OVACARE: A Multi-Agent System for Assistance and Health Care

Juan F. De Paz, Sara Rodríguez, Javier Bajo, Juan M. Corchado, and Emilio S. Corchado

University of Salamanca, (Spain)
{fcofds,srg,jbajope,corchado,escorchado}@usal.es

Abstract. This paper presents a case study in which the OVAMAH architecture is applied in order to obtain a multi-agent system (MAS) that can provide assistance and health care for Alzheimer patients. The system makes use of several context-aware technologies that allow it to automatically obtain information from users and the environment in an evenly distributed way, focusing on the characteristics of ubiquity, awareness, intelligence, mobility, etc., all of which are concepts defined by Ambient Intelligence. OVAMAH is made up of a group of related modules that are well-suited for developing systems in other highly volatile environments similar to these. 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.

Keywords: Multi-Agent Systems, Ambient Intelligence, Virtual Organizations.

1 Introduction

This study presents a dependable solution for using a novel architecture in designing and building a system for assistance and health care for Alzheimer patients. The importance of developing new and more reliable ways of providing care and support for the elderly is underscored by this trend, and the creation of secure, unobtrusive and adaptable environments for monitoring and optimizing health care will become vital. Some authors [14] consider that tomorrow's health care institutions will be equipped with intelligent systems capable of interacting with humans. Multi-agent systems and architectures based on intelligent devices have recently been explored as supervision systems for medical care for dependent people. These intelligent systems aim to support patients in all aspects of daily life [7], predicting potential hazardous situations and delivering physical and cognitive support.

Ambient Intelligence based systems aim to improve quality of life, offering more efficient and easy ways to use services and communication tools to interact with other people, systems and environments. Among the general population, those most likely to benefit from the development of these systems are the elderly and dependent persons, whose daily lives, with particular regard to health care, will be most enhanced [8]. Dependent persons can suffer from degenerative diseases, dementia, or loss of cognitive ability.

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. [13] present a methodology that facilitates the development of interoperable intelligent software agents for medical applications. [9] propose a system to increase hospital efficiency by using global planning and scheduling techniques.

ALZ-MAS (*ALZheimer Multi-Agent System*)[8] is a distributed multi-agent system designed upon Ambient Intelligence and aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. The main functionalities in the system include reasoning and planning mechanisms [10] that are embedded into deliberative BDI agents, and the use of several context-aware technologies to acquire information from users and their environment.

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 [12]. The organizations not only find 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 OVAMAH (*Adaptive Virtual Organizations: Mechanisms, Architectures and Tools*) architecture is used to build an open MAS for assistance and health care for Alzheimer patients. OVAMAH is the evolution of architecture THOMAS (*MeTHods, techniques and tools for Open Multi-Agent Systems*) [6][11]. 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 health-care 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 health care system), 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 OVAMAH Outline

The architecture we used is OVAMAH (*Adaptive Virtual Organizations: Mechanisms, Architectures and Tools*). OVAMAH is based on THOMAS [6][11]; as such it is in THOMAS where it was made the necessary modifications so that the system can be used as a model for adaptive virtual organizations. It 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. It is based on a services oriented approach and primarily focuses on the design of virtual organizations. 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 assistance system. The architecture is basically formed by a set of services that are modularly structured. It uses the FIPA[1] architecture, expanding its capabilities with respect to the design of the organization, while also expanding the services capacity. OVAMAH 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. The architecture 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 OVAMAH is defined to represent the "world" for the system in 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 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: Ambient Intelligence Based Multi-Agent System

ALZ-MAS [8] is a distributed multi-agent system designed upon Ambient Intelligence and aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. In the remainder of this section, the main characteristics

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

of ALZ-MAS are described, followed by a description of the new ALZ-MAS system developed by means of the OVAMAH architecture. ALZ-MAS structure had five different deliberative agents based on the BDI model (BDI Agents), each one with specific roles and capabilities. The description of the functionality of these agents can be seen in [8]. The MAS was implementing into a geriatric residence to improve health care of the patients. Within the requirements of the problem are: the patient's personal data and behaviour (monitoring, location, daily tasks, and anomalies); the doctors, which treats patients; the nurse schedules, i.e., the nurse's working day obtaining dynamic plans depending on the tasks needed for each assigned patient; the patients' location and manages locks and alarms; and finally, the medical record database and the doctor-patient and nurse-patient assignment.

In previous versions of ALZ-MAS [8], each agent integrates its own functionalities into their structure. If an agent needs to perform a task which involves another agent, it must communicate with that agent to request it. So, if the agent is disengaged, all its functionalities will be unavailable to the rest of agents.

