Enhancing Smart-Home Environments using Magentix2

S. Valero, E. del Val, J. Alemany, V. Botti[1]

Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València, 46020 Valencia, Spain
{svalero,edelval,jalemany,vbotti}@dsic.upv.es

Abstract. Multi-agent system paradigm has been envisioned as an appropriate solution for challenges in the area of smart-environments . Specifically, MAS add new capabilities such as adaption, reorganization, learning, coordination, etc. These features allow to deal with open issues in the context of smart-homes such as multi-occupancy, activity tracking or profiling activities and behaviors from multiple residents. In this paper, we present Magentix2 as a suitable MAS platform for the development of dynamic smart environments. Specifically, the use of Magentix2[1] facilitates the management of the multiple occupancy in smart living spaces. Normative virtual organizations provide the possibility of defining a set of norms and organizational roles that facilitate the regulation and control of the actions that can be carried out by internal and external agents depending on their profile. Moreover, Magentix2 provides a tracing service to keep track of activities carried out in the system. We illustrate the applicability and benefits of Magentix2 in a set of scenarios in the context of smart-homes.

1 Introduction

Currently, there is a number of social and economic drivers behind smart-homes that make the research in this area specially important [1]. In particular, there is an increasing number of elderly people in Europe that already live alone or prefer living independently, and they need assistance services. Thus, the value of building efficient and self-adaptive smart-home systems is of primordial relevancy. The multi-agent system (MAS) paradigm is envisioned as a strong candidate solution for challenges in the context of smart-homes [13, 15]. MAS are one of the most representative instances among artificial intelligent systems dealing with complexity and distribution. In the context of smart-homes, agents represent the entities in the environment and can be considered autonomous, adaptive, context-aware and capable of making decisions about actions (behaviors) based on their observations. They can be the software interface of the ubiquitous devices that offer their services or the software interface from the user-side that demands such services. In other terms, agents are called soft-sensors in smart environments and will be designed to learn from previous experiences and to reason about their local information in order to improve their decisions and achieve their goals. In this way, a smart-home (or environment) can be naturally viewed as a multi-agent system with distributed intelligence.

[1] http://gti-ia.upv.es/sma/tools/magentix2/index.php

In this paper, we present the application of the MAS platform Magentix2 and its new functionality to the context of smart-homes. Magentix2 integrates an implementation of the THOMAS framework that includes the management of organizations and norms that allows the control of actions of internal and external agents in open environments. Besides these features, Magentix2 includes a tracing service that reduces the amount of information traffic. The inclusion of these features in Magentix2 facilitates the development of a smart-home environment as a MAS with virtual organizations, normative contexts, and tracing service. Magentix2 allows to enhance smart-environments in the following ways: (i) facilitates the definition of different user profiles for the inhabitants through organizational roles and living spaces using the concept of organization units. This functionality improves the dynamic adaptation of the functionality offered by each space of the smart environment to different profiles of inhabitants; (ii) provides control over the access of external entities to the system. It also facilitates an adaptive and flexible control over the access to services offered by agents that are part of a smart environment considering inhabitant profiles. This feature is performed in Magentix2 through the use of norms in a normative environment; and, (iii) traces the behavior of the inhabitants. The tracing service reduces the amount of unnecessary information that inhabitants receive by filtering, according to certain criteria, all the data generated in the system. Agents only receive relevant information that helps them to fulfil their goals efficiently.

The remainder of this article is organized as follows. Section 2 shows the most relevant works in the literature with regard to MAS approaches applied to smart-environments. Section 3 presents Magentix2 platform components in three levels (i.e., organizational, agent, and interaction level) and an in-depth explanation of the normative context of the organization, which is at the core of the organizational level, and the tracing functionality. Section 4 analyzes the use of Magentix2 platform and the effects of its new functionality in a set of scenarios in the context of a smart-home. Finally, Section 5 presents some concluding remarks and future work.

2 Related Work

Intelligent control systems can potentially allow us to achieve a variety of benefits for humans daily living experience in the environments where they live and work [2]. Traditionally, smart-home environments have been seen as centralized systems oriented to a single-user where home appliances are connected to the home network. Advances in ambient intelligence, ubiquitous computing, and autonomic computing and their application into smart-homes have added intelligence to our housing and have facilitated the decentralization of the system information and functionality. These features provide flexibility and adaptability to the smart-home system [11]. Through the incorporation of intelligent remote control systems that supervise the home appliances and devices it has been possible to improve the quality of living in several aspects: comfort, healthcare, safety, security, and energy efficiency. The research in the area of smart-homes has tackled different technical issues such as heterogeneity in devices and technologies, context awareness, and security in order to facilitate the implementation of intelligent environments.

