An Ambient Intelligent Platform based on Multi-Agent System

Verónica Venturini
Carlos III University of Madrid
Av. De la Universidad Carlos III
(28270) Colmenarejo, Madrid
0034916249105
veronica.venturini@uc3m.es

Javier Carbó
Carlos III University of Madrid
Av. De la Universidad Carlos III
(28270) Colmenarejo, Madrid
0034916249105
javier.carbo@uc3m.es

José Manuel Molina
Carlos III University of Madrid
Av. De la Universidad Carlos III
(28270) Colmenarejo, Madrid
0034916241314
josemanuel.molina@uc3m.es

Abstract

Researches on Ambient Intelligent and Ubiquitous Computing in connection to wireless technologies have been increasing in the past years. Multi-agent systems (MAS) are the ideal technology to support these applications. Nevertheless, only few models of MAS have been under evaluation. In this work, we evaluate a multi-agent architecture that may support the information needs of the new technologies, for heterogeneous domains. This architecture is evaluated using several method described by different authors.

1. Introduction

Ambient Intelligent (AmI) is a phrase used to describe the intelligence embedded in practically any object around us. In this new world of the intelligent software, computers supervise our activities, routines and behaviours to predict what the user wants to do in a particular time [19]. In effect, benefits of AmI have been shown in different environments in the last years such as social environments [10], education [2], tourism [13], medicine [4], etc. Aml works in relationship to Context-Aware Systems. There are different platform of Context-Aware Systems. CoBRa [3] is an agent-based architecture that works with a context broker agent. This centralized agent maintains a context model for the community. Other relevant context platforms are Hydrogen [8], Cortex [16] and Citron [20].

According to the user position and current activities, Context-aware systems have to provide services in correlation to the user preferences. Aml is an ideal workspace for agent technology given their autonomous nature. Taking into account all these features, the contribution of this paper is the evaluation of a context aware and location-based MAS architecture that can be used in different intelligence domains. Several multi-agent architectures were proposed to support Context-aware system. We can classify the MAS for specific or generic domains. Some examples of first case are: ALZ-MAS, a MAS for care health [4]; ASK-IT [17], a system that provides to the tourist a personal travel agent to assist him in recreational activities. In other way, for generic domains, CONSORTS [12] is an ad hoc architecture for Ambient Intelligent. Another example of generic architecture is the work presented in [6], this MAS centralize the tasks in one broker agent.

Most of these MAS models are not evaluated. However, MAS evaluation is booming in these days making the presented evaluation methods in this paper a proper way of analyzing our MAS architecture. The propose architecture is evaluated using several method described in [11, 5]. The method described in [11] analyzes the system take into account the communication between agents, i.e.; it counts the sent message number in a particular process. Then, it assigns a value to each type of messages according to the importance of the message for the agent. The work in [5] reviews the multi-agent architecture from the type of system and its characteristics point of view, i.e., if it is centralized or distributed. Hence, [5] proposes a table of value over each characteristic respect to the system to see which characteristics are the best for a given architecture.
In the present paper, we evaluate our distributed architecture in comparison to a centralized architecture described in [6].

The structure of this paper is the following: section 2 includes the MAS design, section 3 shows the architecture proposed, architecture evaluation is shown in section 4, and finally, we conclude in section 5.

2. Design of MAS in AmI Scenario

To build our multi-agent architecture for context-aware and location-based system, we use an Agent Methodology. After the study of different methodologies such as SIGMA [18], INGENIAS [14], GAIA [6] y Agent UML (AUML) [9], we concluded that GAIA is more suitable for Agent Development. However, GAIA methodology does not allow a direct implementation, apart from losing the relationships between its diagrams. Thus, we hybridize GAIA and AUML. Huhns [9] explains the steps of AUML methodology. It is worth to note that some authors consider AUML as a language of modelling and not as a methodology. The present work uses AUML agent and interaction models with the role and services models that GAIA provides. The AUML agent model summarizes and relates all the models used in AUML methodology, while the AUML interaction model (with GAIA support) allows the system implementation easier for the developer.

AUML constitutes the language used to represent agent-specific concepts [9]. AUML extends UML with the following features: a special organized agent class, the new concept of role, and the new Agent Interaction Protocol Diagram. On the other hand, GAIA is a methodology for MAS design, which works in two phases: analysis and design. The analysis phase tries to understand the system and its structure. This phase include the following steps: (i) the organizational structure, (ii) the environment model, (iii) the preliminary role model, (iv) the preliminary interaction model and (v) the organizational rules. The design phase handles the analysis models in order to implement the agents [6]. GAIA does not consider the requirement phase; this is solved using case diagrams of Agent UML (AUML). In the following subsections, we present a step-by-step design of our MAS.

