Expert Systems with Applications 39 (2012) 10656–10673

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Methodological design and comparative evaluation of a MAS providing AmI

Verónica Venturini, Javier Carbo *, José Manuel Molina

Carlos III University of Madrid, Av. De la Universidad Carlos III, 28270 Colmenarejo, Madrid, Spain

ARTICLE INFO

Keywords:
Multi-agent
Context Aware
Ubiquitous Computing
Ambient intelligence
Location

ABSTRACT

Researches on Ambient Intelligent and Ubiquitous Computing using wireless technologies have increased in the last years. In this work, we review several scenarios to define a multi-agent architecture that support the information needs of these new technologies, for heterogeneous domain. Our contribution consists of designing in a methodological way a Context Aware System (involving location services) using agents that can be used in very different domains. We describe all the steps followed in the design of the agent system. We apply a hybridizing methodology between GAIA and AUML. Additionally we propose a way to compare different agent architectures for Context Aware System using agent interactions. So, in this paper, we describe the assignment of weight values to agents interaction in two different MAS architectures for Context Aware problems solving different scenarios inspired in FIPA standard negotiation protocols.

1. Introduction

Ambient Intelligent (AmI) refers to a world where the intelligence is into any object around us (Wright, Gutwirth, Friedewald, Vildjiounaite, & Punie, 2008). Effectively, applications of AmI have arisen in different domains such as social environments (Kjeldskov & Paay, 2005), education (Gu, Shi, Xu, & Chen, 2005), cultural (Bombara, Cali, & Santoro, 2003), tourist (Paganelli, Bianchi, & Giuli, 2006), and health care (Corchado, Bajo, de Paz, & Tapia, 2008). In them, computers monitor our movements, activities, routines and behaviors to predict what want to do in a particular time. This information that characterizes a situation involving objects, persons and places is often called "context" (Dey, Abowd, & Salber, 2001).

AmI is supposed to be achieved through the use of systems that use context to provide services adapted to the situation defined by the context. In Context Aware Systems, user current situation involves user position. For its acquisition, location services play a relevant role in these systems.

Different platforms of context services provisions were proposed in the last years. Context Toolkit (Dey et al., 2001) is a simple widgets-based application, which works using attribute-value pairs and has a context aggregator that allow add context information as location or sound level. Other context platform is SoCAM (Gu, Pung, & Zhang, 2004), that is a distribute middleware that uses ontologies to model the context, and its architecture is composed by context providers, context interpreters, context database, service-locating services and other more services; this platform works with forward and backward chaining rules engine. Other relevant context platforms are Hydrogen (Hofer, Schwinger, Pichler, Leohartsberger, & Altmann, 2003), Cortex (Sorensen et al., 2004) and Citron (Yamabe, Takagi, & Nakajima, 2005).

An interesting approach to implement Context Aware Systems is the use of agent technology, since AmI is an ideal workspace for agents because autonomous distributed and proactive agent nature (Wooldridge & Jennings, 1995).

Several Context Aware Systems adopted this agent-based approach to AmI. For instance CoBRa (Chen, Finin, & Joshi, 2003) is an agent-based architecture that works with a context broker agent, who maintains a context model for the community. Despite the use of agents, its centralized view does not provide an appropriate solution for AmI applications. Other previous uses of agents in AmI are domain-specific: ALZ-MAS for health care (Glaser, 1997; Moreno, Valls, & Viejo, 2003) for searching a taxi in the city; ASK-IT by Spanoudakis and Moraitis (2006) for assisting users in tourism activities. In other hand, there is some generic multi-agent architecture such as Open Agent Architecture (OAA), which was one of the first generic architecture proposed in Martin, Cheyer, and Moran (1995). It works with a set of agents as facilitator, application, meta-agent and user interface agent. Client agents, surprisingly, are considered external agents that operate through a facilitator agent. The main drawback of OAA is then the lack of interest over client preferences. Finally, CONSORTS agent architecture (Nakashima, 2007) was built taking into account the spatial–temporal reasoning agent, services agent and personal agent. But, in our point of view, it lacks of a specific agent to analyze user profile in relation with the services that providers offer.

* Corresponding author. Tel.: +34 916241314.
E-mail addresses: veronica.venturini@uc3m.es (V. Venturini), javier.carbo@uc3m.es (J. Carbo), josemanuel.molina@uc3m.es (J.M. Molina).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.02.170

We proposed a similar architecture to those previous works in Fuentes, Sánchez, Carbó, and Molina (2007) that involved three kinds of agents: Central, User and Provider. But it is still another centralized approach to the problem, in this paper we intend to overcome this limitation with a truly distributed processing and responsibilities. We also intend to include a specific agent for analyzing user profiles.

Our contribution consists of designing in a methodological way a Context Aware System (involving location services) using agents that can be used in very different domains. We describe all the steps followed in the design of the agent system, including an ontology reusing properties and concepts of similar projects such as Fuentes, Carbó, and Molina (2006).

In spite of the diversity of the already existing agent-inspired methodologies (SIGMA Tiba & Capretz, 2006, INGENIAS Pavón & Gómez-Sanz, 2003, CoMoMAS Glaser, 1997, GAIA Wooldridge, Jennings, & Kinny, 2000 and Agent UML (AUML) Huget et al., 2002), we concluded that the more specific to Agent Development is GAIA. But on the other hand it does not allow to go in the implementation way in a direct form, and also GAIA diagrams lack of relationships among them.

We apply an hybridizing methodology between GAIA and AUML. Specifically we follow Huhns (Huhns, 2004) view of AUML as a methodology instead of considering AUML just as a modeling language. Additionally, Agent UML (AUML) is the proposed language being accepted as part of FIPA-99 standard (Bauer, Muller, & Odell, 2001).

We combine both using AUML agent and interaction models, and we employ the role and services models that GAIA provides. AUML Agent model summarizes and relates all the models used in AUML methodology but without details, while AUML interaction model (with GAIA support) allows the system implementation easier for the developer.

Finally evaluating these complex systems is very problematic due to the distributed nature of agents and the diversity of scenarios that Context Aware Systems address. Additionally we propose a way to compare different agent architectures for Context Aware System. It considers agent interactions as the most important characteristic of agent systems (Wooldridge & Ciancarini, 2001). So, in this paper, we describe the assignment of weight values to agents interaction in two different MAS architectures for Context Aware problems solving different scenarios.

The structure of this paper is the following: Section 2 gives an explanation about problem definition; Section 3 includes the multi-agent system methodological design; Section 4 shows the agent architecture finally proposed; Section 5 shows the comparative evaluation of two agent approaches to Context Aware System and it concludes in Section 6.

2. Problem definition

We assume a generic context-based scenario that includes users requiring satisfaction of their needs, and providers solving these needs. Providers intend to offer services according user preferences. In this generic scenario location services are relevant since services are provided according user's position. For example, if we are in technology fairground domain, a provider could be the Nokia stand, and the provided services could be: give cell phone information, fix cell phone and sell accessories.

A common activity that we can observe in this scenario is that users are continuously in interaction with their pairs. In our model we want to take into account the other users' influence in user decisions, for instance, through recommendations. In fact, recommendations play a key role in order to obtain customized services according to user current situation. For example, users might want to obtain personal recommendations about the places they are currently walking (historic buildings, shopping, etc.).

This customization is implemented using acquired user profiles. In this sense, we intend to acquire in an unobstrusive way user profiles by observing externally user's actions in the system. This is what we call public user profile in opposition to the private user profile that is used internally by the user to make decisions.

Finally, we also intend to address the possibility of users forming coalitions when they have shared interests. Then they would run an activity together or share information inside the coalition.

Summarizing, the relevant concepts of the generic scenario we are intending to model are: a location service, user to user recommendations, unobtrusive acquisition of public user profiles and coalition formation. The idea is to design and build an architecture that includes all the features mentioned before using agent technology.

To illustrate the situation, let us suppose the case of the conference context (Bravo, Hervás, Nava, Chavira, & Sanz, 2005) shown in Fig. 1. A user called John arrives at the conference site and goes at parking. He receives in his GPS an alarm about empty place in the parking. After that, he goes directly to reception. But as he had registered few days ago, he may receive personalized services directly in his mobile device, after logging in. He then may look the list of the incoming sessions in his mobile device without needing to take a look on the RFID screen of the information point. He walks to demo's room. Although the demo has already started he can see in his mobile device contextual information about the speaker (belonging university group, related research and the complete demo show). Since John is not interested in the subject, he decides to leave. When he is located in the coffee room, he will be locatable for anyone who wants to talk to him. Indeed, he receives some messages in the cell phone from people that want to contact with him by some reason and therefore several informal talks take place without any compromising introductions. Among these messages he receives a suggestion to join a group (coalition) visiting a place in the city of particular interest this afternoon and a personal recommendation about an incoming talk. Therefore he accepts the invitation to the group and runs away to the conference hall where the interesting talk is going to take place in the next minutes. When he arrives to session room, his mobile device warn him about the presence of a colleague from his university, John is really surprised. Fig. 1 shows this scenario.

The above is a description that could be applied to any domain, such as fairground, shopping mall, tourist environment. All these heterogeneous domains are very rich in context information, which allows a context-based services provision.

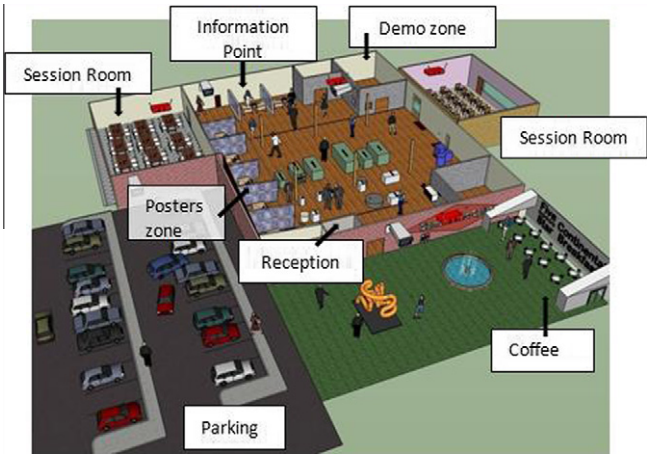

Fig. 1. Conference context scenario.

