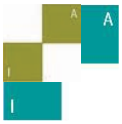
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Sánchez-Pi, Nayat; Carbó, Javier; Molina, José Manuel
Analysis and Design of a Multi-Agent System Using Gaia Methodology in an Airport Case of Use
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 14, núm. 45, 2010, pp. 9-17
Asociación Española para la Inteligencia Artificial
Valencia, España

Inteligencia Artificial 45(2010), 9-17
doi:10.4114/ia.v14i45.1070



INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

Analysis and Design of a Multi-Agent System Using Gaia Methodology in an Airport Case of Use

Nayat Sánchez-Pi, Javier Carbó and José Manuel Molina
Computer Science Department
Carlos III University of Madrid
Avda de la Universidad Carlos III, 22. 28270. Colmenarejo. Madrid. Spain
naya.sanchez@uc3m.es

Abstract Progress in software engineering has been made through the development of natural high-level abstractions with which to model and develop complex systems. The abstraction and modelling of agent technology play an important role in distributed systems. Multi-Agent System paradigm introduces a number of new abstractions when compared to more traditional systems. They may be used by software developers to more naturally understand, model and develop an important class of complex distributed systems. Accordingly new analysis and design methodologies are needed to effectively engineer such systems. In this paper, we propose the analysis and design of a multi-agent system for the provisioning of context-aware services in an ambient intelligence domain: an airport.

Keywords: Multi-Agent systems, methodologies, context-aware.

1 Introduction

Ambient Intelligence is the paradigm to equip environments with advanced technology to create an ergonomic space for the user where they could interact with their digital environments the same way they interact with each other. It is also associated to a society based on unobtrusive, often invisible interactions amongst people and computer-based services taking place in a global computing environment. Services in AmI will be ubiquitous in that there will be no specific bearer or provider but, instead, they will be associated with a variety of objects and devices in the environment, which will not bear any resemblance to computers. People will interact with these services through intelligent and intuitive interfaces embedded in these objects and devices, which in turn will be sensitive to what people need. Another good point seen on the AmI vision is that the electronic or digital part of the ambience (devices) will often need to act intelligently on behalf of people. The components of ambience will need to be both reactive and proactive, behaving as if they were agents that act on behalf of people. If we assume that agents are abstractions for the interaction within an ambient intelligent environment, one aspect that we need to ensure is that their behavior is regulated and coordinated. For this purpose, we need rules that take into consideration the context (location, user profile, type of device, etc…) in which these interactions take place. Taking care this, the system needs an organization similar to the one envisaged by artificial agent societies. The society is there not only to regulate behavior but also to distribute responsibility amongst the member agents. In [1], O'Hare et al. advocate the use of agents as a key enabler in the delivery of ambient intelligence. As is the case, with any new software engineering paradigm, the widespread development of complex software systems based on MASs (Multi-agent systems) requires not only new models but also the identification of the software abstractions. Multi-Agent System paradigm introduces a number of new abstractions when compared to more traditional systems. They may be used by software developers to more naturally understand, model and develop an important class of complex distributed systems. Accordingly new analysis and design methodologies are needed to effectively engineer such systems. Several methodologies for the analysis and design of MASs have been proposed so far but few of them explicitly focus on organizational abstractions. Some agent-oriented methodologies are MASCommonKADS [2], Tropos [3], Zeus [4], MaSE [5] or INGENIAS [6]. MAS-CommonKADS is a methodology for knowledge-based system that defines different models (agent model, task

ISSN: 1137-3601(print); 1988-3064(on-line)

model, organizational model etc.) in the system lifecycle using oriented-object techniques and protocol engineering techniques. Tropos is a requirement-based methodology; Zeus provides an agent platform which facilitates the rapid development of collaborative agent applications; MaSE is an object-oriented methodology that considers agents as objects with capabilities of coordination and conversation with automatic generation of code and UML notation; and INGENIAS proposes a language for multi-agent system specification and its integration in the lifecycle, as well as it provides a collection of tools for modelling, verifying and generate agents' code.

