Developing a Multiagent System for Mobile Devices

Pedro Cuesta, Alma Gómez, Juan C. González, Francisco J. Rodríguez

Dpto. de Informática (Universidade de Vigo)
Ed. Politécnico, C.U. As Lagoas s/n
Ourense 32004
{pcuesta,alma,jcmoreno,franjrm}@uvigo.es

Abstract

The development of applications distributed partly on handheld devices is becoming more and more important. This paper shows the use of mobile agents technology in the development of a distributed system, called SAIPE. It makes easier the reading of user e-mail from different accounts and the searching of documents in Internet with a PDA. SAIPE has been developed following the INGENIAS Agent Oriented Methodology and implemented using JADE-LEAP software framework.

Keywords: Multiagent systems, mobile agents, Agent Oriented Methodologies, INGENIAS, JADE-LEAP.

1. Introduction

Advances in information and communication technologies and the current expansion of wireless connectivity has led an increasingly utilization of mobile devices such as PDAs, Pocket PCs, portable computers, cell phones, etc. These devices provide the capacity of accessing enterprise intranets or browse the Internet using a wireless network like GSM (by means of a GPRS communication).

In this kind of systems, the user generally receives more information than desired, as results of a request, increasing in this way the amount of traffic generated by the operation. This is due mainly to the fact that browsing and request processes in Internet include rescindable information (banners, links to yet non-existing documents, etc.). The associated rise in communication costs (habitually this kind of technologies has a cost proportional to the originated traffic) cannot be controlled by user.

In addition to the problem mentioned above, another usual disadvantage in this kind of networks is that connection can be interrupted at any moment, for instance through lack of coverage. In fact, this causes that operations, for instance the download of an attached document or the request of information in a web page, may be lost. Even more, these mobile devices have a limited calculus capacity and, are, therefore, unable to process the great amount of data they receive.

In the latest years, Agent Oriented Development has become a new Software Engineering Paradigm [3, 4,15]. Agent concept constitutes a powerful abstraction tool in software development, which facilitates the construction of distributed, intelligent and robust systems [12].

Agents are defined by means of the attributes that they must accomplish: autonomy, reactivity, social ability and proactivity, besides other aspects such as: rationality, mobility, etc. [17]. General speaking, the solution to complex problems cannot be provided by a single agent, but by a set of agents which interact in order to achieve system objectives. The use of several agents represents naturally system decentralization and stresses communication in managing the dependencies [16, 18].

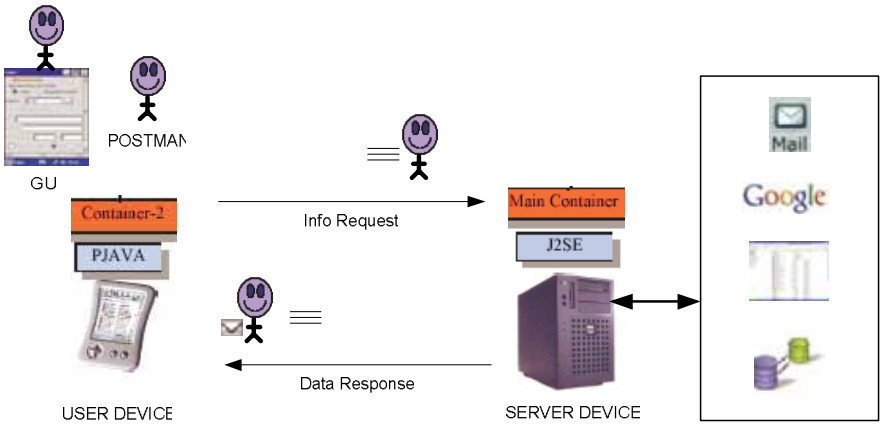

Figure 1. SAIPE operation

Nowadays, one of the most relevant fields of agent orientation is related to mobile agents [13]. That is, autonomous software that can stop execution, move to other device and restart running in the same point. This ability, jointly with the characteristic attributes of agents makes mobile agents suitable for working in a wireless network. The use of mobile agents may solve many of the problems previously exposed, making easier the creation of software for network monitorization or for searching and filtering information. The amount of information, which flows in the net, can be reduced using mobile agents, because the data is processed in the same device where it is stored.