3. Design of MAS with BDI agents

3.1. Agents paradigm and methodologies used

The context-aware multi-agent system that we intend to design follows the BDI (Belief-Desire-Intentionl) paradigm by Rao Georgelf. BDI agents are built according to these three human reasoning-inspired knowledge levels, supporting reactive and proactive behavior. Beliefs stands for the knowledge that agents have about the environment and other agents, desires represent the objectives that each agent has, and finally, a set of intentions that could be created/updated/given-up if the environment changed. The BDI paradigm may then provide agent solutions facing a dynamic environment with a degree of uncertainty.

AUML, as extension of UML, facilitates to introduce BDI in Agent model. AUML is an attempt to move closely with the practical issues regarding agent system design. AUML constitutes the description language used to represent agent-specific concepts (Bauer et al., 2001).

AUML extends UML with the following issues: a special organized agent class, the new concept of role, the new Agent Interaction Protocol Diagram. But, how we explain in the introduction, we just use agent and interaction models from AUML.

GAIA is a methodology for multi-agent system design, which works in two phases: analysis and design. Analysis phase tries to understand the system and its structure through the next steps: (i) the organizational structure, (ii) the environment model, (iii) the preliminary role model, (iv) the preliminary interaction model and (v) organizational rules. On the other hand, design phase works over analysis models in order to implement agents (Wooldridge, Jennings, & Kinny, 2000). GAIA lacks of a requirement phase, and therefore we replace it by uses cases diagrams of Agent UML (AUML).

Particularly we follow the approach of Huhns (Huhns, 2004) who proposes the following steps to build the agents for our architecture: Group and role assignment, Services description, Protocol description, Event, Goals, plans and actions, Knowledge.

Particularly, the two last steps are included in agent diagram in our design. In advance, we show step by step the analysis and design of our multi-agent system.

3.2. GAIA agent role model of a context-aware system

The role model identifies the main roles for agents. Here a role can be viewed as an abstract description of an entity's expected function (Garro, 2012). Each role has associated attributes: responsibilities (a role is created in order to do something) and permissions (relating to the type and the amount of resources that can be exploited when carrying out the role).

According each agent model, we next detail each role-schema using GAIA Model in Table 1.

3.3. AUML agents model of a context-aware system

Agents are defined through agent's diagrams in the AUML methodology. Since we consider our agents to be cognitive/deliberative they were designed taking into account the following concepts: roles, services that offer, protocol description, plans, objectives, actions and knowledge (beliefs) (Huget et al., 2002).

Taking into account the agent roles described in the GAIA role model, we now describe agent characteristics forming AUML agent model (see Fig. 2) for the agents in the system: Locator, User Manager, Broker, Provider and User.

Locator Agent plays the role of user identification and user location inside the environment. To satisfy such desires, the locator agent has to connect with Appear Platform (Sánchez-Pi, Fuentes, Carbó, & Molina, 2007) that provides the location services. Also, this locator agent has to reason over spatial–temporal data. If the system required it, this agent would be a fusion data agent that connects with multiple locators agents that would work in different agent platforms. Its beliefs are: user position and the spatial–temporal data. Table 2 shows the AUML diagram of Locator Agent.

User Manager Agent has the desire of registering/deregistering each user agent that wants to run in the system. In order to accomplish such desire, it receives a request to register; it has then to analyze the user agent situation before registering it in the system. For instance, if the domain is about engineering conferences and a nurse want to register, her register request should be rejected.

Other activity or objective of the user manager agent is to update shared user profiles. User manager is in charge of managing the public profile of the user. Additionally, it has to manage the coalitions over shared interests. For example, if several user agents are interested in buying the last book of the Lord of the Rings, this user manager agent could group them to jointly request a better price from the provider. Other example is when people want to go to the same place; the user management agent could evaluate the minimal way for each one, and it could fix a meeting point. Table 3 shows the AUML diagram of User Manager Agent.

Broker Agent is the responsible of managing services, discovering and filtering providers. Also, it has to make the matching between user profile and the services that provider agent offers. It receives user agent position and according to the coordinates and zone where user agent is, the broker will filter out what provider agents are close. After that, the broker agent warns provider agent about a potential client. The beliefs of broker agent includes provider context information Services and provider agent's location. Its intentions are related to filtering providers' goal and applying matching goal. Table 4 shows the AUML diagram of Broker Agent.

Provider agents offer services to users and they negotiate with them. Their beliefs include their own contextual information and information about broker agent and the other providers that share their knowledge. One of the goals of provider agents is to offer services according to the user preferences, with this intention broker agent matches public user profiles with services offered by providers. The other goal of Provider agents is to achieve agreements with users through a negotiation process. The plans of these agents consist of dialogs with users to reach such goals. Table 5 shows the AUML diagram of Provider Agent.

Finally, Users agents can negotiate with providers, to reach agreements, and it can make recommendations to other users or propose forming coalitions with them. The user agent would begin requiring services or information.

After that, user agent would receive multiple responses from different agents (providers and users), and it would have to decide with whom it should start the negotiation process. User Agent is responsible to manage the private (internal) user profile, and it could improve its decisions according to the recommendations received. Its beliefs include the own profile information (public and private user profiles). Its plans are: to negotiate and reach agreements with providers, to dialog with users (for reputation, recommendation or form groups) and to manage profile (updating the own profile – learning preferences – and sharing information with other users). Table 6 shows the AUML diagram of User Agent.

3.4. Services model

The services model in Gaia methodology tries to identify the functions (services) associated with each agent role and to specific the main properties of these services (Garro, 2012). We must to identify inputs, outputs, pre and post-conditions of each service. This model is detailed bellow role by role.

Table 1
Gaia role model for a context-aware multi-agent system.

Role Schema: User Manager (UM)
Description This role is responsible for registering users, as well as improving their public profiles
Protocols and Activities Receive-registry-profile, agree-registry, register-user, deregister-user, receive-user-sequence, improve-user-profile
Permissions Reads user_profile, sequence Changes user_registry, user_profile
Responsibilities *Liveness:* UM = (Receive-registry-profile · Agree-registry · (Register-user|Deregister-user))^n|(Receive-user-sequence · Improve-user-profile)^n *Safety:* it is necessary to guarantee that no intruders get into the system

Role Schema: Location Manager (LM)
Description This role is responsible for locating and identifying users
Protocols and Activities Check-user-location, identify-user
Permissions Reads user_location
Responsibilities *Liveness:* LM = (Check-user-location · Identify-user)^n *Safety:* it is necessary to ensure the connection with the location system

Role Schema: Services Manager (SM)
Description This role carries out the matching between services offered by providers and user profile
Protocols and Activities Match-services-profiles
Permissions Reads provider_service, user_profile Changes matching_result
Responsibilities *Liveness:* SM = (Match-services-profiles)^n *Safety:* it is necessary the provider_service and user_profile would be available to apply the matching

Role Schema: Provider Discover (PD)
Description This role obtains close providers and communicates with them to warn them about the presence of a user
Protocols and Activities Filter-provider, warn-provider
Permissions Reads matching_result Changes communication_information
Responsibilities *Liveness:* PD = (Filter-provider · Warn-provider)^n *Safety:* if there is a success matching result, it is possible to communicate with the provider

Role Schema: Service Provider (SD)
Description This role is responsible for informing about services and reaching agreements with users after negotiation
Protocols and Activities Offer-service, request-negotiation, agree-negotiation, exchange-information
Permissions Reads agree-negotiation, information_exchange Changes services_offered
Responsibilities *Liveness:* SD = (Offer-service)^n|(Request-negotiation · Agree-negotiation · Exchange-information)^n · (Offer-service) *Safety:* it is necessary to negotiate first for reaching an agreement and exchanging information

(*continued on next page*)

Role Schema: Profile Manager (PM)

Description
This role is responsible for updating internal user profiles and it offers the possibility of sending a shared part to central agent

Protocols and Activities:
Update-internal-profile, send-shared-profile-registry

Permissions
Reads user_profile, recommend_information
Changes user_profile

Responsibilities
Liveness:
PM = (Update-internal-profile)^n|(send-shared-profile-registry)^n
Safety: it is necessary to receive external information, like recommendations to improve or update internal profile

Role Schema: Negotiate Role (NR)

Description
This role let agents negotiate and, according to this, receive new services or improved services

Protocols and Activities
Consult-information, receive-services, ask-for-agreements, receive-request-negotiation, exchange-information

Permissions
Reads negotiate_information
Changes services

Responsibilities
Liveness:
NR = (Consult-information)^n|(Ask-for-agreements · Receive-Request-negotiations · (Exchange-information)^n · Receive-services)^n
Safety: there is a negotiation phase for reaching agreements and receiving new services

Role Schema: Recommend Role (RR)

Description
This role offers the possibility to recommend information between users, and to decide who to trust for sharing opinions

Protocols and Activities
Recommend, ask-for-recommendations, decide-to-trust, receive-recommendation

Permissions
Reads recommendations
Changes opinion_to_others, user_profile

Responsibilities
Liveness:
UM = (decide-to-trust · Recommend)^n|(Ask-for-Recommendations · Receive-recommendation)^n
Safety: it is necessary to assist in the search information process of the user

Role Schema: Coalition Manager (CM)

Description
This role allows to solve the coalitions formation problem between users

Protocols and Activities
Receive-user-request, send-shared-information, group-formation

Permissions
Reads user requests

Responsibilities
Liveness:
CM = (Receive-user-request · Send-shared-information · Group-formation)^n
Safety: it is necessary to manage coalitions between users

Role Schema: Trust Manager (TM)

Description
This role is responsible of managing all kind of negotiation between user agents such as: recommendation information, reputation assessments, and trust relationships

Protocols and Activities
Search-activities/places-to-recommend,Management-reputation-mechanisms

Permissions
Reads user requests

Responsibilities
Liveness:
TM = (Search-activities/places-to-recommend · Management-reputation-mechanisms)^n
Safety: it is necessary to establish trust relationships between users in the system

First role is locator manager role that takes care of check user location and identify users services as it is shown in Table 7.