For the process of analyzing and designing agent-oriented systems, we have selected Gaia methodology [7]. One of the main motivations to use Gaia for developing multi-agent systems is the necessity of viewing the system as a human organization, represented by a social community of agents, since the organization is not only a collection of roles, and Gaia provides this aspect in an intuitive way. Gaia helps in this task by defining organizational abstractions (environment, roles, interactions between roles, organizational structure and organizational rules). One problem with Gaia could be that it is a very high-level methodology. Gaia specifies how a society of agents collaborates to reach the goals of the system, and what is needed of each one to achieve them. This high level view is precisely what it is needed in this research. MaSE, MASCommonKADS and INGENIAS could be used to analyze our multi-agent system problem, but their own characteristics make the process more complex, since our main intention is to focus in the organizational abstractions mention before, without any object-oriented technique or specification language. In this paper, we propose the analysis and design of a multi-agent system using Gaia methodology and its validation using a real-world application, the provisioning of context-aware services in an ambient intelligence domain: an airport.

2 Problem formulation

In this section we will present the definition of our context-aware problem in an airport domain. What we precisely propose is a distributed approach based on agents to address the problem of using context-aware information to offer customized services to users in an airport domain, and especially for the Barajas-Madrid Airport and for services offered to clients, passengers, crew and staff of IBERIA airline. We are familiar to this problem since we previously developed a centralized system with the same final intention using Appear Networks Platform and Aruba Wi-Fi Location System [8]. The current proposal includes agents implemented in JADE and LEAP that will make use of the user profiles to customize and recommend different services to other agents avoiding an obtrusive participation of a central server. These agents use a particular instantiation of a previously-defined generic ontology [9] to represent the airport context including the buildings, rooms, zones, etc…, where users are moving inside with their mobile devices, and several points of interest which provide services to users.

We have divided our Madrid-Barajas airport domain into six different zones that are not overlapped each other. Airport Zone (containing the rest of the zones), Customs Zone (customs), Commercial Zone (stores, cafeterias, restaurants), Offices Zone (airline offices), Check-In Desk Zone (check-in desks) and Finger Zone (finger). This is part of a previous work [8, 9, 10] we have referenced before where we have used a centralized platform to define the behavior of our system. We firstly identified agent types as: Central, Provider and Client agents. Central agent represents the infrastructure that acquires location information from different sensors. Provider agents act on behalf of the different services and Client agents represents the users with a wireless device, see Fig.1.

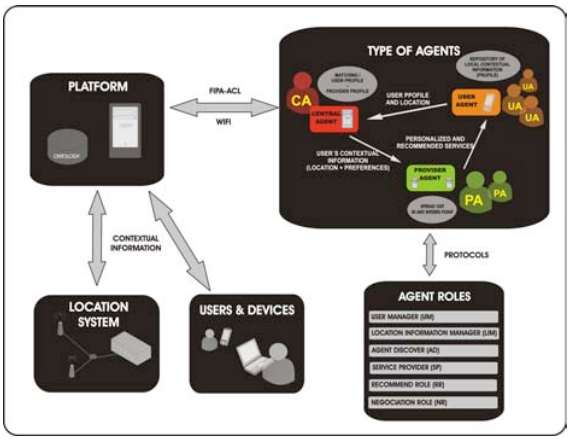


Figure 1. MAS system architecture.

So finally the proposed three types of agents implement the following functionalities:

- Central agent main tasks relies on detecting users, registering and deregistering users, improving shared user profiles, filtering and warning closer providers. Central agent manages the main system activities and it is supported on provider agents in order to balance the activity load.
- Provider agents functionality are closely related with dialogue between client agents and provider agents, since they can reach an agreement with clients and communicate the most suitable services to them, according clients preferences and profile.
- Client agents main goals includes negotiation with providers agents, recommend services to other client agents, trust in other agents, and manage and improve their internal profile to receive better services according to it.

Furthermore, an ontology has been implemented for the agent communication instantiating a high level of conceptualization for the domain. Fig 2 shows part of the ontology used.

