementing them. The model may be specialized in order to support all the different information and knowledge-related requirements of a hospital information system. Meunier proposes [11] the use of virtual machines to support mobile software agents by using a functional programming paradigm. This virtual machine provides the application developer with a rich and robust platform upon which to develop distributed mobile agent applications, specifically when targeting distributed medical information and distributed image processing. While an interesting proposal, it is not viable due to the security reasons that affect mobile agents, and there is no defined alternative for locating patients or generating planning strategies. There are also agents-based systems that help patients to get the best possible treatment, and that remind the patient about follow-up tests [12]. They assist the patient in managing continuing ambulatory conditions (chronic problems). They also provide health-related information by allowing the patient to interact with the on-line health care information network. Decker & Li propose [8] a system to increase hospital efficiency by using global planning and scheduling techniques. They propose a multi-agent solution that uses the generalized partial global planning approach which preserves the existing human organization and authority structures, while providing better system-level performance (increased hospital unit throughput and decreased impatient length of stay time). To do this, they use resource constraint scheduling to extend the proposed planning method with a coordination mechanism that handles mutually exclusive resource relationships. Other applications focus on home scenarios to provide assistance to elderly and dependent persons. RoboCare presents a multi-agent approach that covers several research areas, such as intelligent agents, visualization tools, robotics, and data analysis techniques to support people with their daily life activities [13]. TeleCARE is another application that makes use of mobile agents and a generic platform in order to provide remote services and automate an entire home scenario for elderly people [4].

The architecture we used is THOMAS (MeTHods, techniques and tools for Open Multi-Agent Systems) [6] [9], which 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 the main characteristics of THOMAS as well as the results obtained after having applied the system to a case study..

3 THOMAS Architecture Model

THOMAS architecture basically consists of a set of modular services. Though THOMAS feeds initially on the FIPA[1] architecture, it expands its capabilities to deal with organizations, and to boost its services abilities. In this way, a new module in charge of managing organizations has been introduced into the architecture, along with a redefinition of the FIPA Directory Facilitator that is able to deal with services in a more elaborated way, following Service Oriented Architectures guidelines. As has been stated before, services are very important in this architecture. In fact, agents have access to the THOMAS infrastructure through a range of services included on different modules or components. The main components of THOMAS are the following [9]:

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

- Service Facilitator (SF), this component offers simple and complex services to the active agents and organizations. Basically, its functionality is like a yellow page service and a service descriptor in charge of providing a green page service. The SF acts as a gateway to access the THOMAS platform. It manages this access transparently, by means of security techniques and access rights management. The SF can find services searching for a given service profile or searching for the goals that can be fulfilled when executing the service. This is done using the matchmaking [14] and service composition mechanisms [7] which are provided by the SF. The SF also acts as a yellow pages manager and in this way it can find which entities provide a given service.
- Organization Management System (OMS), mainly responsible for the management of the organizations and their entities. Thus, it allows the creation and management of any organization. The OMS is in charge of organization life-cycle management, including specification and administration of both their structural components (roles, units and norms) and their execution components (participant agents and roles they play, and active organizational units). Organizations are structured by means of organizational units, which represent groups of entities (agents or other units), which are related in order to pursue a common goal. These organizational units have an internal topology (i.e. hierarchical, team, plain), which imposes restrictions on agent relationships and control (ex. supervision or information relationships).
- Platform Kernel (PK), it maintains basic management services for an agent platform. The PK is in charge of providing the usual services required in a multi-agent platform. Therefore, it is responsible for managing the life-cycle of the agents included in the different organizations, and it also makes it possible to have a communication channel (incorporating several message transport mechanisms) to facilitate interaction among entities. On the other hand, the PK provides safe connectivity and the mechanisms necessary for allowing multi-device interconnectivity.

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.

4 Applying THOMAS to Home Care

The Home Care example is an application that facilitates the interconnection between dependent people and their environment and medical staff (doctors, nurses and personal assistant), delimiting services that each one can request or offer. The system controls which services must be provided by each agent. 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. Below, a description of the structure elements of the Home Care organization is detailed. Then, in section 4.2, a dynamical usage of the organization is explained, providing different execution scenarios.