The User Manager role is related with the following services, showed in the corresponding table: request registry profile, agree

<<Agent>>
Role
State Diagram 1, State Diagram 2
Attribute 1, Attribute 2
[pre-cond]operation1 [post-cond] [pre-cond]operation2 [post-cond]
Capabilities, services
Perception1, Perception2
Protocol1:role, Protocol2:role
Organization1: role, Organization2: role

Fig. 2. AUML agent diagram.

Table 2
Locator Agent AUML Diagram.

Locator Agent
Roles – Location Manager
Services – Check-user-location – Identify-user
Beliefs – Location of Users Agents – Spatial–Temporal ontological data
Desires – Locate and identify users agents
Intentions – User position through location services

Table 3
User Manager Agent AUML Diagram.

User Manager Agent
Roles – User Manager – Coalition Manager – Search Process
Services – Request-Registry-profile: *User Manager* – Agree-registry: *User Manager* – Register-user: *User Manager* – Deregister-user: *User Manager* – Receive-user-sequence: *User Manager* – Improve-Profile: *User Manager* – Update-internal-profile: *User Manager* – Send-shared-profile-registry: *User Manager* – Form-user-agents-groups: *Coalition Manager*
Beliefs – Public information of user profile.
Desires – Register/Deregister users – Improve User Shared Profile – Form user groups
Intentions – Register users plan – Manage user profile – Manage coalition formation plan – Plan of recommendation or reputation actions

registry, register and deregister users, and improve user profile as Table 8 shows.

The service that is in relation with Services Manager is the match between user profile and the services that a provider offers (see Table 9).

Provider Discover role is responsible of filtering providers according user preferences and warning providers about user presence (as it is shown in Table 10).

Table 4
Broker Agent AUML Diagram.

Broker Agent
Roles – Services Manager – Provider Discover
Services – Match-Services-Profile: *Services Manager* – Filter-Provider: *Provider Discover* – Warn-Provider: *Provider Discover* – Search-information: *Services Manager*
Beliefs – Identification of User Agents – Location of Provider Agents
Desires – Detect Users – Filter providers
Intentions – Provider warning plan

Table 5
Provider Agent AUML Diagram.

Provider Agent
Roles – Service Provider
Services – Offer personalized services – Request negotiation – Agree negotiation – Exchange information
Beliefs – His own contextual information – Contextual information of Broker Agent and of others Providers Agents with shared knowledge
Desires – Communicate services – Reach a compromise with clients
Intentions – Offer services to users – Negotiate with users for reaching agreements

Table 6
User Agent AUML Diagram.

User Agent
Roles – Recommendation Role – Negotiation Role – Profile Manager
Services – Make recommendations to others users: Recommend role – Ask for recommendations: Recommend role – Receive recommendations: Recommend role – Decide to trust: Recommend role – Consult information: Negotiation Role – Receive services: Negotiation Role – Ask for agreements: Negotiation Role – Receive request negotiations: Negotiation Role – Exchange information: Internal Profile Manager
Beliefs – User profile information (private and public)
Desires – Negotiation between users and providers – Recommendation between users – Trust in other agents – Manage internal profile
Intentions – Negotiation Plan – Dialogue between users – Manage Profile Plan

Table 7
Services of locator manager role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Locator Manager (LM)				
Check-user-location	User location	User Location checked	Users connected	Users located
Identify-user	User location checked	User Identification	Users located	Users identified

Table 8
Services of user manager role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: User Manager (UM)				
Request-Registry-profile	User send a request message	Request for registry and user profile sent	User is in wireless network	Central Agent receive request from user
Agree-registry	Request for registry and user profile	Agree message and registry done	User send a request message	User Registered or not
Register-user	Proposed registry	Request-ted registry	Locate and identify users	User registered
Deregister-user	Proposed deregister or user out of the wireless network	Request deregister or user disconnected	User registered or connected to the wireless network	User deregister-red
Receive-user-sequence	External Information	Improved profile	Receive information from sensors etc.	The profile is improved
Improve-Profile	Information about user behavior	Improved profile	Receive information about user behavior	The profile is improved

Table 9
Services of services manager role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Services Manager (SM)				
Match-services-profiles	User profile known by central and provider information and location	Matching between user profile and provider information and location	User registered	The result of the matching

Table 10
Services of provider discover role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Provider Discover (PD)				
Filter-provider	Result of matching	Closer provider	The result of matching is valid	Provider filtered by location and information
Warn-provider	Filtered Provider	Provider received alert message	Obtain closer provider	Provider informs closer users

Table 11
Services of Service Provider Role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Service Provider (SP)				
Offer-service	Provider alerted to inform	Services offered according to location and user profiles	Closer provider warning	User receive services about his preferences
Request-negotiation	Provider request negotiation to user	Process of negotiation is requested	–	User accepts or rejects the request message
Agree-negotiation	Request message for negotiation	Agree message for negotiation or nothing	Request message	Negotiation Process initiated
Exchange-information	Agree message for negotiation	Information for exchange between provider and user	Agree message	Negotiation process finished

Service Provider role entails the next services: to offer services to users, to request and to agree negotiation with them, and to exchange information as it is shown in Table 11.

Next in Table 12, we represent the services that are related with Recommend Role. These are: asking for recommendations and receiving them, recommending users, deciding to trust.

Profile Manager role is responsible of updating internal profile and sending shared profile registry as it is shown in Table 13.

Negotiate role is in charge of the next services: consulting information about a offer received, receiving personalized services, asking for agreements, receiving request negotiation and exchanging information to finish the process negotiation (see Table 14).

Coalition Manager has to handle with receiving user request, sending share information and forming groups services (see Table 15).

Table 12
Services of recommend role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Recommend Role (RR)				
Recommend-users	Proposed of recommendation	Accept recommendation or refusal	–	Recommendation succeed or failure
Ask-for-Recommendations	Need of recommendation	Request recommendations	–	User agent receive recommendations or not
Receive-recommendations	Request recommendations	Recommendations received	Asked for recommendations	Recommendations received by user agent
Decide-to-trust	–	Decision to trust	–	Share opinions with other agents

Table 13
Services of profile manager role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Profile Manager (PM)				
Update-internal-profile	Other users recommendations	Profile updated	Received other users recommendations	The profile is updated
Send-shared-profile-registry	Need of registry	Send request message and profile to central agent	Users are connected to wireless network	User registered

Table 14
Services of negotiate role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Negotiate Role (NR)				
Consult-information	Propose of negotiation	Agreement or disagree-ment	–	Agreement reached or failure
Receive-services	Request for services or result of matching is valid	Services received	Matching done by central agent and provider is warned for offer services	Users obtain customized services and information
Ask-for-agreements	Need of agree-ments with other agent.	Request for reaching an agree-ment	–	Agreement accepted or rejected
Receive-request-negotiation	User agent ask for agree-ment to provider	Process of negotia-tion requested	User Agent have request to reach an agreement	Negotiation process is open (or not) for exchanging required information
Exchange-information	Request for negotia-tion	Information for exchange between provider and user	Ask for agreements and request for negotiation	Negotiation process finished

Table 15
Services of coalition manager role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Coalition Manager (CM)				
Receive-user-request	User agent requests	Searching user with similar needs	–	Other user are warn to make agreements
Send-shared-information	Users profile updates	–	Some user profile change	Coalition Manager update knowledge about public user profile
Group-formation	–	A new user group	Users with similar interests	Users group start negotiation process with provider agent

To finish with service model, the services related with Trust Manager role are the next ones: searching activities/places to recommend and managing the reputations mechanisms as Table 16 shows.

3.5. Interaction model

The building of AUML Interaction Protocol Diagrams make easier and it is closer to the system implementation in relation with GAIA interaction model. We choose AUML Interaction Diagrams to analyze our agent interaction. In the next figures, we can see these interaction diagrams for our context-aware system, detailing communication protocols and identifying agent roles.

3.5.1. Registering users

Initially, when a user wants to use the system, he has to request creating an account in the system. This is followed by the registration of user agent after filtering which user is going to use the system. It is a simple communication where the user agent (using internal profile manager role) requests a registration to the user management agent (with user management role). Fig. 3 shows this protocol for registering users.

3.5.2. Warning provider

This protocol is based in warn provider role of the broker agent. Broker Agent knows about all possible services in the environment and the providers' position. After locator agent returns the user

Table 16
Services of trust manager role using GAIA.

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Service Schema: Trust Manager (TM)				
Search-activities-places-to-recommend	User agent ask for recommendation	Recommendation information	Trust Manager has reputation values	User agent receive recommendations or not
Management-reputation-mechanisms	Receive updated user profile	–	Check if activity or place which is valorizing exist	Reputation values are updated

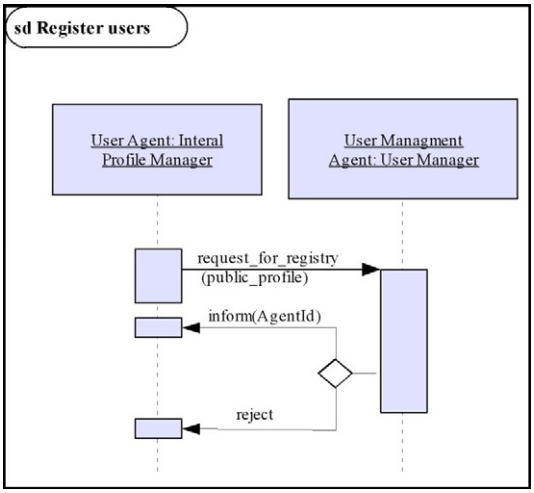

Fig. 3. AUML interaction diagram for register user service.

position to the user agent, it finally gives the position to Broker Agent. The broker agent may filter all providers that are close to user agent. Then it has to match user agent position and available service providers. Fig. 4 shows the protocol for warning providers.