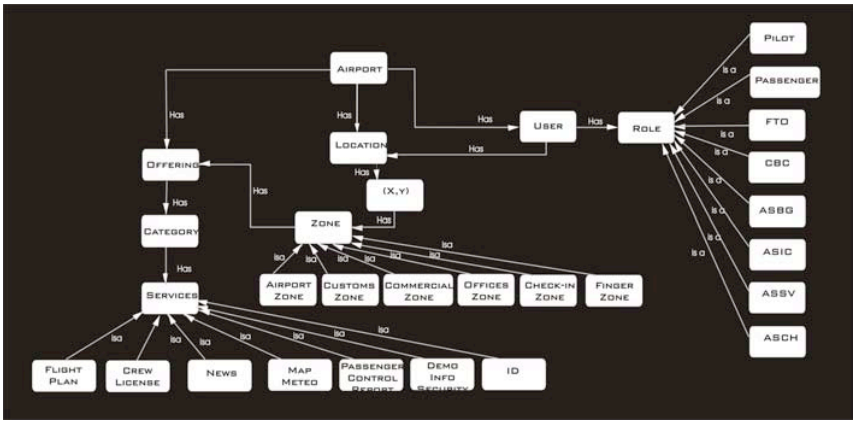


Figure 2. High level concepts of the generic ontology.

The ontology for the airport domain includes the following principal concepts:

- **Airport:** It is the high level concept representing the domain on discourse.
- **Location:** Includes the airport location, their zones and the user location.
  - **(x; y):** There are the coordinates of the airport location, their zones and the user position.
  - **Zone:** It is a segment of the airport area distinguished from adjacent zones by the different characteristics and the coordinates.
- **User:** Every person who has a role in the domain.
  - **Role**: Role of each user in the domain
    - **Pilot:** Captain of the Aircraft and Pilots.
    - **Passenger:** Passenger.
    - **FTO:** Flight Technical Officer
    - **CBC:** Cabin Crew.
    - **ASBG:** Airline Staff in ground. Boarding.
    - **ASIC**: Airline Staff in ground. Incidence.
    - **ASSV:** Supervisor.
    - **ASCH:** Airline Staff in ground. Check-In
- **Offering:** A class containing a set of services grouped by categories.
  - **Category:** A class containing a set of grouped services
  - **Services:** Applications or notifications
    - **Flight_Plan**: It is a web service which delivers the flight plan of the current flight.
    - **Crew_License:** It is also a web service containing the license to flight of the crew who is assigned to the current flight.
    - **Demo_Inf_Security:** It is an informative document about security available just to pilots and passenger.
    - **Map_Meteo**: It is an informative document available to pilots, FTO and CBC.
    - **News**: Access to the url of the principal newspapers in Spanish, English and French.
    - **Passenger_Control_Report:** It is a web service which delivers the flight plan of the current flight.

- **ID:** It is a bar code used to improve the ID data capture.

3 Analysis and Design using GIAA Methodology

Gaia is a methodology for systems design based on agents that allows going from a few initial requirements to a design in a systematical way, so that the detail level is sufficient to be implemented directly. Gaia deals with both the societal level and the agent level aspects of design. Extended Gaia version [7] for developing open multi-agent systems, includes two organizational abstractions (organizational rules and organizational structure, environment, roles and interactions between these roles). This is necessary for analyzing open multi-agent systems as our context-aware system, in contrast with original Gaia definition which is oriented to closed agents communities.

Gaia methodology includes two main phases: analysis phase, design phase. [11] The Gaia process starts with the analysis phase, whose main goal is the collection and organization of the system specification which is the basis for the design of the computational organization. Extended Gaia version considered as well the organization structure and rules. This phase includes: the environment model, the preliminary role model, the preliminary interaction model and organizational rules. The output of the analysis phase is exploited by design phase. The aim of designing is to transform the analysis models into a sufficiently how level of abstraction that design techniques can be applied in order to implement agents. The outcome of the Gaia process is a detailed but technology-neutral specification that should be easily implemented using an appropriate agent-programming framework as Jadex platform in this case. The architectural phase includes definition of the system's organizational structure in terms of its topology and the completion of the preliminary role, and interaction models. And the detailed phase includes two models: (i) agent model and (ii) service model. They identify, respectively, the agent types in the system and the main services that are required to realize the agent's role.