2.1. Role Model

The role model is used to identify the main roles of each agent. Here, a role can be seen as an abstract description of an entity expected function. 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 to each agent model, we detail each role using GAIA Model. The agent’s roles identified in the system are: User Manage, Location Manager (LM), Services Manager (SM), Provider Discover (PD), Service Provider (SD), Profile Manager (PM), Negotiate Role (NR), Recommend Role (RR), Coalition Manager (CM) and Trust Manager (TM).

2.2. Agents model

Agents are defined through agent’s diagrams in the AUML methodology. We consider our agents to be cognitive/deliberative agents as we show in our agent model. Deliberative agents are designed taking into account the following concepts: roles, services that offer, protocol description, plans, objectives, actions and knowledge (beliefs). Taking into account the role described in role model, we describe agent characteristics to show the agent model.

Providers Agents (PA) offer services to users and negotiate with them. They believe in their own contextual information and in the information of the broker agent and the other providers that share their knowledge. One of the goals of the PAs is to offer services according to the user preferences, for this propose broker agent matches them with the user profile. Other objective, of PAs, is to achieve agreements with the users in the negotiation process. The plans of these agents consist of dialogs with users to reach such goals.

Users Agents (UA) can negotiate with providers, to reach agreements, and can make recommendations to other users or negotiate with them. The UA could require certain services or information for this propose. After that, the UA could receive multiple offers from different providers and it has to decide with whom it should
start the negotiation process. The UAs are responsible for managing internal profiles, and they could improve their profile according to the recommendations received. They believe in their own profile information (public and private information). Their plans are: to negotiate and reach agreements with providers, to dialog with users (for reputation, recommendation or form groups) and to manage profile plan.

Locator Agent (LA) (Table 1) plays the role of user identification and user location into the environment. For this, the LA has to connect with Appear Platform [15], which provides the location services. Also, this LA has to reason over spatial-temporal data. If the system required it, this agent could be a fusion data agent that connects with multiple LAs that work in different platforms. These beliefs are: user position and the spatial-temporal data (both defined on the ontology). The objectives or desires of the LA are: the location and identification of UAs. To aim these goals the LA is going to contact with a platform that would provide location services.

<table>
<thead>
<tr>
<th>Locator Agent</th>
<th>Roles</th>
<th>Location Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
<td>Check-user-location</td>
<td>Identify-user</td>
</tr>
<tr>
<td>Beliefs</td>
<td>Location of Users Agents</td>
<td>Spatial-Temporal ontological data</td>
</tr>
<tr>
<td>Desires</td>
<td>Locate and identify users agents</td>
<td></td>
</tr>
<tr>
<td>Intentions</td>
<td>User position through location services</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Locator Agent AUML Diagram

User Manager Agent (UMA) has to register/deregister each UA that wants to run in the system. After receiving a request to register, it has to analyze the user agent situation before register it in the system. For instance, if the domain is about engineering conferences and a nurse want to register, her register could be rejected. Other activity of the UMA is to update user shared profiles. It is the one who manages the public profile of the user. Public profile is its belief (which was introduced when a user registered). Also, it has to manage the coalitions over shared interests.

Broker Agent (BA), is the responsible for managing services, discovering and filtering providers. Also, it has to match the user profile with the services that PA offers. It receives the UA position and according to the coordinates and zone where UA is, the BA will filter which PAs are the closest. After that, the BA warns the PA about a potential client. The beliefs of the BA consist of provider context information as offered services and provider agent’s location. The intentions that this BA has are related to filtering providers’ goal and applying matching goal.

2.3. Services Model

The services model in GAIA methodology tries to identify the functions (services) associated with each agent role, and to specify the main properties of these services [6]. We must identify inputs, outputs, pre and post-conditions of each service. Next, Table 2 shows an example of services model for Profile Manager Role using GAIA.