3.5.3. Negotiation process

After provider was warned about user presence, a negotiation process could be started. The provider sends an offer to a user. When the user agent receives the offer he could reject it, accept

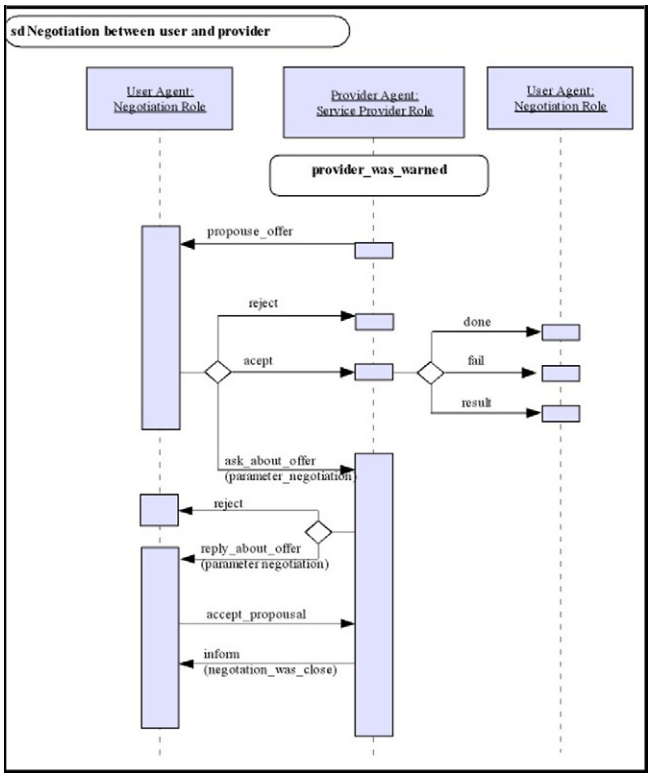

Fig. 5. AUML Interaction Diagram for Offer and Negotiation Process.

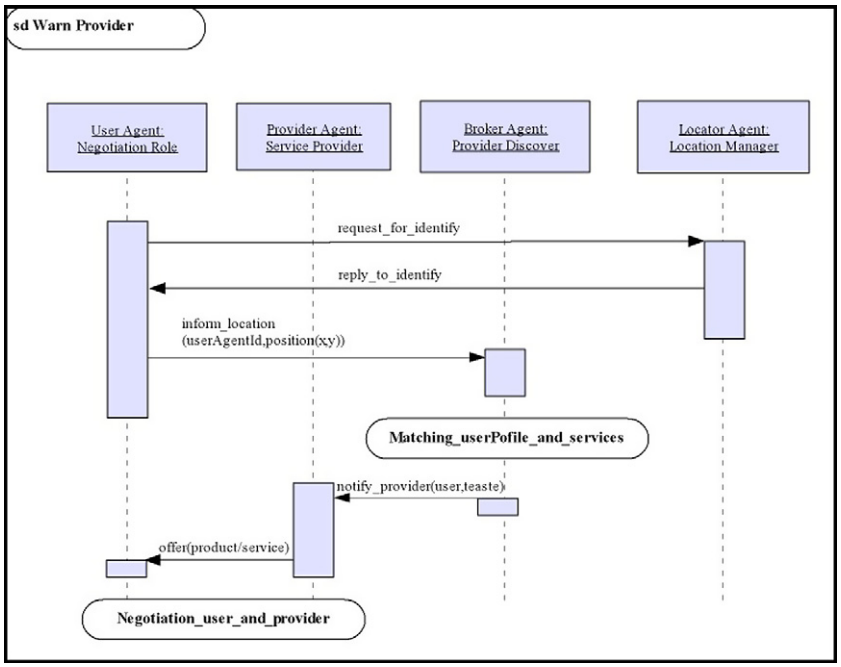

Fig. 4. AUML Interaction Diagram for Warn Provider Service.

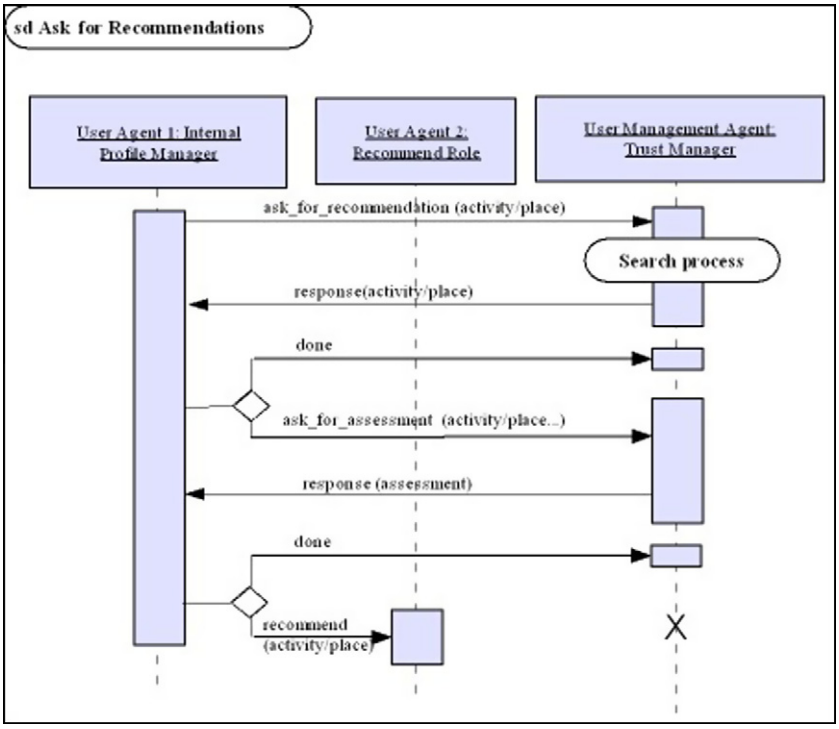

Fig. 6. Ask for Recommendations AUML Interaction Diagram.

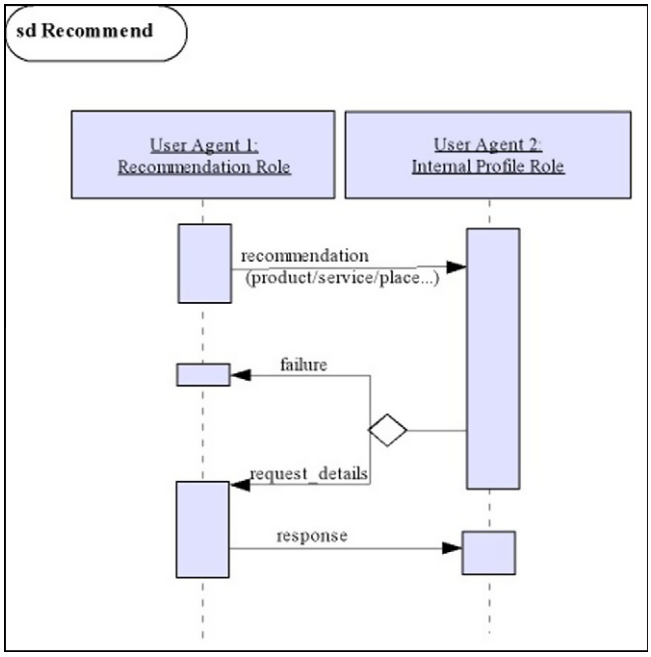

Fig. 7. AUML Interaction Diagram for Recommend service.

it or make an additional question about the offer. In the last step the provider agent is who rejects the agreement, replies user questions or accepts it. Fig. 5 shows the corresponding negotiation protocol.

3.5.4. Ask for recommendations

Sometimes a user agent may be interested in asking about recommendations. User agent sends a request to User Management Agent, who searches the corresponding information and responses with it. User agent could be satisfied and finish the process or, in other case, it could ask about a reputation value (assessment of the recommendation). Finally, the user agent could finish the process or it could start a recommendation process to other users. Fig. 6 shows the protocol for asking for recommendations.

3.5.5. Recommending

Once services are provided, user agents could proactively (de) recommend such services to other user agents (user agent 2). If such recommendation took place, user agent 2 could request additional information. Fig. 7 shows the protocol for recommending services.

3.5.6. Updating user profile

User agent may ask User Manager Agent for updating its own public profile. This is for example, when it consumed a given service or changed dynamically some personal preferences. User agent informs about the particular updating. Then, user Management agent could accept or reject the change according to its beliefs. Fig. 8 shows the protocol for updating user profiles.

3.5.7. Exchanging information

Provider agent may send information about its services or products to user agents with the goal to capture user interest. Fig. 9 shows the protocol for exchanging information.

3.5.8. Coalition formation problem

In some cases, agents might benefit from being aware of shared interests of other agents in the system (Merida Campos & Willmott, 2007). Agents have to explore a search space of agent group combinations in order to solve a problem in a coordinated manner, or to improve their outcomes by creating alliances. In AmI application this is a common problem. For example, user agents 1, 2 and 3 ask to the user manager agent about a shared issue. The user manager agent takes a decision and it informs about the shared issue to the user agents.

In other case, user agents 1, 2 and 3 send a request to user manager agent about which other agents are interested in a given issue.

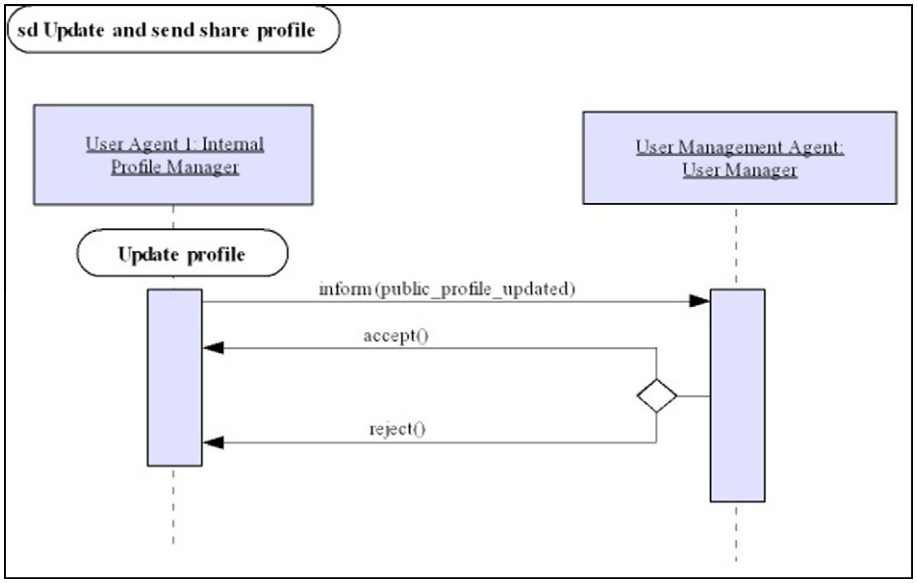

Fig. 8. Update and send share profile Interaction Diagram.