This paper shows a practical use of mobile agents in the development of a personal information system, which allows a user to search the net and access his electronic mail in limited connectivity environments. The system has been developed using INGENIAS methodology [7], and implemented in JADE-LEAP platform for mobile agents [1, 11].

In the reminder of the paper, the different steps followed in the system construction will be documented. After this introduction the second point describes functionally the system, while the third one introduces the agent-oriented methodology used in the development. Next section details the activities carried out during analysis and design. In section five, an overview of the implementation platform is done. Section six explains some aspects of implementation taken into account when programming the system. Finally, the conclusions and future work are presented.

2. System Description

SAIPE is the acronym in Spanish of Access to Personal Information System from limited connectivity environments. It is a system, which facilitates the access to personal information when using mobile devices, and a non-reliable network connection, such as wireless networks based on GPRS [2].

Essentially, SAIPE accesses electronic mail accounts getting only some mails (the currently interesting ones for the user), leaving the rest in the same place. Besides, it allows the users to look for documents, using any of the search mechanisms registered in the system and, afterwards, download only the useful ones. This way of working moves the heaviest processes from the user device, which is executing the application, to another one with a higher calculus capacity (called server device). The current version of SAIPE allows searching POP3 mail accounts, finding web documents using Google, and locating personal files saved in server devices.

Agents on behalf of the user make these operations. SAIPE includes three kinds of agents: *Interface agents*, *DocumentSearcher agents* and *Postman agents*.

The system operates in a simple way. When a search of electronic mail is ordered (via *Interface agents*), this task is assigned to *Postman agent*. This agent, after receiving the order, moves to the server device where all the operations needed for answering the request will be done. Once the information is

achieved, the agent moves back to origin. The results are presented to user by means of the system *Interface agent*. An example of system operation is shown in Figure 1.

The system has been developed using a multiagent architecture that incorporates both mobile and static agents. The *Interface agent* is static. It is responsible of managing communications with user (human) using a graphical interface. The other two agents are mobile. They move to a device different from the one of the user (server device) in order to run there the operations requested by *Interface agent*. Once the agents are in the server device, they execute the operations using the wide band connection offered by the server and, also, the remaining services provided. After having the answers to the operations, they go back to user device and send the *Interface agent* these answers.

The *Postman mobile agent* carries out all the actions for managing electronic mail (read, delete, send, etc.). On the other hand, the *DocumentSearcher mobile agent* does all the operations over documents, such us searching, downloading, etc.

3. INGENIAS Methodology

In the system development, a Multiagent Systems methodology has been followed. The system has been developed using the INGENIAS methodology [7], which has been chosen among different agent-based methodologies [8], with the aim of assessing its suitability when accomplishing the development of a real problem and, also, for training the authors in the methodology use. The development process model suggested by INGENIAS is a simple variant of the RUP process [10], divided in four incremental phases that are performed in several iterations. The INGENIAS process is still architecture centered, but not necessary use cases driven.

The main objective during the INGENIAS inception phase is to ensure system viability. This phase is centered in analysis and design activities. The analysis activities are performed by the generation of the use cases using interaction models. The preliminary system architecture must be specified using a single organization model. Several environment models must be created to translate the obtained requirements to these models. The design activity proposed by INGENIAS is to develop, if needed, a system prototype using agent tools like ZEUS [19] or AgentTool [5].

In what respects to the elaboration phase, INGENIAS proposes as analysis activities to refine the use cases; generate agent models to detail the architecture elements needed; identify workflows and tasks in the organization models, and translate them into task and goal models in order to generate control restrictions in the main goals and to refine the goals identified. Finally, the environment models must be redrawn to include the new elements identified. The suggested design activities