<table>
<thead>
<tr>
<th>Service Schema: Profile Manager (PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
</tr>
<tr>
<td>Update-internal-profile</td>
</tr>
<tr>
<td>Send-shared-profile-registry</td>
</tr>
</tbody>
</table>

Table 2. Services of Profile Manager Role using GAIA

2.4. Interaction model

The building of AUML Interaction Protocol Diagrams makes easier and it is closer to the system implementation in relation with GAIA interaction model. We choose Interaction Diagrams to analyze our agent interaction. These diagrams make possible the following the flow of messages. Agents’ actions present in interaction model are: Registering Users, Warning Provider, Negotiation Process, Ask for Recommendations, Recommending Updating User Profile, Exchanging Information, Deciding to Trust and Coalition Formation Problem. Following, we
show as an example the “Warning Provider” action to explain in detail the communication protocol that we built for the MAS proposed.

**Warning Provider:**

This protocol is based in the warn-provider role of the BA. The BA knows about all possible services in the environment and the providers’ position. After the LA returns the user position to the UA, it gives the position to BA. The BA may filter all providers that are close to UA. Then it has to match the UA with the filtered providers.

### 2.5. Collaboration diagram

Collaboration Diagrams representing instances of agent instead of agent roles. We design a collaboration diagram where the agents interact to obtain the final diagram of MAS architecture. This is presented in Figure 1.

### 3. MAS Architecture

According to the methodology described in section 2 and the BDI model [6], the proposed architecture is composed by the following agents: Providers, Users, Brokers, Locator and User Management. Figure 2 shows an overview of this architecture.

---

**Figure 1. Collaboration Diagram of AUML**

First, each user has to register into the UMA. Then, the UA has to identify itself with the LA who returns the current position. The LA, depending on the system, could be a fusion data agent when the system works with different location technology such as wireless, wimax or ultrawideband. The LA is who reasons over spatial-temporal data. Then, the UA is going to request different services according to the user position. For these proposes, it has to communicate with the BA. This agent is the one who matches user’s preferences with provider’s services when both are in the same zone. The BA is also responsible for services managing and discovering providers.

Once PA was warned about user preferences with a specific profile, the negotiation process could be started. This process could be very 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 the BA) as general information.

An UA can communicate with other UAs directly to negotiate or to ask about recommendations. The UMA plays a main role in this operation. For example, if the design of the system includes reputation information, the UMA would assume the role of matching users. In other case, it could be directly the agent who proactively spreads recommendations. At the same time, the UMA could handle the agent coalitions over common interest.

---

**Figure 2. Context-aware multi-agent system architecture for heterogeneous domain**

UMA, BA and LA are going to run in one or different servers. Each provider agent runs over a particular server. Finally, UAs are executed on mobile devices.

As an example, the architecture could be adapted to a conference domain. Thomas assists to
a conference. First, the LA establishes user position in the parking lot. When Thomas moves into the building, the LA detects if he is nearby the registration sector. At that moment, the UMA asks the Thomas’ UA to register. Then, it situates the user position respect to the conferences rooms and poster sessions. The UA send his position to the BA. Next, the BA matches the listener’s user profile with the information that will dictate the speaker. The UA receives messages of the PA from the session room.

Then, the LA detects that Thomas moves to coffee zone. The UA receives his position again. The UMA knows about the situation and warns the other UAs about Thomas presence. Some other UAs make recommendations to other users about Thomas presence. After a few minutes, the UA moves to the common area and receives a message in the mobile device from the poster PA. Then, information about the research line and other papers are showed in the mobile device. That is possible because the UA has sent his position to the BA again, and it implemented a new matching. An UA send a request to the poster PA, and it starts a negotiation process about a possible job. In the same manner, the task of each agent could be adapted to any context. The use of a generic ontology facilitates the adaptation.

4. Evaluation

4.1. Experimentation

The application building is the simulation of FIPA Protocols. There are three types of agents: client, provider and broker. The client agents receive products and services offer by providers. Agents’ behaviors allow the agent to listen messages, to receive information and to decide the course of an action. Agents could negotiate based on ambient intelligence ontology describe in [7]. In this evaluation, the architecture exposed in [6] (MAS1) is compared with the architecture proposed in present work (MAS2). In MAS1, the architecture has three types of agents: Central Agent, Provider Agents (one per each interest point) and User Agents (one per user). The communication started when the central agent warns a provider agent about the presence of a potentially interested user. Previously, the Central Agent localizes and matches user profile with the profile of each provider. Then, the negotiation process between Clients and Providers starts. In MAS2, while the LA localizes and identifies UA, the BA matches the User Profile with the services and products offers by the PA. The evaluated behaviors in both systems are: Product’s Offer, Contract Net and Dutch Auction protocols.