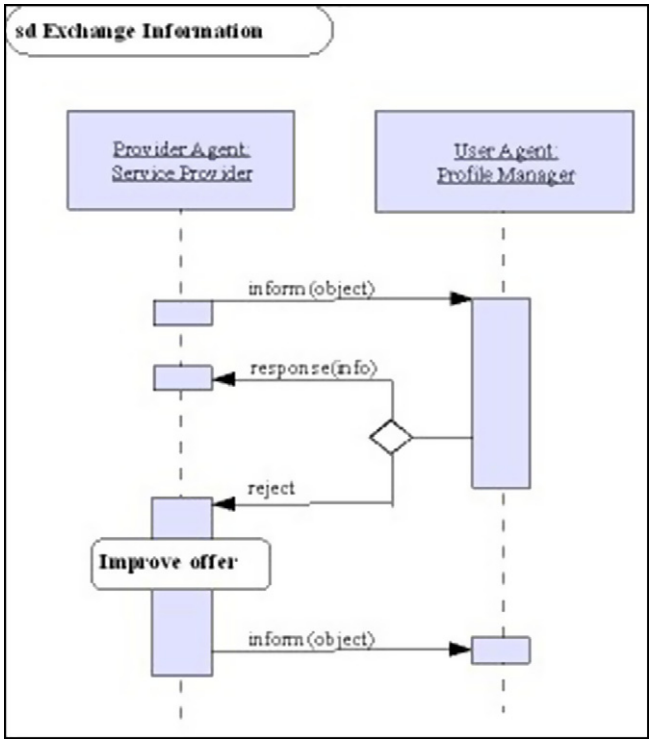

Fig. 9. AUML Interaction Diagram for Exchange Information.

User manager agent searches all agents with the same interest on that issue and, then, it sends the corresponding response to each agent.

Fig. 10 shows the protocol for coalition formation.

3.5.9. Deciding to trust

Since we cannot control the internal reasoning of each agent in the system, because they are autonomous, there is some risk if we trust everyone. This problem can be limited including a trust model to filter out interesting agents. In this way, agents have to include a model of other agents that allows them to decide if they can or cannot trust another agent. The trust decision process often uses the concept of reputation as the key factor a trust decision. Several agent reputation models have been proposed (Brandão, Vercouter, Casare, Sichman, & May, 2007).

In Fig. 11, user agent asks to user manager agent about reputation values of a third agent. It returns a valuation of this third party that the first user want to know if could trust according to the knowledge of the user management agent. This interaction implements a centralized trust of model since the user management agent is supposed to be applying universally accepted objective criteria. We plan to complement this centralized observation of the agents' behavior with a distributed trust model where agents communicate directly their subjective valuations (based on their own observations).

3.6. Collaboration diagram

Collaboration Diagrams representing instances of agent instead of agent roles. Following Odell's proposal (Odell, Van Dyke Parunak, & Bauer, 2000), we design a collaboration diagram where only interaction agents without roles are showed. This diagram allows us to obtain the finally schema of multi-agent system architecture. It is showed in Fig. 12: Each rectangle of the figure is an agent. The sequence of interactions is numbered on the collaboration diagram. Dot line represents the role playing by each user at the interaction moment.

4. MAS architecture

The proposed architecture is composed by the following agents: Providers, Users, Brokers, Locator and User Management. Fig. 13 shows an overview of this architecture.

In first instance, each user has to register into User Management Agent. User agent has to identify with the locator agent who returns the current position. Locator agent, depending on the system, could be a fusion data agent, in the case that the system would work with different location technology like wireless, wimax or ultrawideband. Locator agent is the one who reasons over spatial–temporal data. The user agent is going to request different services according to the user position. For these purposes, it has

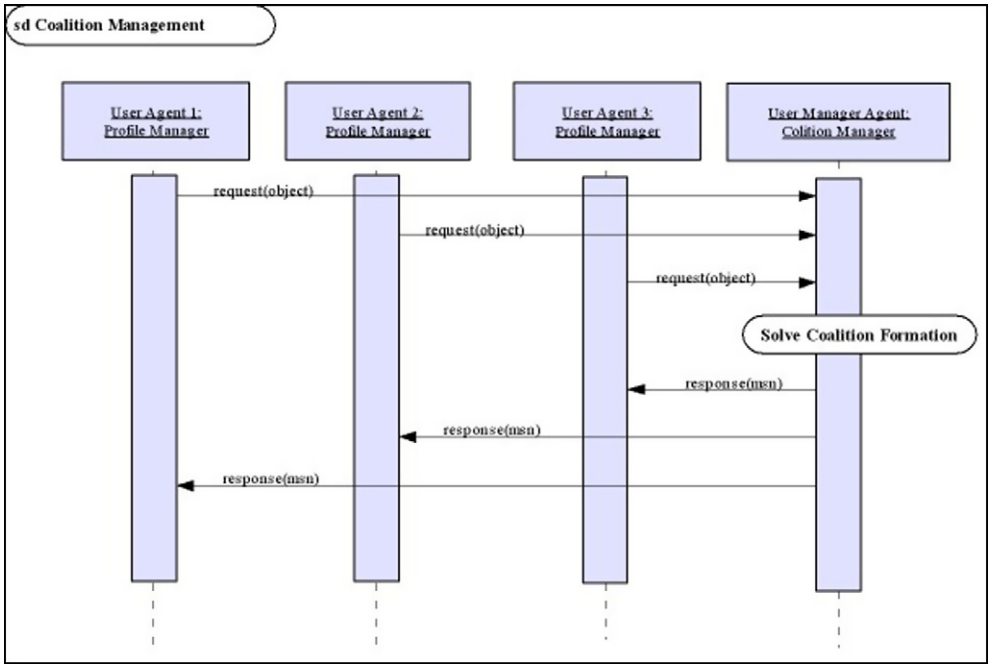

Fig. 10. AUML Interaction for Coalition Management.

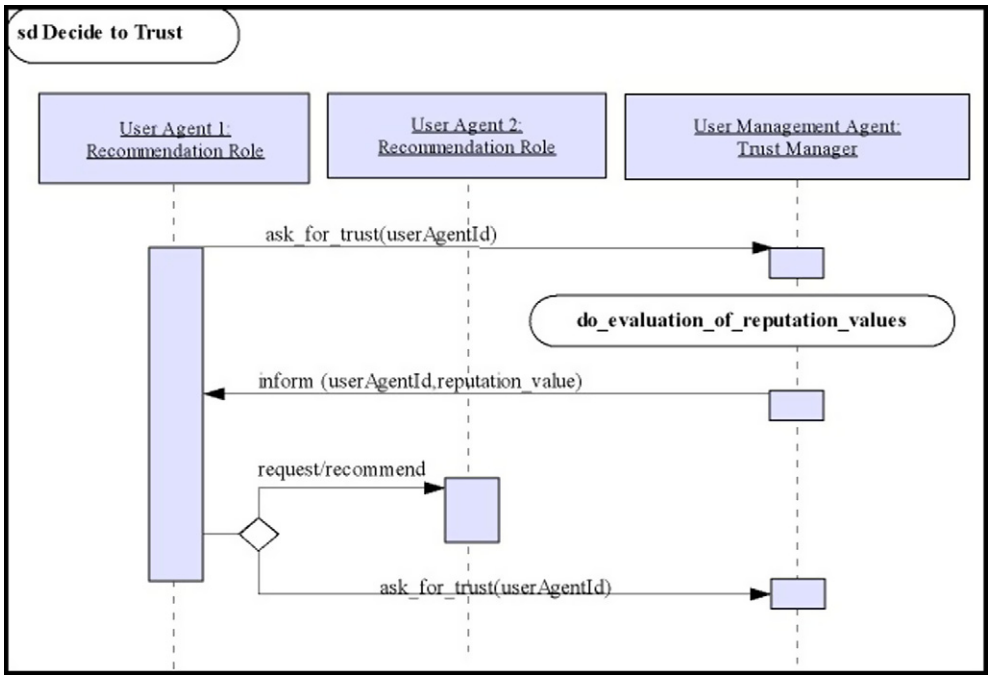

Fig. 11. Deciding to Trust AUML Interaction Diagram.

to communicate with Broker agent. This agent is the one who matches user's preferences and provider's services when both are in the same zone/area. Broker agent is also responsible of managing services and finding providers.

Once provider agent was warned about user preferences with a specific public profile, the negotiation process could be started. This process could be quite simple: a user request a product, to aim this goal, the user could ask about product's price. But this negotiation could be also about services themselves. Also, the provider could send offers without personalization (avoiding the matching process of broker agents), as the case of an user agent looking for general information.

An user agent can communicate with other user agents directly to negotiate or to ask about recommendations. Management user agent plays the main role in this operation. For example, if the design of the system included reputation information, User Management agent would then assume the role of matching users. In other case, it would be directly the user agent who proactively spreaded

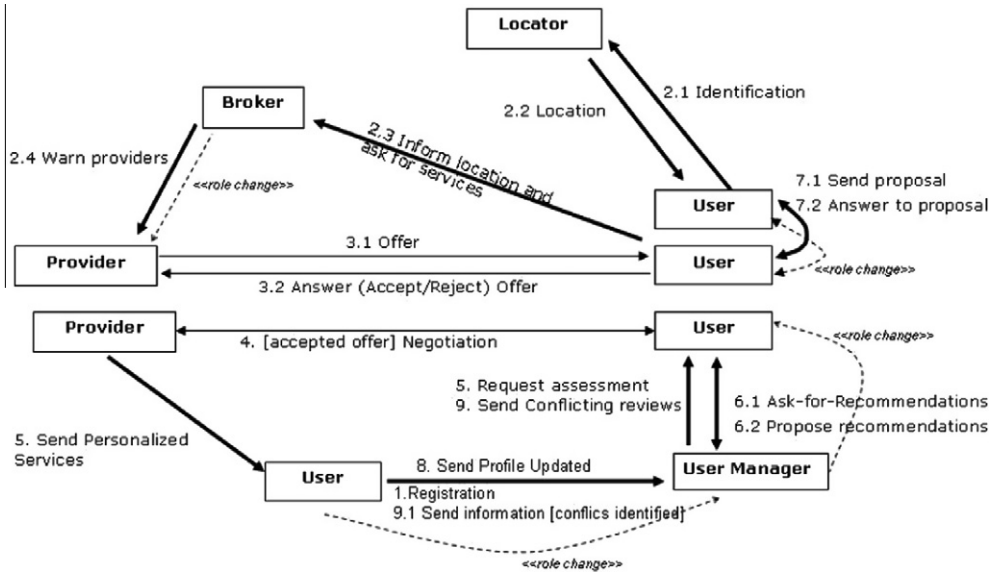

Fig. 12. Collaboration Diagram of AUML Methodology.