Although several technical challenges have been achieved, there are still open issues. On one hand, the major part of research on smart-homes until now has focused on technical challenges required to provide ubiquitous and context-aware services considering that there is a single user in the living space [6, 21]. However, usually more than one resident occupy living spaces. The main challenges related to the multi-occupancy can be classified in three groups: activity tracking [22], profiling activities and behaviors [8], and conflicts in input preferences from multiple residents [7, 9].

There are previous works in the area of MAS that deal with some multiple occupancy problems. Sanchez-Anguix et. al [18] introduce negotiation models with limited devices to solve conflicts among agents. Rodriguez et. al [17] propose a MAS architecture based on virtual organizations that combines data obtained by multiple sensors in order to identify and localize the position of the inhabitants in a residential home for the elderly. Sun et al. [20] propose a multi-agent design framework for a smart-home and home automation applications. They propose a BDI model for agent individual behavior design and a regulation policy-based method for multi-agent group behavior design. Also, a Petri-net based method is developed for system evaluation and analysis. Loseto et. al [14] propose a flexible multi-agent approach for smart-environments based on semantic resource discovery and orchestration. They also include negotiation techniques between user agents and the agents that represent the devices. However, as in the majority of approaches, the user agents are assumed that all of them play the same role. Bajo et. al [3, 16] present a previous version of the THOMAS architecture applied to a similar scenario used in this paper.

Among existing implementation platforms of multi-agent systems, JADE is the most popular. There are several works [19, 4] that present a multi-agent system based on JADE platform to deal with health-care in smart environments and smart-homes due to the use of the Java programming language, its compliance with FIPA standards, and its mobile versions (JADE Leap, SubSense).

In comparison with the above approaches, our proposal presents three different contributions that go a step further into the state of the art. Our proposal presents a set of THOMAS [10] extensions that give support to the dynamic management of normative virtual organizations. These extensions are implemented and provided by Magentix2 and facilitate the consideration of multiple occupancy. Normative virtual organizations provide (i) the possibility of defining living spaces as virtual organizations; and, (ii) a set of norms associated to a virtual organization that facilitates the regulation and control of the actions (services) that can be carried out depending on the profile of an inhabitant. These two features facilitates the adaptation and personalization of the functionality offered by the smart-home to the profile of the inhabitants in a dynamic way. Finally, Magentix2 also includes a trace functionality in order to keep track of the activities inside organizations. This new functionality facilitates the supervision of the services demanded and the selection of the data generated in the system based on the interests of agents reducing the amount of unnecessary information.

3 MAS Platform: Magentix2

Magentix2 [2] is a platform for open MAS. Its main objective is to bring agent technology to real domains such as business, industry, logistics, e-commerce, health-care, or smart-environments among others. In this section, we provide an overview of the platform components and we focus on two functionalities offered by the platform (i.e., *norms* at organizational level and *trace model*) that are relevant for smart-home scenarios.

3.1 Magentix2 overview

Nowadays, Magentix2 provides support at three levels: organizational, interaction, and agent level (see Figure 1):

Organizational level. This level consists on technologies and techniques related to agent societies. This level integrates the THOMAS framework [10] to provide a complete support for virtual organizations and SOA-like services.THOMAS offers a set of modular services provided by two main components, the Service Facilitator (SF) and Organization Manager Service (OMS). The SF offers a yellow/green page service. The OMS is mainly responsible for the management of the organizations. Different types of virtual organizations are supported. Each organization can contain other organizations. Furthermore, it is possible to define roles in each organization. These roles are characterized by attributes (i.e., position, accessibility and visibility), that can restrict the access to the available services. Also, it is possible to define a normative context to further restrict (or permit) the access to THOMAS's services. The normative context can be modified in any moment if necessary, in order to adapt it to changes in the environment.

Interaction level. The platform supports flexible interaction protocols and conversations, indirect communication by means of a tracing service and interactions among agent organizations. Magentix2 uses the Advanced Message Queuing Protocol (A-MQP)[3] as a foundation for agent communication. Moreover, Magentix2 allows heterogeneous agents, even from other type of agent platforms, to interact with each other via FIPA-ACL messages.

Additionally, Magentix2 offers a communication mechanism that allows mass communication among agents of an organization, taking into account the type of roles that agents play or the type of the organization.

Furthermore, the Tracing Service Support of Magentix2 [5] allows agents to share information in an indirect way by means of tracing events. In order to facilitate this labour to platform agents, Magentix2 incorporates a Trace Manager (TM).