Gaia allows clarifying the organizational structure of the multi-agent system and gives the necessary level of detail to develop a better implementation in order to represent the agent-based organization.

3.1 The Environmental Model

Since the importance of environments, because a multi-agent system is always situated in some domain, modeling this environment involves determining which entities and resources take part of the multi-agent system, for reaching the organizational goal. For the proposed context-aware multi-agent system with the main goal of adapting context information for providing customized services to users based on their location and profile, the environments is represented for multiple, heterogeneous and dynamical domains.

One of our previous researches explains how the environment for this context-aware system is represented by an ontology; since there is a need of define the context knowledge for the communication process between agents in the system. Our proposed ontology identifies the next high level concepts: framework is the general application concept which includes high level system concepts and it defines the different system domains with sector and event properties. Location concept is the (x, y) coordinates of any place, participant or object. Spatial region and temporal region, defines the area or map composed by segments according to a range of coordinates, and temporal system information about users in any localization in spatial region, respectively. Place concept represents interest points: stands, departments etc. and they are defined in one segment or a group of segments or in specific coordinates. Participant concept represents people and companies with their correspondent role in system. Service concept stands for different kind of system provision offered to users referred to products in places or interest points. A service can be a notification in user device about preferential user product. Product concept stands for any kind of information or object that users want to be informed. And finally, device concept gathers information about different user's devices profiles and hardware.

3.2 Role Model

A role model identifies the main roles that agents can play in the system. A role is viewed as a description of the functionality of an agent, and it includes four attributes: permissions (resources use while performing a role), responsibilities (expected behaviours of the role), activities (actions without interaction between roles) and protocols (actions than involve interaction between roles). The following roles have been defined in the proposal multi-agent system:

- User Manager: this role is responsible of coordinating all activities about users in order to locate, identify and register/deregister users.
- Information Manager: this role is responsible of managing public part of users' profile and improving them with environmental information. Moreover, this role manages information about services and products offered by providers.

- Matching Manager: this role is responsible of checking the two matching possibilities: between services offered by providers and users profiles, or between two users' profiles, to identify the similitarities. Furthermore, providers' location and users' locations are part of this matching for checking the proximity between a provider and a user, and between one user and another user.
- Agent Discover: it obtains the closest provider to a user, or the closest user to another user, and alerts them for the presence of the proximity of a user in both cases.
- Service Provider: the main function is to communicate the offered services to users according to user location and profile. Another function is to compromise with clients for making agreements.
- Internal Profile Manager: it deals with the update of internal user profile. This profile can be updated with information given by other users.
- Negotiation Role: users can negotiate with providers and vice versa, for reaching agreements about services and products.
- Recommend role: users can communicate with other users to recommend places or points of interest, and, generally, all kind of information related to system services.

Some of the roles described in the role model can be seen in Table 1, according to Gaia specifications. It shows the roles corresponding to central agent, provider agent and client agent, with the permissions, responsibilities, protocols and activities (underlying) associated to them:

Table 1: Gaia Role Model for the context-aware multi-agent system problem.

Role Schema: User Manager (UM)
Description This role is responsible for locating, identifying and registering users, as well as improving their public profiles.
Protocols and Activities Check-user-location, agree-registry, deregister-user, receive-registry-profile, register-user, identify-user, receive-user-sequence, improve-user-profile.
Permissions Reads user_location, user_profile, sequence Changes user_registry, user_profile.
Responsibilities Liveness: UM= (Check-user-location. Identify-user. Receive-registry-profile. Agree-registry. (Register-user | Deregister-user))n | (Receive-user-sequence. Improve-user-profile)n Safety: it is necessary to assure the connection with the location system and knowledge base

Role Schema: Service Manager (SM)
Description This role makes the matching about 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 is available to make the matching.

Role Schema: Provider Discovery (PD)
Description This role obtains closer providers and communicates with them to alert for 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.

There are also other models to take into account: interaction model, organizational rules, organization structure, agent model and service model.