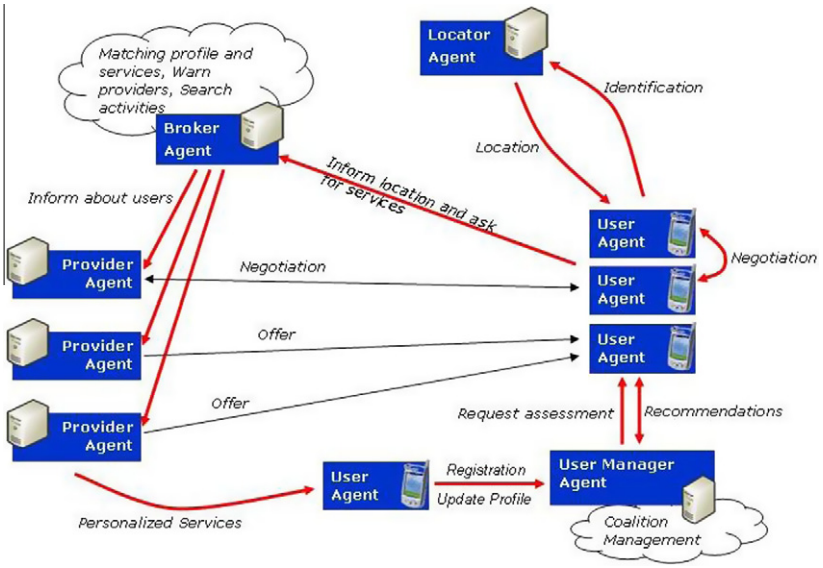

Fig. 13. Context-aware multi-agent system architecture for heterogeneous domain.

recommendations. Additionally, Management User agent could handle the agent coalitions over shared interests in a particular issue.

The framework is designed for a generic domain to support all activities that could be possible in the environment: trust models, negotiation, localization, profile management.

User Management, Broker and Location agents are intended to be running in one or different servers. Each provider agent may also be running over a different server. Finally, users agents are expected to be executed on mobile devices.

According to description of the problem of Section 2, we can observe how the architecture can be easily adapted to conference domain. Initially, locator agent establishes the current user position in the parking area. When he moves into the building, locator agent detects his position in the registration sector. At this moment, User Manager Agent asks user agent for registration. User agent sends then his current position in conference rooms or poster sessions to Broker agent. It therefore matches user profile of the listener and the information that each speaker will dictate. Next, user Agent receives messages of provider agent from session room one. Locator agent detects that user moves to coffee zone. User agent receives his position again. User manager agent knows about the situation and warns the other user agents about its presence. Some other user agents may then recommend talks, users, etc. A few minutes later, such user agent moves to the common area and receives a message in the mobile device from poster provider agent. Then, information about research line and other papers are showed in the mobile device. That it is possible because the user agent has sent his position to broker agent again, and it implemented a new matching. An user agent sends then a request to poster provider agent, and it starts a negotiation process about a possible joint publication/project. In the same form, the task of each agent could be adapted to any context. The use of a generic ontology facilitates such adaptation.

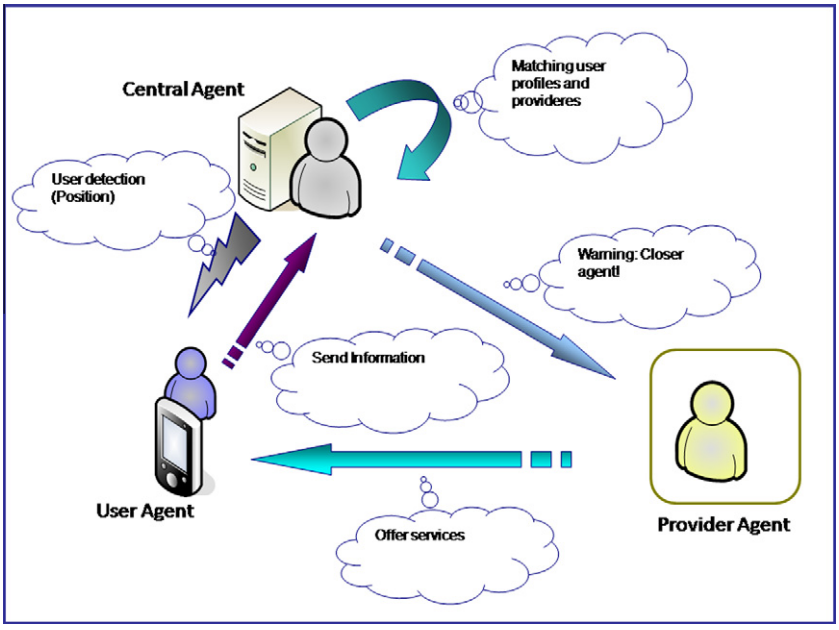

Fig. 14. MAS1 (Centralized architecture).

The implementation of the system was developed with Jade for agents whom runnings over servers while with Jade-Leap (Moreno et al., 2003) for user agents running in mobile devices. We also built an ontology for agent's communication with Protégé Tool.

5. Evaluation

5.1. Experimentation

We intend to evaluate our proposal through a pair-comparison of how they carry out several standard negotiation Protocols defined by FIPA.

We compare the agent-based centralized architecture exposed in Fuentes et al. (2007) (noted in advance as MAS1) with the architecture proposed in present work (MAS2).

In our evaluation both architectures include client agents receiving services offered by providers while brokers facilitate the matching between them. Both of them also use the same ontology (published in Fuentes et al. (2006)). The main differences between them are:

- The architecture of MAS1 (Fig. 14) has been largely tested in realistic scenarios and has generated several publications (Fuentes, Sanchez-Pi, Carbo, & Molina, 2006) involves three types of agents: Central Agent, Provider agents (one per each interest point) and User agents (one per user). Central Agent is in charge of locating users and matching user profile with the profile of each provider in the same zone/area. Then, the central agent warns a particular provider agent about the presence of a potentially interested user. Next, the negotiation process between Clients and Providers starts.
- On the other hand in our proposal a Locator Agent localizes and identifies User agents, while the Broker Agent matches the public User Profile with the services offered by Provider agents when User Agent requested Broker Agent to do it.

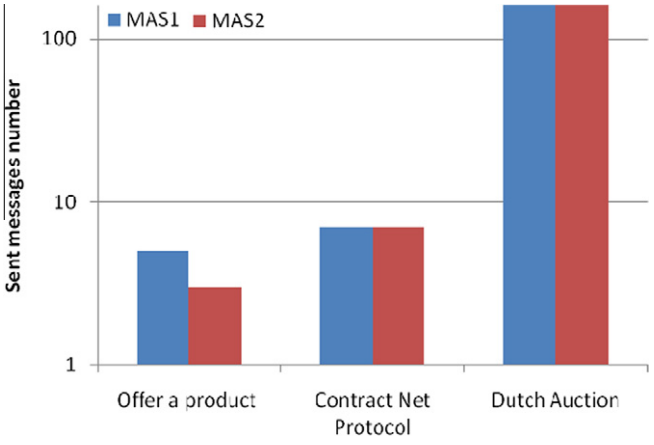

Fig. 15. Sent Messages Number.

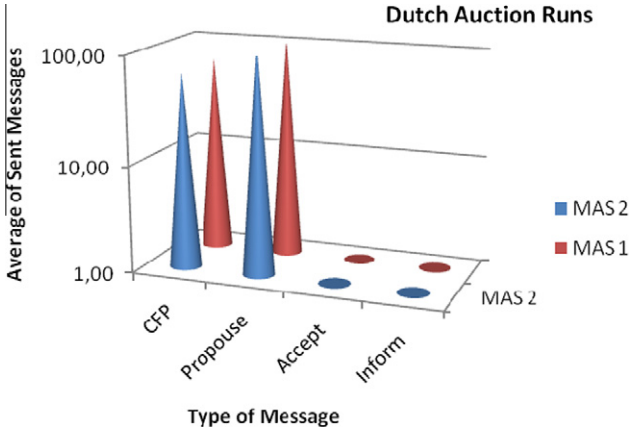

Fig. 16. Dutch Action Process for MAS1 and MAS2.

The protocols used to evaluate both architectures are first a set of them inspired in those defined by FIPA specifications: Product's Offer, Contract Net and Dutch Auction protocols, and later a reputation scenario defined adhoc by us was simulated to test the versatility of our architecture for recommendation services.

Table 17
Weight assigned for each type of message in negotiating protocols depending on their importance.

Type of message	Offer a product			Contract net protocol			Dutch auction		
	Number of messages	Weight	Results	Number of messages	Weight	Results	Number of messages	Weight	Results
MAS 1									
Inform	2	0.75	0.75	1	0.75	0.75	1	0.75	0.75
Propose	1	1	0	2	0.5	1	132.80	1	132.8
Accept	1	0.5	0	1	1	1	1	0.5	0.5
CFP	0	0	0	1	0.5	0.5	66.40	1	66.4
Request	1	0.75	0.75	0	0	0	0	0	0
Query If	0	0.5	0.5	0	0	0	0	0	0
Reject Proposal	0	0.5	0	2	1	2	0	0.5	0
MAS 2									
Inform	1	0.75	0.75	1	0.75	0.75	1	0.75	0.75
Propose	0	1	0	2	0.5	1	130.16	1	130.16
Accept	0	0.5	0	1	1	1	1	0.5	0.5
CFP	0	0	0	1	0.5	0.5	65.08	1	65.08
Request	1	0.75	0.75	0	0	0	0	0	0
Query If	1	0.5	0.5	0	0	0	0	0	0
Reject Proposal	0	0.5	0	2	1	2	0	0.5	0

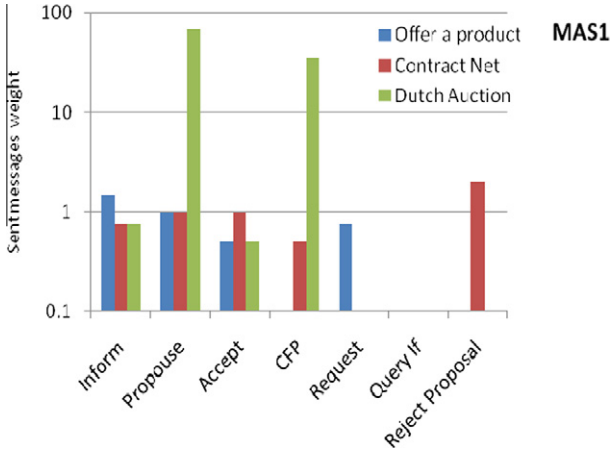

Fig. 17. Weighted exchanged messages in MAS1 simulations.