4.1 Case Study Organization Structure

This case study is modelled as an organization (HomeCare) within which there are three organizational units (HCServiceUnit, LocationUnit and AlertUnit) each of which represents a group of agents. Each unit is dedicated to home care services, location services or alert services, respectively.

Four kinds of roles can interact in the Home Care example: patient, doctor, family and provider roles. The *Patient role* requests system services. More specifically, it can request home automation services, through the alert service communication with the medical service or the family and more services in their home. The *Doctor role* is specialized in three subroles according to communication with each unit (HCServiceDoctor, LocationDoctor and AlertDoctor). The *Provider role* is in charge of performing services. A provider agent offers home automation, location or alert search services. The provider role is also specialized into HCServiceProvider, LocationProvider and AlertProvider. Finally, the *Family role* provides the advances consultation service. It represents the family in which relatives can check the patient status. As it is a private role, agents are not able to acquire this Family role. Figure 1 shows the Home Care structure, with its organizations/units, roles and relationships with each other.

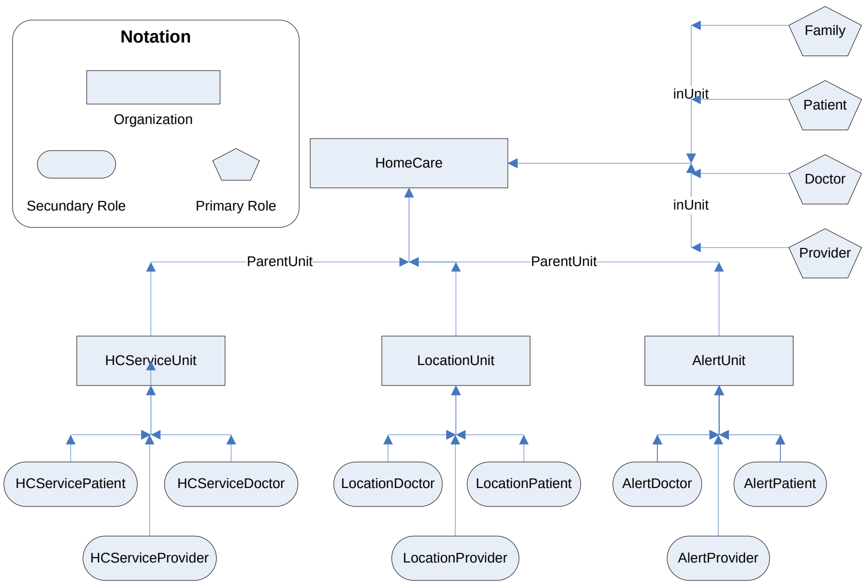

Fig. 1. Home Care structure (units and roles).

The HomeCare organization offers three services: Automation, Location and Alert service. These services are specialized for each unit. A brief description of the profiles of all these services is shown in Table 1.

Table 1. Service Profiles for the HomeCare system.