Agent level. Developers have available different classes of agents. For example, in the smart-home context, it might be suitable to use two of them: CAgent (i.e., an agent that allows the automatic creation of simultaneous conversations) and JasonAgent (i.e., BDI agents that can participate in simultaneous conversations).

[2] http://www.gti-ia.upv.es/sma/tools/magentix2/index.php

[3] http://www.amqp.org/

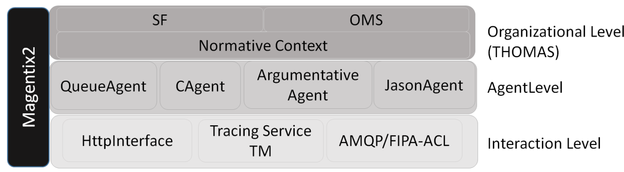

Fig. 1: Magentix2 platform: levels, components, and services.

To summarize, the Magentix2 platform supports flexible interaction protocols and conversations, indirect communication and interactions among agent organizations. Also, the platform permits the definition of a normative context, which can be easily adapted to the changes in the system requirements. In this paper, we focus on the organizational level and its *normative context*, and the *trace model*. In the following sections, we describe in more detail each of these features of Magentix2.

3.2 Normative Context at Organizational Level

THOMAS framework, that is integrated in Magentix2 platform, allows defining norms that prescribe agent rights and duties in terms of who can provide/ask for a service, when and under which circumstances. In addition, norms can be seen as a coordination skill for organizing MAS, since they specify the desired behavior of the society members.

From the functional point of view, agents must play a certain type of role inside the organization to access some OMS services (predefined norms). Table 1 shows these requirements for some of the OMS services (also there are other services to deregister or obtain information about all the THOMAS components). However, agents can add PERMITTED or FORBIDDEN norms to relax or restrict the default access control to the OMS services in an organization by means of the *registerNorm* service of the OMS. The OMS agent is responsible for verifying if a norm applies before providing a service. In particular, the order in which norms are checked before providing a service of the OMS is as follows: (i) PERMITTED norms, if one is fulfilled, the service is provided with no restrictions; (ii) FORBIDDEN norms, if one is satisfied, the service is not provided; and finally (iii) STRUCTURAL NORMS, that is, all predefined norms of the service are checked before providing the service. If no one is satisfied, the service is provided as usual.

Norms are registered into an organizational unit with a unique name inside that organization. They should be written following the syntax of the THOMAS normative language (see [12] for a detailed explanation), which is based on AgentSpeak language. Specifically, a norm is defined as follows:

$$@normName[Deontic, Target, Action, Activation, Expiration] \quad (1)$$

where,

OMS Services	Position			
	Creator	Member	Supervisor	Subordinate
RegisterUnit	✓	-	-	-
RegisterRole	✓	✓	✓	-
RegisterNorm	✓	✓	✓	-
AcquireRole	✓	✓	✓	✓ (1)
AllocateRole	✓	✓	✓	-
InformAgentRole	✓	✓	✓	✓
(1) The agent could acquire another role with position=*creator* or position=*supervisor* in its organization				

Table 1: Example of some structural norms for accessing to some OMS services. They take into account the position played by the requesting agent and the type of organization affected by the service.

- $Deontic \in \{f, p\}$, where f is used for forbidden norms, which restricts the access to services; and p is used for permitted norms, which relaxes the access to services.
- $Target =< type, id >$ where $type \in \{agentName, roleName, positionName\}$, whereas id is its associated value (for example, the id associated to positionName could be *creator*) or the anonymous variable "_", when any value is accepted. The field *Target* allows users to determine which agents will be affected by a norm. For example, taking into account an scenario in which a norm was registered in an organizational unit with the target $< agentName, consumer >$, this norm only affects to agents whose name is *consumer*.
- $Action$ is the name of the service. THOMAS only manages norms that correspond to an OMS service.
- $Activation$ is a well-formed formula expressed by means of first-order predicates that indicates the conditions to fulfil a norm. Users should add predicates related to data known in the THOMAS world (role details, played roles, organization structure, agent names, etc). The *Activation* could be empty ("_"), in that case the norm is always fulfilled.
- $Expiration$ is a well-formed formula expressed by means of first-order predicates that indicates when a norm expires. If the expiration of a norm is satisfied, the norm is not applied although the activation is fulfilled. Users should add predicates related to known data in the THOMAS world. The *Expiration* could be empty ("_"), in that case the norm never expires.