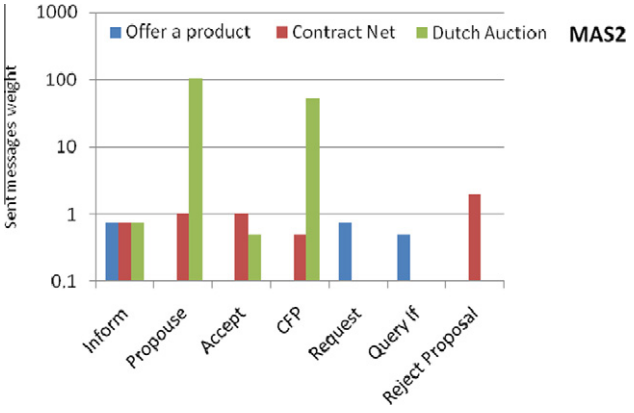

Fig. 18. Weighted exchanged messages in MAS2 simulations.

Both architectures addressed equal conditions: same initial parameters and situation, same hardware and same procedure.

5.2. Evaluation method

Evaluating Agent Architectures is a complex task, and a few evaluation methods have been defined until now (Davidsson, Johansson, & Svahnberg, 2006, 2008). It is worth to mention that none of these evaluation methods cover all the aspects in the architecture evaluation. The purpose of our evaluation is to apply the evaluation method of Joumaa, Demazeau, and Vincent (2008) that allows us to compare architectures from the communicative point of view. This method suggested an evaluation based on the weight of the information brought by messages.

5.3. Evaluating FIPA negotiation protocols with MAS1 and MAS2

Fig. 15 just presents the total amount of exchanged messages between agents for both architectures MAS1 and MAS2 when Product's Offer, Contract Net negotiation and Dutch Auction were executed.

Stochastic data were used in the Dutch Auction simulations. For this reason, we present in Fig. 16 several runs of this protocol with the average value presented in Fig. 15.

Next, we consider all possible type of messages involved in these protocols according to their performatives: Inform, Propose, Accept-Proposal, Call-for-Proposal (CFP), Request, Query-If and Reject-Proposal.

Table 18
Weight assigned corresponding to performatives.

Type of message	Pertinent	Weight
Inform	Yes	1
Propose	No	0
Accept	No	0
CFP	Yes	0
Request	Yes	0.75
Query If	No	0
Reject Proposal	No	0

A value among "high" (1), "medium" (0.75), "low" (0.50) and "not pertinent" (0) has to assigned to each type of message depending on the importance of messages in the simulation (Joumaa et al., 2008).

The assignment criteria of these weights we chose to each type of message were: If the behavior tried to convince the user about any service, then the maximum value would be assigned to CFP or Propose message. Otherwise, if the interest in the negotiation fell on the User Agent, the maximum value would be assigned over final result of the operation, i. e., accept or reject messages. The final weight values are shows in Table 17. These values were used to compute the aggregated weight of each type of message in each evaluated protocol (for MAS1 in Fig. 17 and for MAS2 in Fig. 18).

Table 19
Weights assigned for each type of message in recommendation protocols depending on their importance.

MAS 2	1 Reputation request to 4 users			2 Reputation request to 3 users		
Type of message	Number of messages	Weight	Results	Number of messages	Weight	Results
Inform	4	1	4	6	1	6
Propose	0	0	0	0	0	0
Accept	0	0	0	0	0	0
CFP	0	0.5	0	0	0.5	0
Request	4	0.75	3	6	0.75	4.5
Query If	0	0	0	0	0	0
Reject Proposal	0	0	0	0	0	0

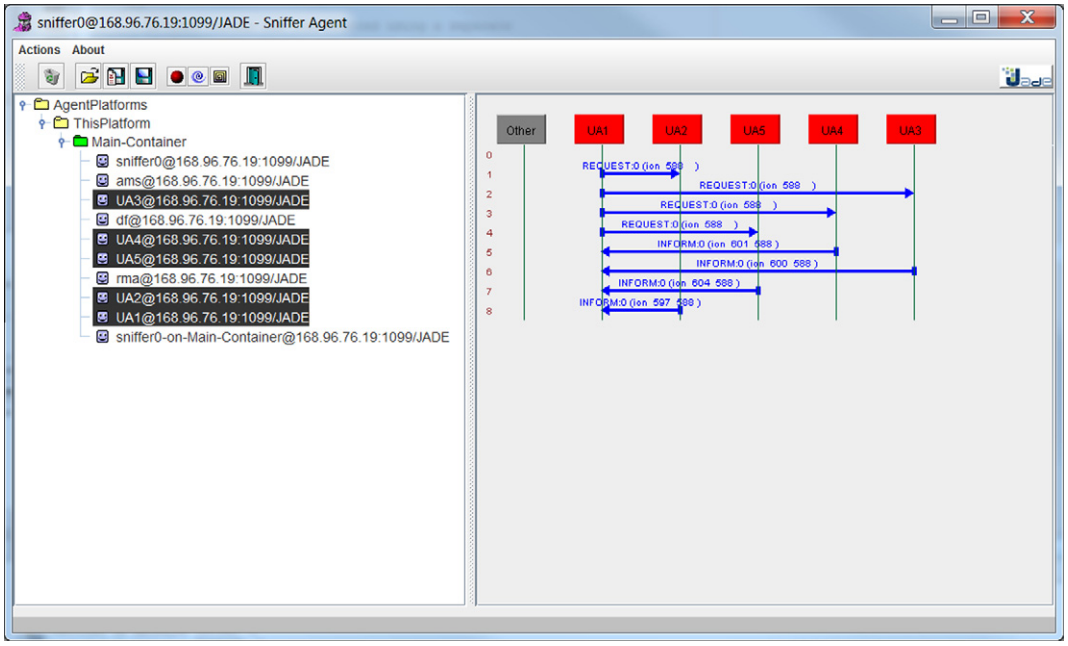

Fig. 19. User agent 1 asks about reputations of a provider.

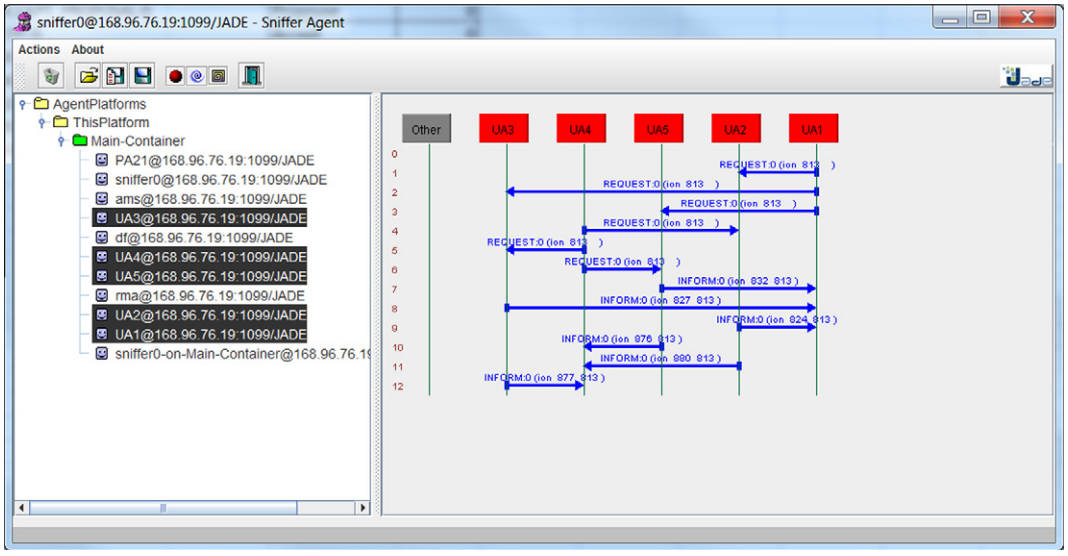

Fig. 20. User agent 1 and 4 ask about reputations of a provider.

Then, from these data, it is inferred that the difference between the two systems, when performing the comparison on the number of messages sent is minimal. Therefore, MAS2 is valid alternative. Additionally since our present proposal (noted as MAS2) corresponds to an Asynchronous Distributed System in contrast to MAS1 that it is an Asynchronous Centralized architecture (with the corresponding desirable properties of robustness and scalability of distributed over centralized systems). MAS2 may represent a better alternative.

5.4. Evaluating an adhoc reputation scenario

To introduce this additional experimentation case, the interactions over a trust model were evaluated. Assigning weights to this reputation scenario opens the possibility of comparing other context-aware systems including trust models with our agent architecture.

For instance, once a potential client Thomas House receive a notification about Arbys provider agent.

Thomas agent contacts other agents registered in the system. He asks for recommendations about Arbys' reputation. Each user agent sent to Thomas a message with its own opinion over such provider. Then Thomas would aggregate all opinions into an Arbys' reputational image to decide about requesting the services and negotiating it.

In order to assign weights to each type of messages, for recommendation system the maximum value 1 was given to inform message and the value 0.75 was for request message. It is shown in Table 18. The other types of messages were not relevant for this example.

In the case of MAS2, two cases are evaluated. One of them takes one user agent only and five recommenders. The other covers two users and four recommenders. Both request over a single provider agent. Table 19 shows the amount of messages for each execution.