Profiles of HCServiceUnit		
Service: OnOffLight **UnitID**: HCServiceUnit **Inputs**: idlight: string operation: string	**ProfileID**: OnOffLightPF **ClientRole**: HCServicePatient **Outputs**: [light ok] idlight: string state: string	**Description**: On or off a light. **ProviderRole**: HCServiceProvider **Outputs**: [not ok light] error
Service: LockUnlockAccess **UnitID**: HCServiceUnit **Inputs**: idaccess: string operation: string	**ProfileID**: LockUnlockAccessPF **ClientRole**: HCServicePatient **Outputs**: [access ok] idaccess: string state: string	**Description**: Lock or unlock a access. **ProviderRole**: HCServiceProvider **Outputs**: [not ok acces] error
Profiles of LocationUnit		
Service: SearchPatient **UnitID**: LocationUnit **Inputs**: idhome: string idpatient: string	**ProfileID**: SearchPatientPF **ClientRole**: LocationProvider, LocationDoctor, Family **Outputs**: [patient ok] name: string location: string	**Description**: Search for a patient in their home. **ProviderRole**: LocationProvider **Outputs**: [not in home] error
Service: IdentifyPatient **UnitID**: LocationUnit **Inputs**: idpatient: string	**ProfileID**: SearchPatientPF **ClientRole**: LocationProvider **Outputs**: [patient ok] location: string date: time	**Description**: Identify a patient. **ProviderRole**: LocationProvider **Outputs**: [not ok patient] error
Service: addServPatient **UnitID**: LocationUnit **Inputs**: idpatient: string operation: string	**ProfileID**: AddServPatientPF **ClientRole**: LocationProvider **Outputs**: [patient ok] location: string date: time	**Description**: Add a patient service. **ProviderRole**: LocationProvider **Outputs**: [not ok add patient] error
Profiles of AlertUnit		
Service: SendSms **UnitID**: AlertUnit **Inputs**: sms: string phone: string	**ProfileID**: SendSmsPF **ClientRole**: AlertProvider, AlertDoctor, Family, AlertPatient **Outputs**: [phone ok] idsms: string state: string	**Description**: Send a SMS. **ProviderRole**: AlertProvider **Outputs**: [not ok phone] error
Service: ProcessSms **UnitID**: AlertUnit **Inputs**: sms: string phone: string	**ProfileID**: ProcessSmsPF **ClientRole**: AlertProvider, AlertDoctor, Family, AlertPatient **Outputs**: [sms ok] sms: string phone: string	**Description**: Process a SMS. **ProviderRole**: AlertProvider **Outputs**: [not ok sms] error

All these services have been registered in the SF component of the THOMAS platform. In this example, we have assumed that the Home Care system does not initially have any agent registered as a service provider, nor any agent acting as a patient and nor any agent acting as a doctor. Therefore, this system has initially only been structured as a regulated space in which agents might enter to provide or request all of those specific services registered in the SF component. Consequently, in the initial state of the system, there is no provider attached to the HomeCare services.

In the following section, different scenarios are considered, in which patient and/or provider and/or doctor agents enter and participate in the system.

4.2 System Dynamics

In this section, the use of THOMAS meta-services in the HomeCare example is detailed. System dynamics are shown through the specification of different scenarios: (i) a Patient is registered; (ii) the patient is registered as a PatientLocation; (iii) new services patients are included; (iv) a doctor is registered; (v) some services are requested; (vi) malicious agents are expulsed; and (vii) a new unit is created.

4.2.1 Patient registering

In this scenario, the process for registering a new Patient is detailed (Fig 2). Once HC1 has been registered as a member of the THOMAS platform, it asks SF which defined services have a profile similar to its own “home care service”. This request is carried out using the SF *SearchService* (Fig 2, message 1), in which *HomeCareServiceProfile* corresponds to the profile of the patient search service implemented by HC1.

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 HC1 executes *GetProfile* (message 3) in order to obtain detailed information about the *OpenCloseDoor* service. Service outputs are “service goal” and “profile” (message 4). The *OpenCloseDoor* profile specifies that service providers have to play a *Patient* role within *HCService*. Thus, HC1 requests from the OMS the *AcquireRole* service to acquire this patient role (message 5). *AcquireRole* service is carried out successfully (message 6), because *HCService* is accessible from *Virtual* organization, thus HC1 is registered as a *Patient*.

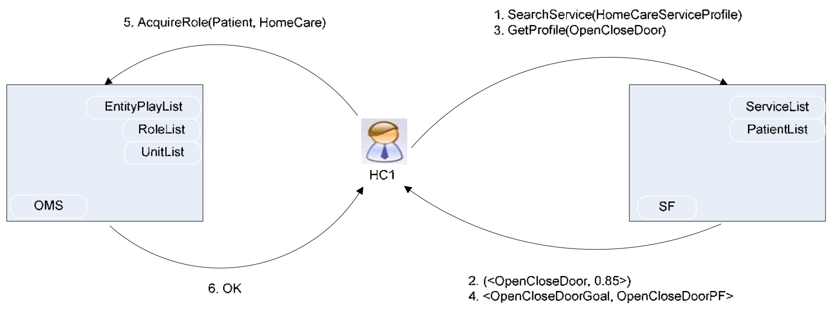
5. AcquireRole(Patient, HomeCare)
1. SearchService(HomeCareServiceProfile)
3. GetProfile(OpenCloseDoor)
EntityPlayList
RoleList
UnitList
OMS
HC1
ServiceList
PatientList
SF
6. OK
2. (<OpenCloseDoor, 0.85>)
4. <OpenCloseDoorGoal, OpenCloseDoorPF>