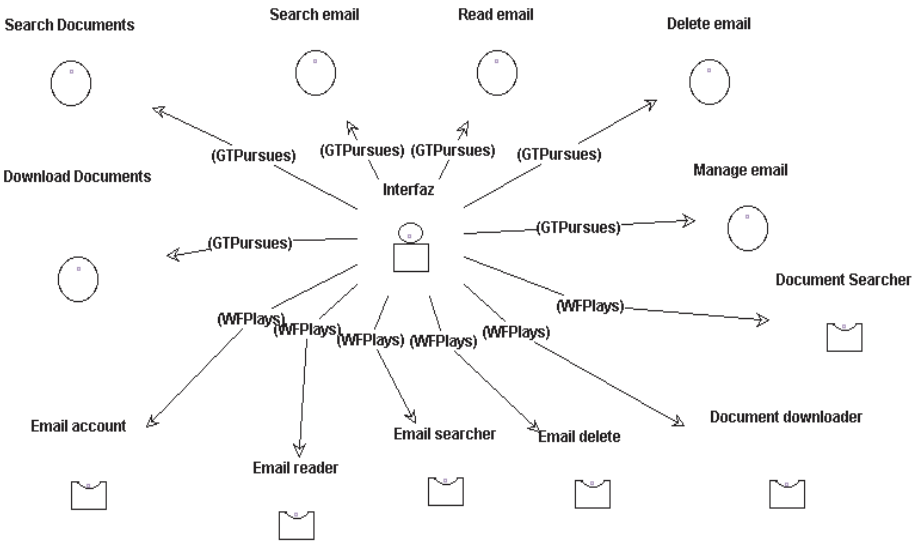

Figure 2. Organization Model

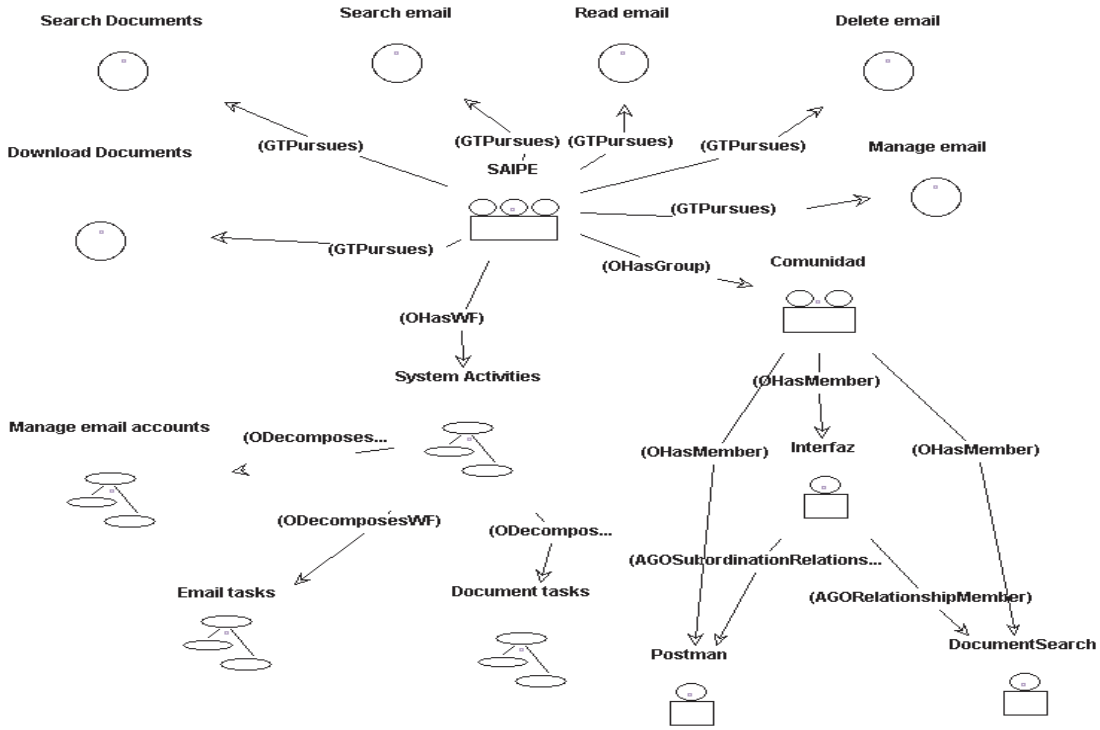

Figure 3. Interface Agent Model

in this phase are oriented to complete the organization model, develop workflows that specify the restrictions identified in the task and goal models to remark their relation with the system goals; specify the task execution in the interaction models; and generate agent models to detail mental state patterns.