Figs. 19 and 20 show image captures of running simulations of both recommendation cases.

6. Conclusions and further work

We have analyzed the principal characteristics from several AmI scenarios and we have defined a generic framework to context-aware and location-based multi-agent system. We have focused in agents' architecture design, showing the methodological steps followed. This model was built using GAIA and AUML methodologies. The resulting architecture becomes flexible to any service that could be included in AmI environment. Of particular interest is the inclusion of a locator agent that would be a data fusion agent if connections with several kind of location systems were needed. And another innovative feature of our contribution is that different mechanisms of reputation or recommendation were included.

Finally evaluating these complex systems was addressed comparing our proposal with an already tested centralized agent-based alternative architecture. We did it considering agent interactions as the most important characteristic of agent systems (Wooldridge & Ciancarini, 2001). So, in this paper, we described the assignment of weight values to agents interaction in two different MAS architectures for Context Aware problems facing FIPA standard negotiation protocols: Product's Offer, Contract Net and Dutch Auction protocols.

Finally we also implemented this evaluation method to adhoc recommendation scenarios to show the versatility of our architecture and to facilitate future comparisons of this application of trust models in context-aware systems.

Finally, we conclude that communication technologies are the actual revolution which needs software advances. In this sense, and in relationship with sensors and actuators in the environment, ambient intelligence applications will be very soon assisting human in all activities: daily life, tourism, education, health care, etc. We have contributed to such final goal with a generic agent architecture that provides context-aware services, that was designed following an hybridizing methodology between GAIA and AUML. We also compared our architecture with an already published alternative through an evaluation method based on the relevance of exchanged messages, and we showed the applicability of using trust models in such Context Aware Systems with two cases of use.

References

Bauer, B., Muller, J. P., & Odell, J. (2001). Agent UML: A formalism for specifying multiagent interaction. In P. Ciancarini & M. Wooldridge (Eds.), *Agent-oriented software engineering* (pp. 91–103), LNCS 1957, Berlin: Springer-Verlag.

Bombara, M., Cali, D., Santoro, C. (2003). Kore: A multi-agent system to assist museum visitors. In *Workshop from objects to agents, Villasimius, (WOA 2003), CA, Italy* (pp. 175–178).

Brandão, A. A. F., Vercouter, L., Casare, S., & Sichman, J. Exchanging reputation values among heterogeneous agent reputation models: An experience on ART testbed, May 14–18, 2007, AAMAS'07, Honolulu, Hawai'i, USA.

Bravo, J., Hervás, R., Nava, S., Chavira, G., & Sanz, J. (2005). Display-based services through identification: An approach in a conference context. In *The ubiquitous computing and ambient intelligence (UCAmI'05), CEDI'05*. Granada, Spain.

Chen, H., Finin, T., & Joshi, A. (2003). An intelligent broker for context-aware systems. In *Adjunct proceedings of Ubicomp 2003* (pp. 183–184). UbiComp.

Corchado, J. M., Bajo, J., de Paz, Y., & Tapia, D. I. (2008). Intelligent environment for monitoring Alzheimer patients agent technology for health care. *Decision Support Systems, 44*(2).

Davidsson, P., Johansson, S., & Svahnberg, M. (2006). Characterization and evaluation of multi-agent system architectural styles. In *Software engineering for multi-agent systems IV. Lecture notes in computer science*. Springer-Verlag.

Dey, A. K., Abowd, G. D., & Salber, D. (2001). A conceptual framework and toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction (HCI) Jorunal, 16*(2-4), 97–166.

Fuentes, V., Carbó, J., & Molina, J. M. (2006). Heterogeneous domain ontology for location based information system in a multi-agent framework. In E. Corchado, H. Yin, V. Botti, & C. Fyfe (Eds.), *Intelligent data engineering and automated learning, 20–23 Setember, 2006, IDEAL 2006, Burgos, Spain. Lecture notes in computer science* (Vol. 4224, pp. 1199–1206). Springer-Verlag.

Fuentes, V., Sanchez-Pi, N., Carbo, J., & Molina, J. M. (2006). Reputation in user profiling for a contextaware multiagent system. *4th European workshop on multi-agent systems* (*EUMAS'06*), Portugal: Lisbon.

Fuentes, V., Sánchez, N., Carbó, J., & Molina, J. M. (2007). Generic context-aware BDI multi-agent framework with GAIA methodology. In *International workshop on agent-based ubiquitous computing, international joint conference on autonomous agents and multi-agent systems, May 14–18, 2007 (AAMAS 2007)*. Hawai, USA.

Garro, A. (2012). Modeling-notation source: GAIA, version 03-03-12 17:25EST. Document author – Alfredo Garro. <http://www.fipa.org/>.

N. Glaser. The CoMoMAS Approach: From Conceptual Models to Executable Code. In: Modeling Autonomous Agents in Multi-Agent Worlds (MAAMAW), Ronneby (Sweden), 1997.

Gu, H., Shi, Y., Xu, G., & Chen, Y. (2005). A core model supporting location-aware computing in smart classroom. *International Journal of Computer Science and Network Security., 6*(3), 161–168.

Gu, T., Pung, H. K., & Zhang, D. Q. (2004). A middleware for building context-aware mobile services. *IEEE vehicular technology conference (VTC – Spring 2004)* (Vol. 5, pp. 2656–2660). IEEE Computer Society.

Hofer, T., Schwinger, W., Pichler, M., Leohartsberger, G., & Altmann, J. (2003). Context-awareness on mobile devices – The hydrogen approach. In *Proceedings of the 36th Hawaii international conference on system sciences (HICSS'03)*. IEEE Computer Society.

Huget, M. P. (2002). Agent UML class diagrams revisited. In *Proceedings of agent technology and software engineering. Erfurt, Germany, 2002*. United Kingdom: Agent ART Group, University of Liverpool.

Huhns, M. N. (2004). Agent UML notation for multiagent system design. *IEEE Internet Computing*, 63–71.

Joumaa, H., Demazeau, Y., & Vincent, J. (2008). Evaluation of multi-agent systems: The case of interaction. In *Proceedings of the 3rd international conference on information and communication technologies: From theory to applications, April 2008, Damascus, Syria*. IEEE Computer Society Publisher.

Kjeldskov, J., & Paay, J. (2005). *Just-for-us: A context-aware mobile information system facilitating sociality. Mobile HCI 2005*, Salzburg, Austria.

Martin, D. L., Cheyer, A. J., & Moran, D. B. (1995). The open agent architecture: A framework for building distributed software systems. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95), USA*.

Merida Campos, C., & Willmott, S. (2007). Exploring social networks in request for proposal dynamic coalition formation problems. *5th international central and eastern European conference on multi-agent systems* (*CEEMAS*) (pp. 143–152).

Moreno, A., Valls, A., & Viejo, A. (2003). *Using JADE-LEAP to implement agents in mobile devices. TILAB "EXP in search of innovation, 2003, Italy*. <http://jade.tilab.com/papers-exp.htm>.

Nakashima, H. (2007). Cyber assist project for ambient intelligence. In J. C. Augusto & D. Shapiro, (Eds.), Advances in ambient intelligence (Vol. 164, pp. 1–20).

Odell, J., Van Dyke Parunak, H., & Bauer, B. (2000). Extending UML for agents. In G. Wagner, Y. Lesperance, & E. Yu (Eds.), *Proc. of the agent-oriented information systems workshop at the 17th national conference on artificial intelligence* (pp. 3–17), Austin, TX.

Paganelli, F., Bianchi, G., & Giuli, D. (2006). A context model for context-aware system design towards the ambient intelligence vision: Experiences in the eTourism domain. In *Procedding of 9th ERCIM workshop "user interfaces for all", Königswinter (Bonn), Germany, 27–28 September, 2006*. Florencia, Italia: Departamento de Electrónica y Telecomunicaciones, Universidad de Florencia (pp. 173–191).

Pavón, J., & Gómez-Sanz, J. (2003). Agent oriented software engineering with INGENIAS. In V. Marik, J. M. üller, & M. Pechoucek (Eds.), *Multi-agent systems and applications II, 3rd international central and Eastern European conference on multi-agent systems (CEEMAS'2003). LNAI* (pp. 394–403). Springer-Verlag.

Sánchez-Pi, N., Fuentes, V., Carbó, J., & Molina, J. M. (2007). Knowledge-based system to define context in commercial applications. In *Proceedings of the 8th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing, July 30 – August 1, 2007, SNPD 2007* (pp. 694–699). Qingdao, China.

Sorensen, C., Wu, M., Sivaharan, T., Blair, G. S., Okanda, P., Friday, A., et al. (2004). A context-aware middleware for applications in mobile ad hoc environments. In *Proceedings of the 2nd workshop on middlerware for pervasive and ad hoc computing* (pp. 107–110). New York: ACM.

Spanoudakis, N. I., & Moraitis, P. (2006). Agent based architecture in an ambient intelligence context. In *Proceedings of the 4th European workshop multi-agent systems (EUMAS'06)* (pp. 163–174).

Tiba, F. K., & Capretz, A. M. An overview of the analysis and design of SIGMA: Supervisory intelligent multi-agent system architecture. In *Information and communication technologies, 24–28 April, 2006, ICTTA'06* (Vol. 2). Canada: University of Western Ontario.

Wooldridge, M., & Ciancarini, P. (2001). Agent oriented software engineering: The state of the art. In *First international workshop on agent-oriented software engineering*. Springer.

Wooldridge, M., & Jennings, N. R. (1995). Agent theories, architectures, and languages. In M. Wooldridge & N. R. Jennings (Eds.), *Intelligent agents* (pp. 1–22). Springer-Verlag.

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems, 3*, 285–312.

Wooldridge, M., Jennings, N. J., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems, 3*(3), 285–312.

Wright, D., Gutwirth, S., Friedewald, M., Vildjiounaite, E., & Punie, Y. (2008). *Safeguards in a World of AmI, 2008*, 10th ed. Springer (p. 291).

Yamabe, T., Takagi, A., & Nakajima, T. (2005). Citron: A context information adquisition framework for personal device. In *Proceedings of the 11th international conference on embedded and real-time computing systems and applications (RTCSA'05)* (pp. 489–495). IEEE Computer Society.