3.3 Interaction Model

Interaction model is used to represent the dependencies and relationships between the roles in the multi-agent system, according to the protocol definitions. For the previous roles, several protocols are defined. There are some of them represented in the following table (Table 2):

Table 2: Gaia Interaction Model for the context-aware multi-agent system problem.

Receive-Registry-Profile		
Profile Manager	User Manager	Request registry
Receive a request of registry to the user with the shared part of the user profile.		Received Request and Profile

Agree-Registry		
User Managerr	Profile Manager	Confirmation message
Send a message to confirm the user registry.		User registered or not

Warn-provider		
Provider Discover	Service Provider	Alert message
Send a inform message to the closer provider role for alerting the presence of a user.		Received message

Offer-Service		
Service Provider	Negotiate Role	Offer Sent
Send offers about services to users, that can be negotiated.		Offer Received

The following figure shows a little example of how the interaction between the tree kinds of agents would be: Provider with Central agent, Central agent with Client (User) agent and Provider with Client agent.

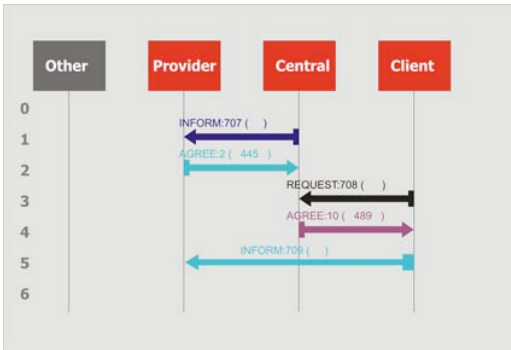


Figure 3. Message exchanges captured in the experiment with JADE Agents.

3.4 Organizational Rules and Agent Model

According to Gaia, the preliminary phases capture the basic system functionalities, characteristics and interactions in an independent way of the organizational structure. However, there are some relationships between roles and protocols which can be complemented with organizations rules for a best capture. These organization rules are responsibilities of the organization. [4] In these cases, organization rules correspond to the responsibilities analyzed in role model. For showing this more clearly, here there are some examples of organizational rules defined about roles and protocols. For instance, there is a rule like the following:

(Filter- Provider)□ → Warn-Provider

This means that Filter Provider must precede Warn-Provider and it can be executed n times. Each time Filter-Provider rule is executed, the action Warn-Provider will be executed after it. Otherwise, if we add the role, for instance Provider Discover role (PD) played by central agent, the organizational rule should be:

(Filter-Provider (PD))□ → Warn-Provider (PD)

According to Gaia, agents are entities that play different roles in the system. The definition of the agent model consist on identifying which specific roles play agents and how many instances of each agent have to be instantiated in the actual system. In this paper we have considered not to extend more this section because it corresponds to the role model and focus in other important phases.

3.5 Service Model

The service model in Gaia methodology represents all protocols, activities, responsibilities and liveness, associate to the roles that agents plays in the system. This model is detailed as follows in the Table 3:

Table 3: Gaia Service Model for the context-aware multi-agent system problem.

Services Schema: Profile Manager (PM)

Service	Inputs	Outputs	Pre-Condition	Post-Condition
Update-internal-profile	Other users reco-mmen-dations	Profile updated	Received other users recommen-dations	The profile is updated
Send-shared-profile-registry	Need of registry	Send request message and pro-file to central agent	Obtain closer provider	Provider informs closer users

Services Schema: Services Manager (SM)