During the construction phase, analysis activities are reduced to study and refinement of use cases still not implemented. The design activities proposed are firstly generate and refine agent models; and secondly depurate the organization models doing a fully specification of the social relationships found.

The last phase, Translation, does not suffer modification from those proposed in the RUP process.

The next section presents briefly how the main activities of analysis and design performed during development have been accomplished in SAIPE system.

4. SAIPE Development

4.1. Requirements activities

A Requirements Specification document, which incorporates the necessary information for a global system vision, is fulfilled. From this document, the initial system functionalities are defined, and therefore, the system Use Cases can be identified and described in a Use Case Diagram. Each use cases has a priority value in order to decide in which one of the development iterations will be included. Finally, each use case is detailed in depth.

4.2. Analysis activities

Taking into account the Use Cases defined previously, the next step is to identify the ones in which an interaction among agents takes place; these ones must be associated with the correspondent Interaction Models. After developing these Interaction Models, an initial Organization Model is created (see Figure 2). This model outlines the system architecture and identifies workflows. Next the Environmental Model is generated and Agent Models are defined. Finally, the Tasks and

Goals Models are created, in order to define the control restrictions (that is, the main goals and the relationships between objectives and task). In Figure 3, the Agent Model for Interface agent is presented.

4.3. Design activities

In this phase the Organization Model is completed, detailing the workflows identified during analysis step. Figure 4 shows the Email Search Workflow. Next, the Tasks and Goals Models and Interaction Models are finished attaching the GRASIA

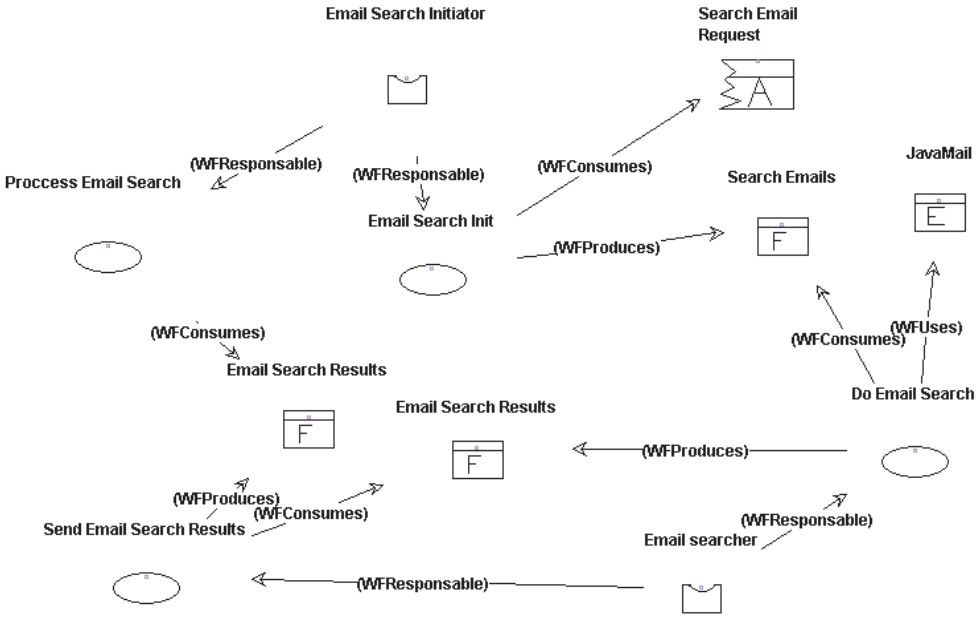

Figure 4. Email Search Workflow Diagram

specifications of the interactions (other notations, such as AUML [14], can be used to specify interactions). Finally, the design of system classes is done.

Besides, system ontology is defined. In this case, the ontology was created using the tool Protégé 2000 while the Bean generator module has been used for generating the associated Java classes.