3.3 Trace Model

This section describes the Tracing Service Support available in Magentix2. This Tracing Service Support allows agents in a Multiagent System (MAS) share information in an indirect way by means of trace events. In Magentix2, a trace event is a piece of data representing an action which has taken place during the execution of an agent or any other component of the multiagent system. Trace events are generated each time the execution flow of an application reaches certain instructions (tracing points) in its source code. In fact, agents which are interested in sharing their trace events, offer them as *tracing services*. Agents publish their available tracing services and other agents can

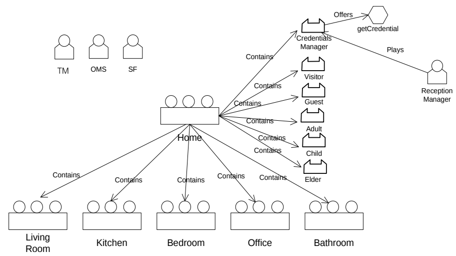

Fig. 2: Organizational level of a smart-home.

request those tracing services in which they are interested in. In this way, trace event traffic is reduced and agents do not have to process other trace events which they are not interested in. Magentix2 provides this service through the *Trace Manager* agent (TM), which is in charge of managing the publications, subscriptions and listings of tracing services. Among other, the most used actions associated to tracing events are:

$$\begin{array}{l} publishTracingService(applicantAgent, serviceName, description) \\ unpublishTracingService(applicantAgent, serviceName) \\ requestTracingService(requesterAgent, serviceName, originEntity) \\ cancelTracingServiceSubscription(requesterAgent, serviceName, originEntity) \\ listTracingEntities(requesterAgent) \end{array} \tag{2}$$

where, *applicantAgent* represents the agent that publishes its service in the tracing service; *originEntity* is agent which offers the tracing service; *serviceName* identifies the tracing service; *requesterAgent* is the agent which is requesting the subscribing/unsubscribing a tracing service.

4 Smart-Home based on Magentix2

In order to illustrate the use of Magentix2 in a smart environment, we have considered a smart-home an scenario where the smart-home is modeled as a service-oriented MAS based on virtual organizations. In the following sections, we describe the organizational level, the normative context, and the system dynamics of our proposal.

4.1 Organizational level

The organizational view describes the components of the system and their relationships. In this view, we define the agents, the organizational units, and the roles defined inside these organizational units. The organizational level of the MAS for smart-homes consists of the following entities (see Figure 2):

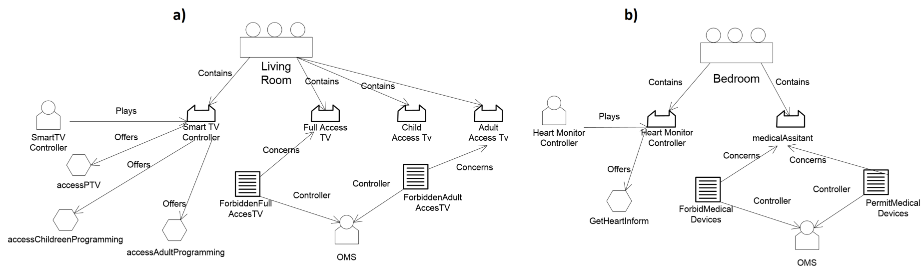

Fig. 3: Organizational view of the living room and the bedroom.

- Agents: there is an agent that plays the organizational manager role *OMS* and an agent that plays the service facilitator role *SF*. There is also an agent responsible for the tracing service that plays the *TM* role.
- Organizational Units (OU): we consider an organization unit called *Home OU*. Inside *Home OU*, we define a organizational unit for each of the physical rooms of the house.
- Roles: at the *Home OU* there is defined a set of organizational roles. These roles define the profiles that can be played by the agents inside the *Home OU*. We have considered a role for each type of inhabitant of the living space. The roles *Adult*, *Child*, and *Elder* represent the usual inhabitants of our smart-home scenario. *Visitor* and *Guest* represent temporary inhabitants. Finally, the role *Credentials manager* represent the profile of an agent that manages who enters in the *Home OU* and which role an external agent must acquire to be member of the organization unit.

As an example, Figure 3.a describes with detail the *LivingRoom OU*. The *LivingRoom OU* is an organization unit inside *Home OU* (see Figure 2). Inside the *LivingRoom OU* there is a set of roles defined. The *smartTVcontroller* role offers a set of services related to available TV programs in the TV. The roles *FullAccessTV* and *ChildAccessTv* are defined in order to control the access to the TV programs. In a similar way, *Bedroom OU* is described in Figure 3.b. In this case, *HeartMonitorController* role offers a service to provide information about which is monitoring a heart monitor device. Also, the *medicalAssistant* role controls the access to the services provided by the medical devices of the room.