Although the research of MAS architecture is continuously rising, only few evaluation methods had been investigated [11, 5]. The present section looks into several solutions in order to evaluate the architecture described earlier. It is worth to mention that none of these solutions cover all the aspects in the architecture evaluation. The purpose of this section is to obtain a certain characteristic of the different evaluation methods in order to achieve a completed evaluation of the architecture presented in this paper. The work of [11] described a method that evaluates the MAS from the communication agents’ point of view. This method suggested an evaluation based on the weight of the information brought by a message. Figure 3 presents the amount of sent messages between agents for each MAS during different situations such as Product’s Offer, Contract Net negotiation and Dutch Auction.

4.2. Evaluation Method

Although the research of MAS architecture is continuously rising, only few evaluation methods had been investigated [11, 5]. The present section looks into several solutions in order to evaluate the architecture described earlier. It is worth to mention that none of these solutions cover all the aspects in the architecture evaluation. The purpose of this section is to obtain a certain characteristic of the different evaluation methods in order to achieve a completed evaluation of the architecture presented in this paper. The work of [11] described a method that evaluates the MAS from the communication agents’ point of view. This method suggested an evaluation based on the weight of the information brought by a message. Figure 3 presents the amount of sent messages between agents for each MAS during different situations such as Product’s Offer, Contract Net negotiation and Dutch Auction.
messages presented represent the actual amount simulated. Next, the follow possibilities of messages type have been considered: Inform, Propose, Accept-Proposal, Call-for-Proposal (CFP), Request, Query-If and Reject-Proposal.

![Dutch Auction Runs](image1.png)

**Figure 4. Dutch Action Process for MAS1 and MAS2**

A value among “high” (1), “medium” (0.75), “low” (0.50) and “not pertinent” (0) is affected for each type of message depending on the importance of the message for the agent [11]. The results of this evaluation are showed in Figure 5. The assignment criteria of these weights to each type of message are the following: If the behavior tries to convince the user about any product or service, the maximum value is assigned to CFP or Propose message. Otherwise, if the interest over negotiation lays on the User Agent, the maximum value is assigned over the final result of the operation, i.e., accept or reject messages. The values used in Figure 5 are presented in Table 3.

Then, following the [5] evaluation method, it is inferred that the present proposal (MAS2) corresponds to an Asynchronous Distributed System in contrast to MAS1 that it is an Asynchronous Centralized architecture. Among the MAS2 properties the reactivity, communication, robustness and scalability highlight apart from the others. On the other hand, the MAS1 is more robust when evaluating fairness and load balancing. In this sense, the difference between the two systems, when performing the comparison on the number of messages sent is minimal. Therefore, MAS2 is valid.

![Sent messages weight for MAS1 and MAS2](image2.png)

**Figure 5. Sent messages weight for MAS1 and MAS2.**

5. **Conclusions and Future Works**

We have focused in agents’ architecture for AmI, showing the design of the system. This model was built using GAIA and AUML methodologies. The proposed architecture is flexible to any service that could be included in AmI environments. Here, the architecture proposed was evaluated according to [13] evaluation method. We take into account the communication between agents. The evaluation method consist in analyze the sent messages number and the importance of each message for each agent. Then, the approach has been implemented on two different architectures that implement the same agents’ behaviors. The results show the robustness and scalability of the architecture. It is proposed for future works to analyze the MAS architecture from the performance point of view described in [1].
<table>
<thead>
<tr>
<th>MAS 1</th>
<th>Offer a product</th>
<th>Contract Net Protocol</th>
<th>Dutch Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Message</td>
<td>Number of messages</td>
<td>Weight</td>
<td>Results</td>
</tr>
<tr>
<td>Inform</td>
<td>2</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>Propose</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Accept</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>CFP</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Request</td>
<td>1</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>QueryIf</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Reject</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAS 2</th>
<th>Offer a product</th>
<th>Contract Net Protocol</th>
<th>Dutch Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Message</td>
<td>Number of messages</td>
<td>Weight</td>
<td>Results</td>
</tr>
<tr>
<td>Inform</td>
<td>1</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>Propose</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Accept</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>CFP</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Request</td>
<td>1</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>QueryIf</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Reject</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Proposal</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3. Weight affected for each type of message depending on the importance of the message for the agent.

References