Fig. 2. Example of patient registering.

4.2.2 LocationPatient registering

Once the "patient registering" process has been detailed, the registration of a location patient is illustrated (Fig 3). HC1 is able to provide a search patient in the home care domain. Therefore, it asks SF whether an available service description with a closer profile exists, requesting *SearchService* from SF as before (Fig 3, message 1).

In this case, SF returns both *SearchPatient* and *IdentifyPatient* since these two services are visible within *HomeCare* unit. As indicated in the service result, *IdentifyPatient* service is more appropriate for HC1 functionality. Therefore, HC1 requests information about this service from SF, using *GetProfile* (message 3). The *IdentifyPatient* profile returned (message 4) specifies that service providers must play *LocationPatients* within *LocationUnit*. Then HC1 requests OMS to adopt *LocationPatient* role (message 5). *AcquireRole* service is carried out successfully (message 6), so HC1 agent is registered as a *LocationProvider*.

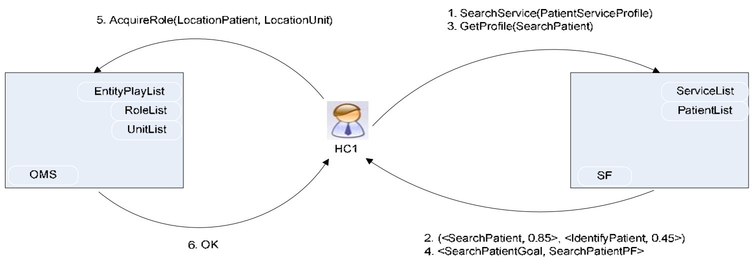
5. AcquireRole(LocationPatient, LocationUnit)
1. SearchService(PatientServiceProfile)
3. GetProfile(SearchPatient)
EntityPlayList
RoleList
UnitList
OMS
HC1
ServiceList
PatientList
SF
6. OK
2. (<SearchPatient, 0.85>, <IdentifyPatient, 0.45>)
4. <SearchPatientGoal, SearchPatientPF>

Fig. 3. Example of LocationPatient registering.

4.2.3 Adding new service patient

This section exemplifies how HC2 has already adopted the *LocationPatient* role and HC1 has been registered as a provider of the *SearchPatient* service. HC2 initially asks what the registered implementations of *SearchPatient* service are (Fig 4, message 3). SF provides a list that contains service implementations details (message 4). HC2 decides to employ the same service process as HC1, so it uses *AddServPatient* service in order to request its inclusion as a provider of *SearchPatient* service (Figure 4, message 5).

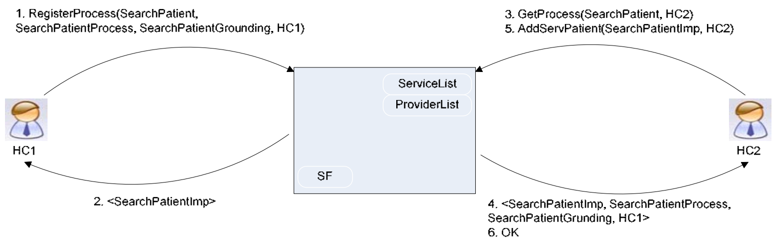
1. RegisterProcess(SearchPatient, SearchPatientProcess, SearchPatientGrounding, HC1)
3. GetProcess(SearchPatient, HC2)
5. AddServPatient(SearchPatientImp, HC2)
ServiceList
ProviderList
SF
HC1
HC2
2. <SearchPatientImp>
4. <SearchPatientImp, SearchPatientProcess, SearchPatientGrunding, HC1>
6. OK

Fig. 4. Example of service implementation and patient registering.