In the version of ALZ-MAS presented in this paper, called OVACARE (*Adaptive Virtual Organizations for Assistance and Health Care*), these mechanisms have been modelled as services in an open agent organization, so any agent can make use of them. The entire ALZ-MAS structure has been modified according to the OVAMAH model, separating most of the agents' functionalities and roles from those to be modelled as services.

OVACARE is a multi-agent organization-based system. The system was designed according to the findings in [1], which apply a MDD (Model Driven Development) focus on organization-oriented methodologies. It is possible to design an organization that is unified, intuitive, and has a high level of abstraction. Given these features, it becomes easier and simpler to design a MAS for organizations and obtain a model for a virtual organization that can be implemented on different platforms. The fundamental idea is to create different models for different levels of abstraction, and then combine them to achieve a full implementation. We used GORMAS (*Guidelines for Organization-based MultiAgent Systems*) [2] as the design methodology and OAVAMAH as the final platform design.

GORMAS is a guide methodology for the design of open MAS from the perspective of human organizations. It includes an analysis phase, a structural organization design phase, and a dynamic organization design phase. Following the guidelines indicated in the methodological guide from [2], one of 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.

User: in charge of managing the users' personal data and behaviour (monitoring, location, daily tasks, and anomalies). User Agent maintains continuous communication with the rest of the system agents, especially with the ScheduleUser Agent (through which the scheduled-users can communicate the result of their assigned tasks) and with the SuperUser Agent. The User Agent must ensure that all the actions indicated by the SuperUser are carried out, and sends a copy of its memory base (goals and plans) to the Admin Agent in order to maintain backups.

SuperUser: in charge of inserting new tasks into the Admin Agent to be processed by a Case-Based Reasoning mechanism. It also needs to interact with the User Agents to impose new tasks and receive periodic reports, and with the ScheduleUser Agents to ascertain the evolution of each plan. There is one agent for each doctor connected to the system.

Scheduler: in charge of managing a Case-Based Planning (CBP) mechanism embedded in its structure. It schedules the users' daily activities and obtains dynamic plans depending on the tasks needed for each user. It manages scheduled-users profiles (preferences, habits, holidays, etc.), tasks, available time and resources. Every agent generates personalized plans depending on the scheduled-user profile. There is one ScheduleUser Agents for each nurse connected to the system.

Admin: plays two roles: the security role that monitors the users' location and physical building status (temperature, lights, alarms, etc.) through continuous communication with the Devices Agent; and the manager role that handles the databases and the task assignment. It must provide security for the users and ensure the efficiency of the tasks assignments. There is just one Admin Agent running in the system.

Device Manager: makes it possible for the interactive elements within the environment to interact. It deals with devices that use technologies such as RFID (Radio Frequency Identification), etc. It monitors the users' location (continuously obtaining/updating data from sensors), interacts with sensors and actuators to receive information and control physical services (temperature, lights, door locks, alarms, etc.), and also checks the status of the wireless devices connected to the system (e.g. PDA or Laptops). The information obtained is sent to the Admin Agent for processing. This agent runs on a Workstation. There is just one Devices Agent running in the system

Incident Manager: manages and resolves incidents, offers a user location service, and manages an alarm system.

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 [2]. Our case study is modeled as a conglomerate (*OVACareUnit*) made up of four units, each one dedicated to one type of functionality within the setting. The five units are: (i) *PlanningUnit*, contains the roles dealing with users' daily activities and dynamic plans: *Scheduler*; (ii) *UserUnit*, contains the roles associated with the user: *Communicator*, *User*, and *SuperUser*; (iii) *AdminUnit*, contains the roles assigned with global management tasks for the health-care system: *Admin*, *Incident Manager*, and *Analyst*;(iv) *DeviceUnit*, which contains the roles associated with the management of devices: *Device Manager*. This role inherits two specialized subroles: *CommApp*, which is responsible for all communications between applications and the platform, and *CommServ*, which is responsible for all communications between services and the platform.

The diagram in Figure 1a 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. In addition, *Provider* and *Client* roles are refined into these new units to specialize in functionality or in the use of specific services associated. 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 part of the internal model of the *UserUnit*. The internal structure for *OVACareUnit* and the remaining units was modeled in the same way.