4.2 Normative Context at Organizational Level

Once the organizational units *OU* are defined in the smart-home, we can define a set of norms that controls the acquirement of the required roles to access to the services offered by the *OU* of the system. As an example, Figure 4 shows the norm called *forbiddenAcquireRole*. This norm is registered in the *Home OU* and prevents agents from acquiring roles directly. Thus, they only can acquire roles (credentials) through the *credendialsManager* agent. This norm is described using the Magentix2 normative language described in definition (1).

```
@forbiddenAquireRole[f, <agentName:_>, acquireRole(_,home,_),_,_]
```

Fig. 4: Forbidden norm registered in the *Home OU* that prevents agents from acquiring roles directly.

In a similar way, Figure 3 shows, in the organizational level, a set of norms that controls which agents can play the roles defined in the *LivingRoom OU*. Specifically, the norm *forbiddenFullAccessTV* is shown in Figure 5. This norm avoids that agents which play *child* or *visitor* roles in *Home OU* buy films, because they cannot obtain the role required to request the service.

```
 @forbiddenFullAccessTV[f,<agentName:Agent>,acquireRole(fullAccessTV,
livingRoom,Agent),isAgent(Agent)& not playsRole(adult,home,Agent),_]
```

Fig. 5: Forbidden norm registered in the *LivingRoom OU* that prevents agents with roles that differ from *adult* acquiring the required role to have full access to the TV.

In the *Bedroom OU*, there is a set of norms (Figure 6) which controls the access to information services provided by medical devices, such as a heart monitoring, a sphygmomanometer, etc. These information services require than requester agents play the role *medicalAssistant*. These norms indicate that, in the *Bedroom OU*, only agents who play the role adult in the *Home OU* can access to the required role *medicalAssistant*. Thus, in order to allow other agents access to that information services (i.e., medical staff), the inhabitants of the house should dynamically add norms that explicitly permit acquire that role.

```
@permitMedicalDevices[p, <agentName:Agent>,  acquireRole(medicaAssistant,bedroom,Agent),
 isAgent(Agent)& playsRole(adult,home,Agent),_]
@forbidMedicalDevices[f, <agentName:_>,  acquireRole(medicaAssistant,bedroom,_),_,_]
```

Fig. 6: Norms registered in the *Bedroom OU* which control the access to the medical devices of the bedroom.

4.3 System dynamics

In the above sections, we have defined the organizational units and the required norms to control the access to the services defined inside of them. In this section, we focus on the system dynamics (i.e., how agents interact with the system) considering the previous *OU* and *norms* defined. In order to illustrate how agents interact in the smart-home system, we present a set of scenarios.

a) Access to the Smart-home. An external agent a_i wants to get into the smart-home system. The entry point is the *Home OU*. To be a member of the organization, the agent a_i must play at least one of the roles defined inside of *Home OU*. In order to acquire one of these roles, the external agent a_i contacts with a *ReceptionManager* agent that is inside the organization and plays the role *CredentialsManager*. The *ReceptionManager* agent offers a service (i.e., *getCredentials*) that allocates roles to newcomers based on the information contained in their profiles. The *ReceptionManager* interacts with the OMS agent to register members in the organization. After that, the newcomer agent a_i is part of the *Home OU*. An example of the sequence of steps to acquire a role in order to access to *Home OU* is shown in Figure 7.

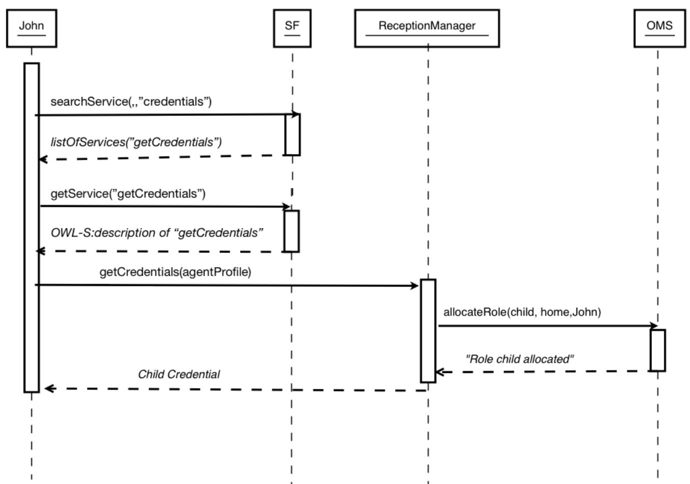

Fig. 7: Access to the smart-home. A child named John gets his corresponding credential.