4.2.4 Doctor registering

The following scenario shows the set of service calls for registering new agents as service doctors within the *HomeCare* (Fig 5). A new doctor agent D1, which has already been registered in the THOMAS platform, requests *SearchService* from SF (message 1). As a result, D1 obtains *SearchPatient* service identifier and ranking value (message 2). The *Ranking value* are calculate automatically by the SF. *Ranking value* indicates the degree of suitability between a service and a specified service purpose. Then, D1 employs *GetProfile* (message 3), which specifies that service doctor must play *Doctor* role within *HomeCare* (message 4). Therefore, D1 must acquire *Doctor* role to demand this service (messages 5 and 6). Once D1 plays this doctor role, it employs *GetProcess* service in order to find out who the service providers are and how this service can be requested (message 7). However, there are no providers for the general SearchPatient service (message 8).

Within the *HomeCare* unit, D1 requests *SearchService* again (message 9). In this case, SF returns *IdentifyPatient* services because both services are accessible from *HomeCare* organization. D1 demands the profile of *IdentifyPatient* service (using *GetProfile*, message 11), since this service is more appropriate for its needs. Taking the *IdentifyPatient* profile into account (message 12), D1 requests the adoption of *LocationDoctor* role within *LocationUnit* (message13).

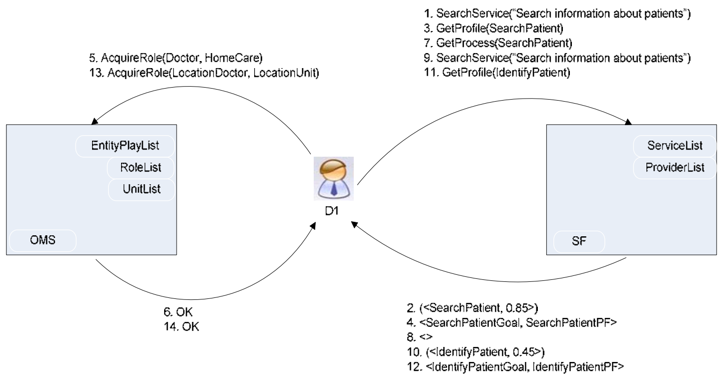
1. SearchService("Search information about patients")
3. GetProfile(SearchPatient)
7. GetProcess(SearchPatient)
9. SearchService("Search information about patients")
11. GetProfile(IdentifyPatient)
5. AcquireRole(Doctor, HomeCare)
13. AcquireRole(LocationDoctor, LocationUnit)
EntityPlayList
RoleList
UnitList
OMS
D1
ServiceList
ProviderList
SF
6. OK
14. OK
2. (<SearchPatient, 0.85>)
4. <SearchPatientGoal, SearchPatientPF>
8. <>
10. (<IdentifyPatient, 0.45>)
12. <IdentifyPatientGoal, IdentifyPatientPF>

Fig. 5. Example of doctor registering.

4.2.5 Service requesting

This scenario shows how doctor agents make demands for services (Fig 6). Once D1 adopts the doctor role for *SearchPatient* service, it is allowed to demand services from providers. Assuming that D1 wants to make an information search about patients, it should use *GetProcess* service to obtain the implementations of available services and also its provider identifiers (message 1).

An implementation of *SearchPatient* has previously been registered by HC1 and HC2. After comparing providers of *SearchPatient* service returned in message 2, D1 chooses to make a service request from HC1 agent (message 3).