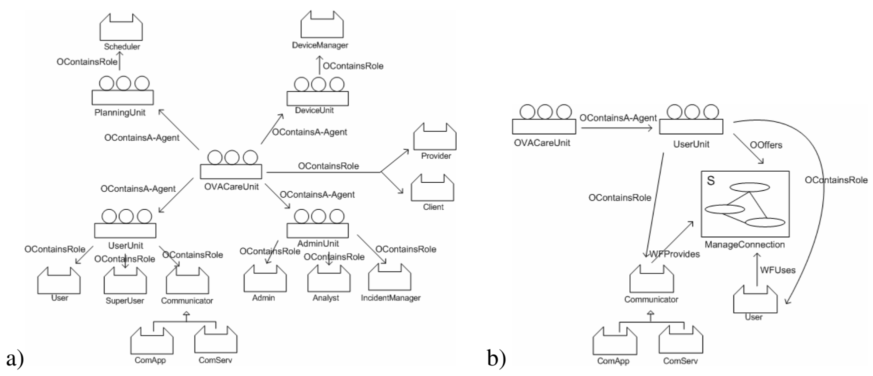

Fig. 1. a) Diagram of organization model: structural view b) Diagram of organization model: functional view of *UserUnit*

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 *UserUnit* 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 users to control their connection to the system.

Similarly, within the the services related to *AdminUnit* involve the overall management tasks within a health-care system (e.g., system incidents, data analysis, surveys, user management, notices, etc.). *PlanningUnit* is comprised of services that request guides based on user preferences and certain restrictions (time, specialization of the nurses, etc.). It also includes planning and replanning the route that the nurses will follow based on the suggested plans, 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 [11]. 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 user 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 *UserUnit* 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 UserUnit))*

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

Table 1. *ManageConnection* service in *UserUnit*

Service Specification					
Name: ManageConnection **Supplied by:** SF		**Description:** Manages the connection of an user **Required by:** **ClientRole:** User **ProviderRole:** Communicator			
Input Parameters					
Name	**Description**	**Mand.**	**Type**	**Value Range**	**Default**
requestTime	Time connection	Yes	date		
connectionData	Data connection	Yes	string		
operacion	Type of operation on the connection	Yes	string		
Output Parameters					
Name	**Description**	**Mand.**	**Type**	**Value Range**	**Default**
connection	Established connection between devices	Yes	connection		
Preconditions and Postcondition: ...					

3.1.1 Example of Service Planning with OVAMAH

The system considers the available time, time to initiate the task, task description, priority of the task, length of the task, and the patient associated with each task, and proposes the optimal route according to the nurse's profile. We will see the series of steps that are taken within the system when a planning route is requested, and how OVAMAH 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 will be used, or the API will directly execute the same functionality. This way, a community type *OVACareUnit* will be created, representing the organization, whose purpose is to control the helathcare system. It has four internal unit planes: *UserUnit*, *PlanningUnit*, *AdminUnit* 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 users (nurses, doctors) 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 system. In order to carry out this function, we have, for example *Co1*, *De1* and *Sc1* acting as agents that will carry out the roles of *Communicator*, *DeviceManager* and *Scheduler* respectively. Agents *U1* and *U2* represent the users that would like to receive a planning route.

Table 2. SF: Basic services

Service Facilitator					
Entity	**Action**	**Service**	**ClientRole**	**ProvRole**	**Profile**
UserUnit	Requires	manageConnection	User	Communicator	UserSP
DeviceUnit	Requires	Locate	Communicator/IncidentManager	DeviceManager	DeviceSP
…	…	…	…	…	…

Initially, all the agents head towards the OVAMAH 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: `UserUnit Requires manageConnection ClientRole=User;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 *UserUnit*: `AcquireRole(UserUnit, Communicator)`
If all goes well, the OMS will register *Co1* in the role of *Communicator* in *UserUnit* within the *Entity Play List*. This list shows the roles that the different agents assume within OVAMAH.

The *Co1* agent has carried out all of the regular steps for acquiring a role within OVAMAH. This process is illustrated in Figure 2 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 *CommunicatorInformationServiceProfile* corresponds to the profile of the *manageConnection* 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 *UserUnit*. 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 *UserUnit* 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. *AdminUnit*, *PlanningUnit* and *DeviceUnit* will return the services that are necessary for planning. The SF will again return a list (similar to Table 2).