4.4. Development tool

During this development a tool, which provides support to methodology, has been utilized. The documentation of analysis and design steps has been done using INGENIAS Development Kit (IDK) [9].

The software intends to facilitate the development of MAS by supporting the INGENIAS Development Process (IDP). It is provided by INGENIAS methodology creators and allows the definition of models proposed by the methodology. Even more, it allows rapid application development, decoupling of specification and implementation and verification of the specification according to the needs of the implementation.

5. JADE-LEAP platform

Java Agent DEvelopment Framework (JADE) [11] is an agent-oriented tool, implemented in JAVA and FIPA compliant [6]. It is composed of an agent platform (execution environment) and a set of packages that provide the basic support for multiagent systems construction.

JADE can be distributed among several computers and is customizable by means of a remote graphical interface. In addition, it provides the mandatory components (FIPA) for agents management:

- AMS (Agent Management System), which is responsible of providing the white page, agent lifecycle and agents directory services.
- DF (Directory Facilitator), that provides yellow page service.

- ACC (Agent Communication Channel), which controls message interchange.

The development environment incorporates a set of graphical tools, which facilitate the platform management, providing support to agents debugging and execution. These tools are: RMA (Remote Monitoring Agent), Dummy Agent, Sniffer Agent, Introspector Agent, GUI-DF.

In order to create a JADE agent, it is necessary to instantiate a class that extends the Agent class [1]. Each agent is implemented as a thread following the lifecycle proposed by FIPA. Codifying an agent is defining the task it must accomplish during its execution; each task is an instance of Behaviors class. In addition, each Agent class will have methods to add and delete behaviors; this action may be done at any time in its lifecycle, depending on what the programmer had decided.

Communication architecture offers a flexible and efficient message passing mechanism; for each agent, the framework creates and manages a queue for private messages. The communication paradigm used is based on asynchronous message passing, which follows FIPA ACL standard. Each message is an object of ACLMessage class, and has methods for managing message parameters, for sending and receiving messages, for message filtering, etc.

Trying to facilitate the development of agents conversations, JADE offers a library where interaction protocols defined by FIPA (Query, Request, Contract-Net...) are implemented. Interaction protocols define the order and type of the messages involved in a conversation. For each task, it is necessary to add a behavior in the agent which initiates the conversation and other in each of the agents which will respond (AchieveREInitiator/ AchieveREResponder) and to implement the suitable handlers to initiate or respond each protocol message.

Agents must share semantics if the communication is to be effective. Therefore, exchanged messages must have a content written in a particular language and must share the same ontology. JADE provides support for using content languages and ontologies. That is, information is represented inside the agent as a JAVA object (easy to manipulate) and inside the message as a character sequence (easy to transfer). Working with both ways of information representation implies using an instance of ontology class and a codec for a content language (that is, an instance of Codec class). Ontology validates the information from a semantic point of view, while the codec translates it to a character sequence, following the syntactic rules of the content language. Having into account that building manually ontologies is tedious, JADE allows the utilization of other tools for automating its construction (for instance, Protégé for definition and Bean Generator for generating ontology classes).

The construction of mobile agents using JADE allows then migrating or copying themselves across multiple network hosts. A JADE mobile agent can navigate across different agent containers but inside a single JADE platform. Moving or cloning is considered a state transition in the life cycle of the agent, which can be initiated either by the agent or by the AMS. The two public methods doMove() and doClone() of the Agent class allow the agent to migrate elsewhere or to spawn a remote copy of itself under a different name.

JADE-LEAP is a runtime environment for enabling FIPA agents to execute on lightweight devices, such as mobile phones and PDAs.

JADE cannot run, as it is, on small devices due mainly to the complete JADE runtime environment is bigger than the usual memory of handheld devices and that it requires JDK1.4 (or later) while these devices only support PersonalJava or MIDP.

The LEAP add-on was created to solve these problems and allows deploying JADE agents on a wide range of devices. JADE-LEAP can be used in three different ways corresponding to the three types of Java environments: J2SE, PJAVA (to execute JADE-LEAP on handheld devices supporting PersonalJava like PDAs) and MIDP (for devices supporting only MIDP1.0 such as the majority of Java enabled cell phones). The three versions of JADE-LEAP provide the same set of API with minor differences to developers.