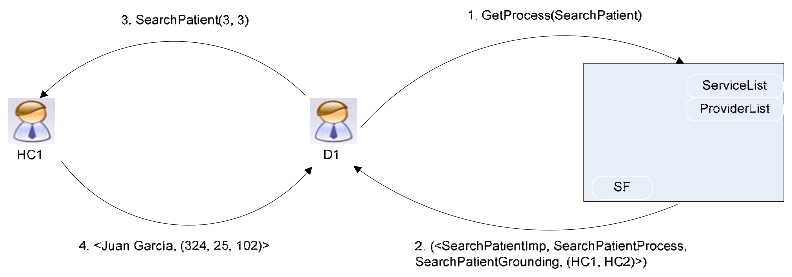
3. SearchPatient(3, 3)
1. GetProcess(SearchPatient)
ServiceList
ProviderList
HC1
D1
SF
4. <Juan Garcia, (324, 25, 102)>
2. (<SearchPatientImp, SearchPatientProcess, SearchPatientGrounding, (HC1, HC2)>)

Fig. 6. Example of service requesting.

4.2.6 Agent expulsion

In this scenario, the expulsion of a malicious agent is carried out (Fig7). *Provider* agent detects that different doctor agents (D1 and D2) have registered with the same identifier number. It consults its database and determines that D2 has been employing an identifier number that does not belong to it. D2 is punished for its fraudulent behaviour and is expelled from *HomeCare*. *Provider* requests the expulsion of D2 from OMS employing *Expulse* service (message 1).

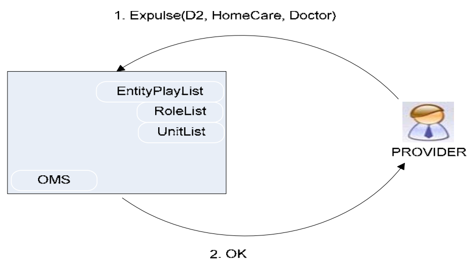

Fig. 7. Example of agent expulsion.

4.2.7 Unit creation

This last scenario illustrates the creation of new units within *HomeCare* (Fig 8). Agent L1 represents a luxury home care company which specializes in luxury services. It is interested in providing information and services very luxurious. This L1 has already adopted the *Provider* role within *HomeCare* unit. However, since the services offered within *LocationUnit* and *AlertUnit* are specialized in location and alert domains, L1 decides to create a new unit (*LuxuryUnit*) within *HomeCare* (Fig 8, message 1). This new unit will be focused on luxury home care. Once the OMS informs L1 about the successful creation of the new unit, L1 defines luxury specific roles and services (messages 3 to 6). Finally, luxury agents would be able to adopt the *LuxuryProvider* role and start offering services to patient agents.

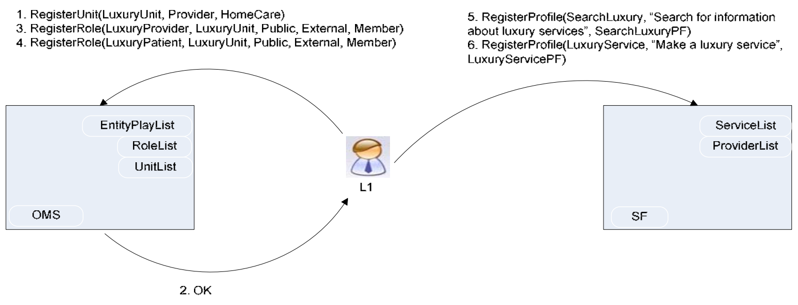
1. RegisterUnit(LuxuryUnit, Provider, HomeCare)
3. RegisterRole(LuxuryProvider, LuxuryUnit, Public, External, Member)
4. RegisterRole(LuxuryPatient, LuxuryUnit, Public, External, Member)
5. RegisterProfile(SearchLuxury, "Search for information about luxury services", SearchLuxuryPF)
6. RegisterProfile(LuxuryService, "Make a luxury service", LuxuryServicePF)
EntityPlayList
RoleList
UnitList
OMS
L1
ServiceList
ProviderList
SF
2. OK

Fig. 8. Example of new unit creation.

After all these scenarios, several agents have joined the THOMAS platform and offer or request services within this system. Table 2 shows the evolution of the *EntityPlayList* content, in which all of the new elements and relationships included due to the execution of these scenarios are emphasized.

Table 2. Final content of OMS internal lists after execution of all scenarios.