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 users devices and make them comprehensible to the system. It will allow the order requested by the user 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

Table 3. *Entity Play List*

Entity Play List		
Entity	**Unit**	**Role**
Co1	UserUnit	Communicator
Co1	DeviceUnit	DeviceManager
Sc1	PlanningUnit	Scheduler
U1	AdminUnit	User
U2	AdminUnit	User

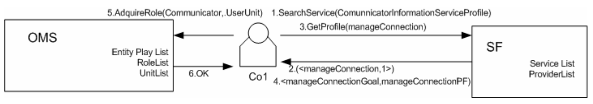
5.AdquireRole(Communicator,.UserUnit)
1.SearchService(ComunnicatorInformationServiceProfile)
3.GetProfile(manageConnection)
OMS
Entity Play List
RoleList
UnitList
6.OK
Co1
2.(<manageConnection,1>)
4.<manageConnectionGoal,manageConnectionPF)
SF
Service List
ProviderList

Fig. 2. Agent *Co1* registering

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 *De1* and *Sc1* agents will act in a similar fashion, registering at the end for the corresponding units *DeviceManager* and *Scheduler*. They too will be required to register the services as indicated by the defined norms. (*GenerateProfile*, *ConsultProfile*, *UpdateProfile*, *MSState, UpdateMSGState, Replan, ValidateRoute, ValueRoute, TaskListRecovery*) Each one is required for generating the optimal route for the user to follow. The *U1* and *U2* agents will request acquiring the *User* and *SuperUser* roles in order to access the basic services: *FindUserst, 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. *U1* can request existing services from the SF, and will receive a list with all the agents that offer their services. The *Entity Play List* would end up as shown in Table 3.

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 OVAMAH architecture for a dynamic MAS environment. This architecture has allowed us to directly model the organization of a health-care system 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.). OVAMAH provides us with the level of abstraction necessary for the development of our system, and the set of tools that facilitate its development. OVACARE makes use of OVAMAH distributing resources and enhancing its performance. It is demonstrated that a open approach is adequate to build distributed and highly dynamic Ambient Intelligence based multi-agent systems. In OVAMAH 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. OVACARE was employed as an illustration of not only the usage of OVAMAH components and services, but also of the dynamics of the applications to be developed with this architecture. In this way, examples of OVAMAH service calls have been shown through several scenarios, along with the evolution of different dynamic virtual organizations.

Acknowledgements. This research has been partially supported through projects JCyL BU006A08, MICINN TIN 2009-13839-C03-03 and MICINN CIT-020000-2008-2.

Referentes

1. Agüero, J., Rebollo, M., Carrascosa, C., Julián, V.: Agent design using model driven development. In: 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009), pp. 60–69 (2009)
2. Argente Villaplana, E.: Gormas: Guías para el desarrollo de sistemas multi-agente abiertos basados en organizaciones. Ph.D. thesis, Universidad Politécnica de Valencia (2008)
3. Bernon, C., Cossentino, M., Pavón, J.: Agent Oriented Software Engineering. The Knowledge Engineering Review 20(5), 99–116 (2005)
4. Boissier, O., Hübner, J.F., Sichman, J.S.: Organization Oriented Programming: From Closed to Open Organizations. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 86–105. Springer, Heidelberg (2007)
5. Carbó, J., Molina, J.M., Dávila, J.: Fuzzy Referral based Cooperation in Social Networks of Agents. AI Communications 8(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 (2009) (in press)
7. Cesta, A., Bahadori, S., Cortellesa, G., Grisetti, G., Giuliani, M., Locchi, L., Leone, G., Nardo, D., Oddi, A., Pecora, F., Rasconi, R., Saggese, A., Scopelliti, M.: The RoboCare Project, Cognitive Systems for the Care of the Elderly. In: Proceedings of International Conference on Aging, Disability and Independence (ICADI), Washington, D.C, USA (2003)
8. Corchado, J.M., Bajo, J., De Paz, Y., Tapia, D.I.: Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. In: Decision Support Systems. Eslevier, Amsterdam (2008b)
9. Decker, K., Li, J.: Coordinated hospital patient scheduling. In: Demazeau, Y. (ed.) Proceedings of ICMAS 1998, Paris, France, vol. 1134, pp. 104–111 (1998)
10. Glez-Bedia, M., Corchado, J.M.: A planning strategy based on variational calculus for deliberative agents. Computing and Information Systems Journal 10(1), 2–14 (2002)
11. GTI-IA. An Abstract Architecture for Virtual Organizations: The THOMAS project, `http://www.fipa.org/docs/THOMASarchitecture.pdf`
12. 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)
13. Lanzola, G., Gatti, L., Falasconi, S., Stefanelli, M.: A framework for building cooperative software agents in medical applications. Artificial Intelligence in Medicine 16(3), 223–249 (1999)
14. Nealon, J., Moreno, A.: Applications of Software Agent Technology in the Health Care domain. Whitestein series in Software Agent Technologies. Birkhauser, Basel (2003)