From the point of view of application developers and users JADE-LEAP for J2SE is almost identical to JADE.

6. SAIPE Implementation

In this section, some decisions and actions taken during implementation, which could help in system understanding, are addressed. These decisions are related to agent platform versions and development tools, system interfaces, agent behaviors, communication among agents, information source connections and security concerns.

JADE-LEAP 3.1 platform and the tools provided by it have been used as the agent platform for system. On the other hand, implementation has been done using NetBeans IDE 3.5.1, the Java application development interface.

Two kinds of graphical interfaces have been implemented. One of them is based on swing library and it is used in devices, which have a Java virtual machine 1.2, such as a personal computer. The other one uses AWT libraries for devices where the Java virtual machine is restricted to PersonalJava specification or to the more recent Java 2 Micro Edition (J2ME) in its CDC version. It is used in mobile phones and personal devices like a PDA. Figure 5 shows the SAIPE interface in this latest case.

These visual user interfaces have to be added to JADE-LEAP agents in order to provide interaction between users and interface agents. Interface agent inherits from GuiAgent provided by the platform. This GuiAgent also inherits from Agent Class because in JADE-LEAP all agents are subclasses of Agent class.

Figure 5: SAIPE Interface for PDA

The Interface Agent adds to its behavior list the suitable ones to answer a request, one for each which requires collaboration with other agents. These behaviors manage the communications with the agent in charge of satisfying the request, demanding it to execute the appropriate action and processing afterwards the obtained result. These communications follow the FIPA standard [6].

The Postman agent and the DocumentSearcher agent have a similar way of working than the Interface agent. At starting point, they add a cyclic behavior that waits for the arrival of messages. When one arrives, if it is understood, the agent sends an Agree message and adds to its behavior list the behavior, which executes the action requested in the message.

The behaviors that carry out actions are sequential. They are constituted of sub-behaviors that take place in a sequential way responding to the following steps. In first place they execute a behavior that moves the agent from user device to the device where the Main Container (this container is the place where the services of the platform are installed) is running. Afterwards, the behavior, which computes the answer to the requested action, is executed. Once the result is obtained, the migration behavior is executed again for coming back to user device. And, finally, the result from action is sent to Interface agent.

Mobile agents need a connection between user device and server device to be able to move. SAIPE has been developed and tested using wireless connections based on standard TCP/IP networks. To improve SAIPE connectivity in a short future other alternative technologies will be considered: Bluetooth connections, GPRS, Infrared connections, or UMTS.

Internal messages among agents are controlled by JADE-LEAP platform. These communications are done at a high level of abstraction using TCP/IP communications between JADE LEAP containers. Communications between two agents inside the same container are performed in the same way.

As it is been said, SAIPE allows searching POP3 mail accounts, searching web documents using Google, and locating personal files saved in server devices. POP3 email processing is implemented with JavaMail package. Postman agent reading mail

headers make the searches in mail accounts. The agent is also able to process multipart messages and carry enclosed files. Google searches are performed by DocumentSearch Agent using Google API. It is also capable of processing HTML Google interface to avoid Google API limitation to 1000 searches. Personal files are found using Java standard methods.

In a short future all database connections can be registered in the system in order to obtain relevant information for the user.

7. Conclusions and future work

This paper has shown how methodologies and tools have been used in mobile agents development, by constructing a distributed system where users connectivity is conditioned by external factors. The system has fulfilled the goals and expectations of its designers and has been successfully used in a Pocket PC HP 5500 with wireless connection.

INGENIAS methodology has facilitated the analysis and design of SAIPE. The models developed have been of great help in implementation process. In what refers to drawbacks, it must be highlighted that methodology does not integrate two basic aspects of SAIPE: ontology description and the detailed modeling of agent activities with respect to mobility. This lack is present both at a theoretical level and in the development tool.

Taking into account the platform used, JADE-LEAP has provided suitable mechanisms for multiagent system implementation. In particular the task of programming the agents and the communication among them has been made easier.