b) Asking for a service that an agent is not allowed to ask. An agent a_i that plays the role *Child* in *Home OU* enters in the organization *LivingRoom OU*. The agent a_i wants to request a service in order to buy a film using the smart TV device. The agent a_i asks the SF agent for a service of films and gets information about services that offer pay-per-view movies and which are their provider agents. The agent a_i selects the smart TV provider agent and asks this provider for the service. In order to provide the service to the agent a_i, the smart TV agent checks if a_i plays the required role. This verification is done by asking the OMS about the role that plays the requester agent a_i. Based on this information, the smart TV agent decides whether offers or not the service to a_i. Figure 8 shows an example of the sequence of steps that agent John follows to try to get access to the service *buyFilm*.

c) Asking for a service that an agent is allowed to ask. An agent a_i that plays the role *Adult* in *Home OU* enters in the *LivingRoom OU*. The agent a_i wants to request

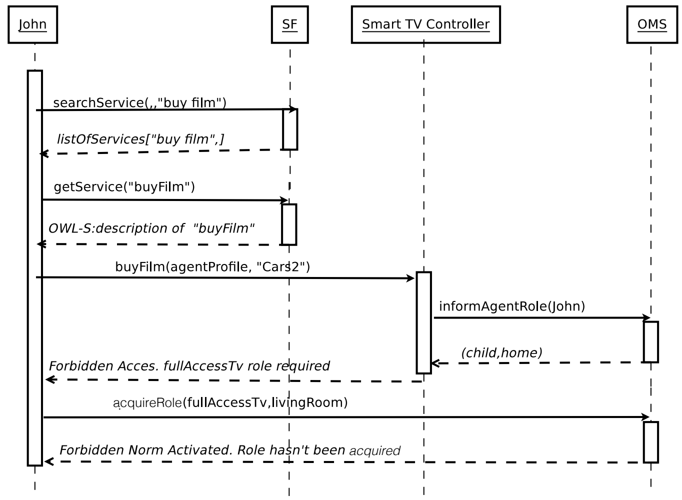

Fig. 8: Asking for a service that an agent is not allowed to ask. A child named John tries to access to the *buyFilm* service.

a service in order to buy a film. Therefore, it asks the SF agent for a service of films and then it gets the information about the available services and the agents that provide them. The agent a_i analyzes the information about the service profiles of the provider agents. Agent a_i notices that to ask for the service, it is necessary to play the role *FullAccessTV*. Therefore, agent a_i interacts with the OMS agent in order to acquire this role. Once agent a_i plays this role (i.e., *FullAccessTV*), it asks the smart TV agent for the service. Figure 9 illustrates the protocol that an agent with the required roles follows to ask for a service.

d) Access of a visitant (medical assistant) and trace events. An internal agent a_j that plays the role *adult* decides to allow an external agent to access into the organization *Home OU* in order to, periodically, take care of an elderly inhabitant. For this reason, agent a_j registers a new norm that permits this agent to access to the information services provided by the medical devices (see Figure 11).

Then, an external agent a_i, that represents a medical assistant, wants to get into the smart-home system in order to control the health of one of the inhabitants. The entry point is the *Home OU*, as explained before (in case a). The *CredentialsManager* allocates newcomer agent with the role *Visitant*, taking into account its profile. After that, the newcomer agent a_i is part of the *Home OU*. Then, agent a_i only needs to acquire the role *medicalAssistant* to access to the information services provided by the agents which control each of the medical devices. With the results of these services, it will create an inform about the current situation of the elderly inhabitant.

Moreover, agent a_j would be interested in tracing some activities performed by other inhabitants. For instance, agent a_j that plays the role *Adult* can ask for a trac-