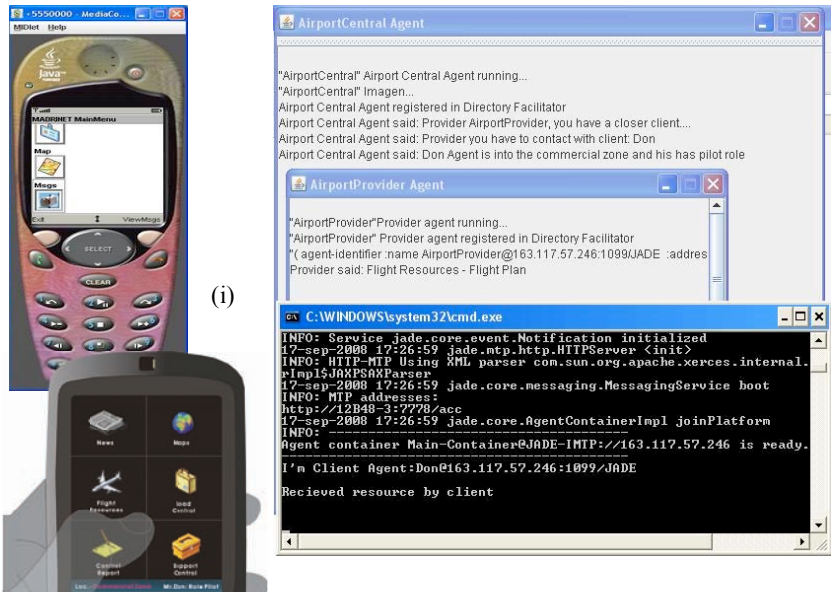
Service	Inputs	Outputs	Pre-Condition	Post-Condition
Match-services-profiles	User profile known by central and provider informa-tion and location	Location checked	Users connected to wireless network	Users located

4 Validation using an airport case of use

We have implemented a multi-agent system following the architecture of the multi-agent system explained in section 1, and using the particularization of the generic context-aware ontology described above.

Although other alternatives exist, JADE-LEAP agent platform was chosen since it is a java-based and FIPA compliant agent platform, where agents communicate by sending FIPA ACL messages over a TCP/IP connection between different runtime environments on local servers or running on wireless devices. One local server will host the central agent that would rule over all the sensors of the airport (these sensors are just simulated inside the implementation of the central agent) that provide location information of clients, while one or more local servers host provider agents acting on behalf of airport services. Finally each portable client device runs a JADE-LEAP container that hosts one single agent that is used to provide a way to interact with the user (through a GUI), so the user can be informed and interact with the other agents running on different servers. JADE/LEAP platform logically connects the agents on the different servers with each other and with the corresponding client agents running on the users' devices. First we assume an initial minimal profile known by the client LEAP agent: name, agent role, passport number, nationality and travel info (flight numbers, companies, origin and target). This is what we consider the public profile that Client agent includes in the content of the first message exchanged with the central agent.

Other additional data can be included in the profile as: smoker/nonsmoker, languages spoken, number of times that the passenger visited cities of current travel, professional activity, travel motivation, relationship with other passengers, etc. Some of these would be considered private information therefore this private information would be just internally used inside Client agent to provide customized filter of the services offered by provider agents. The user decides whether he introduces these confidential data or not. Finally our intention is to include reputation information linked to agents, and specific services inside client profiles, but this is not already done. Once client agent in LEAP, provider and central agents in JADE were implemented, we have chosen a specific case study to test such implementation in the airport application domain. We will describe how the system works by following all the messages exchanges when a passenger "Don" arrives to the airport .When Don enters into the system's wireless network, Don client agent (running the LEAP code on the mobile device) asks for registration to the central agent (running JADE code in a PC acting as server). Registration includes identifying itself, so future location of passenger Don can be assigned to a particular AID (agent identifier) in advance. Once Receive-registry-profile protocol has concluded, the central agent evaluates its position (through simulated movements) and computes the geographical proximity to the location of different services (platforms/PCs that host a particular JADE provider agent). Then, central agent notifies about the presence of such Don client agent with 'Notify agent' protocol to the corresponding geographically close provider agents. When provider agent has been notified, it will offer its particular services customized according to the context exchanged by the central server (the public profile of the client agent). After that, negotiation included in offer-service protocol takes place in order to reach a possible agreement about services. For instance check-in provider agent would have received information about the flight of Don, and according to this info the provider agent might offer Don agent an automated check-in (asking Don agent about the absence of baggage). Or alternatively check-in provider agent might inform the client about the place to carry out the manual check-in, jointly with other context-specific info such as the expected delay time to check-in, or the number of people in the row, etc.



(ii)

(iii)

Figure 4. Validation. (i) Java Mobile simulator; (ii) Console agents; (iii) HTC Touch HTC Touch terminal with a TI's OMAP™ 850, 200 MHz processor and Windows Mobile 6.