In the development of SAIPE using mobile agents it has become evident that an aspect that must be improved in JADE-LEAP is the management of errors oriented specifically to mobile agent models. This is a general drawback of implementation tools for agent-oriented system. In general and due to the intrinsic complexity of this kind of systems, there are not tools, such as simulators or proof managers, which help programmers to verify the developed programs. For this reason, proofs must be done directly causing the anomalous circumstances and verifying the results.

Mobile agents provide a faster processing of user tasks using a narrower network bandwidth. That is, SAIPE mobile agents only move relevant information using the network and heaviest calculus operations are done near the sources. The system is suitable for doing parallel processing performing, for instance, two different searches by two different mobile agents.

Among the possible extensions to SAIPE, one issue, which must be addressed, is an improvement in both internal and external security for mobile agents. Besides, the inclusion of new algorithms, which allow the agents to discover and access new information sources in an efficient and precise way, must be studied.

Because security concerns were not a goal of this first version, the system has been developed without considering any security mechanisms, that is, all the processes, actions and agents are considered trusty. Security of mobile agents must to be improved and latter versions will include security issues.

Acknowledgements

This research is supported by the Spanish national projects TIC 2002-04516-C03-01 and TIC 2001-5108-E.

References

[1] Bellifemine, F., et al, 2002. Jade Programmer's Guide (v.2.2) http://sharon.cselt.it/projects/jade/.

[2] J. Cai and D. Goodman. General packet radio service in GSM. IEEE Communications Magazine, 35:122--131, October 1997.

[3] Ciancarini, P. and Wooldridge, M., 2001. Agent-Oriented Software Engineering. First International Workshop AOSE 2000, Lecture Notes in Computer Science Vol. 1957. Springer-Verlag, Berlin.

[4] DeLoach, S. A., et al, 2001. Multiagent systems engineering. In The International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3.

[5] DeLoach, S. A. and Wood, M., 2001. Developing multiagent systems with agenttool. In Intelligent Agents VII. Agent Theories Architectures and Languages, 7th International

Workshop (ATAL 2000), C. Castelfranchi, Y. Lesperance (Eds.). Lecture Notes in Computer Science. Vol. 1986, Springer Verlag, Berlin.

[6] FIPA, 2000. FIPA Specifications. http://www.fipa.org/.

[7] Gómez-Sanz, J. J. 2002. Modelado de Sistemas Multi-Agente. PhD thesis, Departamento de Sistemas Informáticos y Programación, Universidad Complutense Madrid.

[8] Gómez-Sanz, J., Pavón, J. 2004. Methodologies for Developing Multi-Agent Systems. Journal of Universal Computer Science, 10 (4), 359-374.

[9] Ingenias Development Kit (IDK), http://ingenias.sourceforge.net/

[10] Ivar Jacobson, Grady Booch, and Jim Rumbaugh, 1999. Unified Software Development Process, AddisonWesley.

[11] Java Agent Development Framework: http://sharon.cselt.it/projects/jade/

[12] Jennings, N. R., 1999. Agent-based computing: Promise and perils. In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol2). Stockholm, Sweden, pp. 1429-1436.

[13] Lange, D. and Oshima, M., 1999, Seven Good Reasons for Using Mobile Agents, In Communication of ACM, vol. 42, no. 3, pp. 88-89.

[14] Odell, J., Van Dyke Parunak, H. and Bauer, B., 2000. Extending UML for Agents, Proceedings of the Agent-Oriented Information System, Workshop at the 17 National Conference on Artificial Intelligence, pp. 3-17, Austin, USA.

[15] O'Malley, S.A, and DeLoach, S.A., 2002. Determining When to Use an Agent-Oriented Software Engineering Paradigm. Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE-2001), Lecture Notes in Computer Science, Vol. 2222. pp.188-2005.

[16] Sycara, K. P., 1998. Multiagent systems. The AI Magazine, Vol. 10, No. 2, pp. 79-92.

[17] Wooldridge, M. 2002. An Introduction to Multiagent Systems. Published by John Wiley & Sons.

[18] Weiss, G., 1999. Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, USA

[19] The ZEUS Agent Building Toolkit: http://more.btexact.com/projects/agents/zeus/