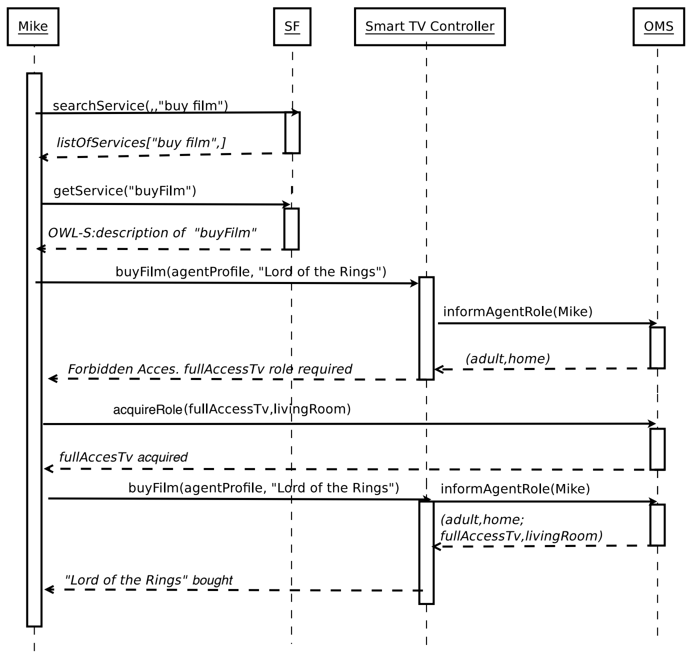

Fig. 9: Asking for a service that an agent is allowed to ask. An adult named Mike asks for the *buyFilm* service.

ing service that informs him when other agents access some services provided by the medical devices of the *Bedroom OU*. With the trace event obtained, agent a_j controls if the medical assistant arrived at home and the results obtained by the medical control services. The trace can be also useful for controlling the activity of other inhabitants. For instance, if an agent that plays the role *Child* has tried to ask for a service that it is not allowed for agents that play the *Child* role.

5 Conclusions

In this paper, we have presented Magentix2 as a suitable platform for the development of smart environments. Specifically, it provides capabilities such as adaptation, reorganization and coordination that enhance the current development of smart-homes. In this paper, we focus on the following features. First, the inclusion of norms and virtual organizations that allows to define different user profiles for the inhabitants and living spaces using the concept of organization units. These features improve the adaptation of the functionality offered by each space of the smart environment to different profiles of inhabitants dynamically. Second, the control over the access of external entities to the system. Third, an adaptive and flexible control over the access to services offered by agents that are part of a smart environment considering inhabitant profiles. This fea-

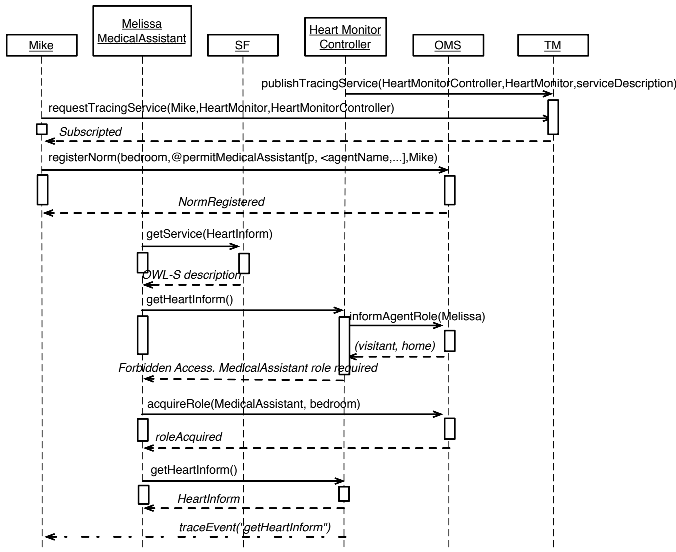

Fig. 10: Access of a medical assistant: shows the actions performed to obtain some informs from medical devices and how the activity of the medical assistant can by traced

```
@permitMedicalDevicesStaff[p, <agentName:melissa>, acquireRole(medicaAssistant,
bedroom,Agent),isAgent(melissa)& playsRole(visitor,home,Agent),_]
```

Fig. 11: New norm registered in the *monitoredBedroom OU* to permit medical staff the access to the medical devices.

ture is performed in Magentix2 through the use of norms in a normative environment. Fourth, Magentix2 facilitates the monitoring of the activities of the inhabitants. Monitoring is carried out by the tracing service. This service allows agents to only receive relevant information that help them to fulfil their goals efficiently.

A set of scenarios has been analyzed in detail to show in which situations Magentix2 facilitates the management of multiple inhabitants in a smart-home and the control of the access to services offered by the system, taking into account living stays (organizations) and inhabitants profiles (roles). Among the scenarios, we considered the access of external entities to the system, the request of services of an organization when the agent has a certain role, the access of external entities to services of an organization, and the tracking of the activity inside the system in order to be aware of what is happening in the smart-home environment.

Acknowledgements.

This work is supported by the Spanish government grants PROMETEOII/2013/019, TIN2014-55206-R, TIN2015-65515-C4-1-R, H2020-ICT-2015-688095.