EntityPlayList		
Entity	**Unit**	**Role**
Doctor	HomeCare	Doctor
HC1	LocationUnit	LocationPatient
HC2	LocationUnit	LocationPatient
D1	LocationUnit	LocationDoctor
D2	HomeCare	Doctor
L1	LuxuryUnit	LuxuryPatient

5 Conclusions

In the development of real open multi-agent systems, it becomes necessary to have methods, tools and appropriate architectures that can use the concept of agent technology in the development process, and apply decomposition, abstraction and reorganization methods. The THOMAS architecture has allowed us to directly model the organization of a home care environment 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. Moreover, the proposal aims to instigate the total integration of two promising technologies, that is, multi-agent systems and service-oriented computing. In THOMAS architecture, agents can offer and invoke services in a transparent way from other agents, virtual organizations or entities, plus external entities can interact with agents through the use of the services offered.

A case study example has been applied to home care to illustrate the usage of THOMAS components and services. Also the dynamics applications are developed with such architecture. In this way, examples of THOMAS service calls have been shown through several scenarios, along with the evolution of different dynamic virtual organizations. THOMAS creates a multi-agent system that facilitates the development of intelligent distributed systems and renders services to dependent person in home care environments by automating certain supervision tasks.

References

1. Anastasopoulos, M., Niebuhr, D., Bartelt, C., Koch, J. & Rausch, A. (2005). Towards a Reference Middleware Architecture for Ambient Intelligence Systems. ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications.
2. Angulo, C., & Tellez, R. (2004). Distributed Intelligence for smart home appliances. Tendencias de la minería de datos en España. Red Española de Minería de Datos. Barcelona, España.
3. Augusto, Juan C., & McCullagh, Paul. Ambient Intelligence: Concepts and Applications. Invited Paper by the International Journal on Computer Science and Information Systems, volume 4, Number 1, pp. 1-28, June 2007.
4. Camarinha-Matos, L. Afsarmanesh, H. TeleCARE: Collaborative Virtual Elderly Care Support Communities. The journal on Information Technology in Healthcare 2004: 2(2): pp. 73-86.
5. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M. and Botti, V. Hybrid multi-agent architecture as a real-time problem-solving model. Expert Systems With Applications. Volumen 34. Numero 1. pp. 2-17. Elsevier.ISSN:0957-4174 (2008).
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. Corchado, J.M. y Laza, R. (2003). Constructing Deliberative Agents with Case-based Reasoning Technology. International Journal of Intelligent Systems, 18, 1227-1241.
8. Decker, K. and Li, J. (1998). Coordinated hospital patient scheduling. In Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98) (pp. 104-111). IEEE Computer Society.
9. GTI_IA. A. Giret, V. Julian, M. Rebollo, E. Argente, C. Carrascosa and V. Botti. An Open Architecture for Service-Oriented Virtual Organizations. Seventh international Workshop on Programming Multi-Agent Systems. PROMAS 2009. pp. 23-33. 2009
10. Lanzola, G., Gatti, L., Falasconi, S. and Stefanelli, M. (1999). A Framework for Building Cooperative Software Agents in Medical Applications. Artificial Intelligence in Medicine, 16(3), 223-249.
11. Meunier, J. A. (1999). A Virtual Machine for a Functional Mobile Agent Architecture Supporting Distributed Medical Information. In Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems (CBMS'99). IEEE Computer Society, Wahington, DC.
12. Miksch, S., Cheng, K. and Hayes-Roth, B. (1997). An intelligent assistant for patient health care. In Proceedings of the 1st international Conference on Autonomous Agents (AGENTS'97) (pp. 458-465). California, USA: ACM, New York.
13. Pecora, F. and Cesta, A. (2007). Dcop for smart homes: A case study. Computational Intelligence, 23 (4), 395-419.
14. Sycara K,Widoffand S, Klusch M, Lu J (1982) Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Journal on Autonomous Agents and Multi-Agent Systems.
15. Want, R., Pering, T., Borriello, G., and Farkas, K. Disapearing Hardware. Pervasive Computing, 1(1), (2002).