In any case, the given client agent has to internally filter out the interesting services from the received services based on the private profile of the user. This filter would avoid an over exposition of information to the user and would protect the privacy of the user. We have presented an application of agent technology to an ambient intelligence problem in an airport domain. We adapted a previous centralized system to this domain into a distributed, agent-based system. We made an analysis and design of the multiagent system using Gaia methodology, we implemented three types of agents: central, providers and clients with JADE and LEAP and finally we made the validation using a console interface, a simulator interface running Java and an HTC Touch HTC Touch terminal with a TI's OMAP™ 850, 200 MHz processor and Windows Mobile 6.

5 Conclusions

We have presented the analysis and design of a multi-agent system using Gaia methodology in application to an ambient intelligence problem for an airport domain. The key concepts in Gaia, roles which are associated with them responsibilities, permissions, activities, and protocols are previously seen and validated with an airport case of use. We adapted a previous centralized system to this domain into a distributed, agent-based system. We have implemented three types of agents: central, providers and clients with JADE and LEAP with the intention to avoid bottle necks and to increase the possibilities of customization through the use of enriched user profiles..

Acknowledgements

This work was supported in part by Projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB and CAM MADRINET S-0505/TIC/0255

References

[1] O'Hare, G. M. P., O'Grady, M. J. Kegan, S., O'Kane, D., Tynan, R. and Marsh, D., "Inteligent Agile Agents: Active Enablers for Ambient Intelligence." ACM's Special Interest Group on Computer-Human Interaction (SIGCHI), Ambient Intelligence for Scientific Discovery (AISD) Workshop, Vienna, April 25, 2004..

[2] Iglesias, C.A., Garijo, M., Gonzalez, J.C., Velasci, J.R.: A Methodological Proposal for Multiagent Systems Development Extending CommonKADS http://citeseernj.nec.com/ (1996).

[3] Giunchiglia F., Mylopoulos J., Perini A., "The Tropos Software Development Methodology: Processes, Models and Diagrams", 2002 *Autonomous Agents and Multi-Agent Systems* (AAMAS 2002), Bologna, Italy, July 2002. ACM

[4] H. S. Nwana, D. T. Ndumu, L. C. Lee and J. C. Collis, "ZEUS: A Toolkit and Approach for Building Distributed Multi-Agent Systems", in J. M. Bradshaw, ed*., Proceedings of the Third International Conference on Autonomous Agents* (Agents '99), ACM Press, Seattle, USA, 1999, pp. 360-361.

[5] M. F. Wood and S. A. DeLoach, "An Overview of the Multiagent Systems Engineering Methodology". *Agent-Oriented Software Engineering,* Volume 1957 of LNCS, Berlin: Springer, January 2001, 207-221.

[6] Pavón, J. and Gómez-Sanz, J., INGENIAS web site: ttp://grasia.fdi.ucm.es/ingenias/, consulted at December 2006.

[7] F. Zambonelli, N. R. Jennings, M. Wooldridge, "Developing Multiagent Systems: The Gaia Methodology," ACM *Transactions on Software Engineering and Methodology*, Vol. 12, No. 3, Jul. 2003, pp. 317-330.

[8] Sánchez-Pi, N., Fuentes, V., Carbo, J., Molina, J.M., Knowledge-based System to define Context in Commercial Applications. *In 8th Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, Qingdao, 2007.

[9] Fuentes, V., Carbo, J., Molina, J.M., Heterogeneous Domain Ontology for Location Based Information System in a Multi-agent Framework. *In 7th Int. Conf. on Intelligent Data Engineering and Automated Learning*, Burgos, Spain, 2006.

[10] Fuentes, V., Sánchez-Pi, N., Carbó, J., Molina, J.M., Reputation in User Profiling for a Context-aware Multiagent System. *In 4th European Workshop on Multi-Agent Systems, Lisbon, Portugal (2006)*.

[11] M. Wooldridge, N. J. Jennings and D. Kinny. "The Gaia Methodology for Agent-Oriented Analysis and Design". *Journal of Autonomous Agents and Multi-Agent Systems* 3(3):(2000) 285-312.