References

1. M. R. Alam, M. B. I. Reaz, and M. A. M. Ali. A review of smart homespast, present, and future. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):1190–1203, 2012.
2. J. C. Augusto, V. Callaghan, D. Cook, A. Kameas, and I. Satoh. Intelligent environments: a manifesto. *Human-Centric Computing and Information Sciences*, 3(1):1–18, 2013.
3. J. Bajo, J. A. Fraile, B. Pérez-Lancho, and J. M. Corchado. The thomas architecture in home care scenarios: A case study. *Expert Systems with Applications*, 37(5):3986–3999, 2010.
4. K.-I. Benta, A. Hoszu, L. Văcariu, and O. Creţ. Agent based smart house platform with affective control. In *Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship*, page 18. ACM, 2009.
5. L. Búrdalo, A. Garca-Fornes, V. Julian, and A. Terrasa. TRAMMAS: A tracing model for multiagent systems. *Engineering Applications of Artificial Intelligence*, 24(7):1110–1119, 2011.
6. C. Cetina, P. Giner, J. Fons, and V. Pelechano. Autonomic computing through reuse of variability models at runtime: The case of smart homes. *Computer*, 42(10):37–43, 2009.
7. D. J. Cook. Multi-agent smart environments. *Journal of Ambient Intelligence and Smart Environments*, 1(1):51–55, 2009.
8. A. S. Crandall and D. J. Cook. Using a hidden markov model for resident identification. In *6th International Conference on Intelligent Environments*, pages 74–79. IEEE, 2010.
9. S. Davidoff, M. K. Lee, J. Zimmerman, and A. Dey. Socially-aware requirements for a smart home. In *in Proc. of the International Symposium on Intelligent Environments*, pages 41–44, 2006.
10. E. del Val, N. Criado, M. Rebollo, E. Argente, and V. Julian. Service-Oriented Framework for Virtual Organizations. In *International Conference on Artificial Intelligence (ICAI)*, volume 1, pages 108–114. CSREA Press, 2009.
11. E. del Val, M. Rebollo, and V. Botti. Combination of self-organization mechanisms to enhance service discovery in open systems. *Information Sciences*, 279:138–162, 2014.
12. Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA), http://www.gti-ia.upv.es/sma/tools/magentix2/archivos/Magentix2UserManualv2.1.0.pdf. *Magentix2 User's Manual v2.0*, January 2015.
13. L. Klein, J.-y. Kwak, G. Kavulya, F. Jazizadeh, B. Becerik-Gerber, P. Varakantham, and M. Tambe. Coordinating occupant behavior for building energy and comfort management using multi-agent systems. *Automation in Construction*, 22:525–536, 2012.
14. G. Loseto, F. Scioscia, M. Ruta, and E. Di Sciascio. Semantic-based smart homes: a multi-agent approach. In *13th Workshop on Objects and Agents (WOA 2012)*, volume 892, pages 49–55, 2012.
15. I. Mocanu, L. Negreanu, and A. M. Florea. A multi-agent system for service acquiring in smart environments. In *Intelligent Distributed Computing VI*, pages 297–306. Springer, 2013.
16. S. Rodriguez, V. Julián, J. Bajo, C. Carrascosa, V. Botti, and J. M. Corchado. Agent-based virtual organization architecture. *Engineering Applications of Artificial Intelligence*, 24(5):895–910, 2011.

17. S. Rodríguez, J. F. D. Paz, G. Villarrubia, C. Zato, J. Bajo, and J. M. Corchado. Multi-agent information fusion system to manage data from a {WSN} in a residential home. *Information Fusion*, 23(0):43 – 57, 2015.
18. V. SáNchez-Anguix, S. Valero, V. Julián, V. Botti, and A. GarcíA-Fornes. Evolutionary-aided negotiation model for bilateral bargaining in ambient intelligence domains with complex utility functions. *Information Sciences*, 222:25–46, 2013.
19. C.-J. Su and C.-Y. Wu. Jade implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring. *Applied Soft Computing*, 11(1):315–325, 2011.
20. Q. Sun, W. Yu, N. Kochurov, Q. Hao, and F. Hu. A multi-agent-based intelligent sensor and actuator network design for smart house and home automation. *Journal of Sensor and Actuator Networks*, 2(3):557–588, 2013.
21. C.-L. Wu, C.-F. Liao, and L.-C. Fu. Service-oriented smart-home architecture based on osgi and mobile-agent technology. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(2):193–205, 2007.
22. J. Yin, Q. Yang, D. Shen, and Z.-N. Li. Activity recognition via user-trace segmentation. *ACM Transactions on Sensor Networks (TOSN)*, 4(4):19, 2008.