Some Issues and Extensions of JADE to Cope with Multi-Agent Operation in the Context of Ambient Intelligence

A. Paz-Lopez, G. Varela, S. Vazquez-Rodriguez, J. A. Becerra, R. J. Duro

Integrated Group for Engineering Research, University of A Coruna

Mendizabal s/n, 15403, Ferrol, Spain

www.gii.udc.es

Abstract Ambient Intelligence (AmI) can be taken as one of the principal test beds for the integration of Systems, Humans and Cybernetics. Multi-agent based Ambient Intelligence solutions have proliferated in the last few years in order to cope with the intrinsically distributed and complex interaction based nature of the problem. Different multi-agent platforms have been used in this regard, being JADE one of the more standard solutions adopted due the variety of services it provides and to its FIPA compliance. Nevertheless, JADE was designed as a general Multi-agent platform and not for the particular demands of AmI systems. Consequently, its behavior in this realm is not as optimal as it could be. In this paper, we analyze some of the problems JADE presents for its application to AmI, especially in terms of communications, and describe some of the extensions we have developed in order to solve them. The resulting system exhibits enhanced communication capabilities, promotes the division of tasks into decoupled components and makes component reutilization easy. This platform constitutes an environment with tools for the development and deployment of AmI applications.

1 Introduction

As already pointed out by [1], Ambient Intelligence (AmI) is a multidisciplinary paradigm that "fosters novel anthropomorphic human–machine models of interaction". In this field, and taking inspiration from Ubiquitous Computing ideas, the technology disappears into the background, allowing the user to take control of its environment in a natural and completely transparent way. In fact, the idea is to make the environment proactive so that the user may have this control without even being aware of it. Many authors have worked on applying artificial intelligence based approaches to achieve this end in contexts like smart homes, elderly and health care or leisure activities, among others [2, 3].

Several problems arise when implementing such systems, especially due to the large number of sensors and distributed infrastructure they require to perform even the simplest tasks and consequently, to the ever more complex and abundant communications that must be handled as the systems scale up. These may be classified into hardware organization and integration related problems and software modularization and scaling related problems. Different and stimulating solutions have been proposed in terms of whole architectures from two points of view. On one hand, numerous solutions have adopted a service oriented approach like the PERSONA project [3], the AMIGO project [4] or [5]. On the other hand, some groups have resorted to agent based middleware for AmI such as in the case of the AIRE Project [6], the CHIL project [7] or our own HI^3 project, structured as an agent based layered architecture [8, 9].

Multi-Agent System (MAS) based platforms, when adequately designed, are a natural way to distribute intelligence and provide for component mobility, autonomy, scalability and fault tolerance [10]. In this line, many authors have resorted to tools and platforms that were developed for MAS but not necessarily for AmI [11, 12]. Thus, even though they simplify the generation and management of the MAS, many of the functionalities and requirements that arise in AmI have to be dealt with in an ad hoc manner. This is the case of the most widely used standard MAS platform, the JADE agent platform [13]. It provides basic capabilities like inter-agent distributed communication, simple agent mobility, basic support for agent discovery and FIPA [14] compatibility. However, to achieve some of the requirements and be able to easily implement high level and AmI related features, JADE must be extended in certain regards. Some authors have already introduced some extensions aimed at their specific problems. A clear instance of this is the work by Moraitis and col. [11] with their Im@gine IT project [15] where they extended the FIPA architecture through the introduction of new agent types. However, they did not deal with some of the infrastructure or support elements that would be required for all agents to perform their tasks in a more efficient and trustworthy manner. These are the types of extensions we will consider here, they are not agents, but rather services provided by the platform for the agents to use as required such as improvements in the inter-agent communication capabilities, facilities to manage multi-agent conversations, introduction of a publish/subscribe communication paradigm or improved multi-agent model with support for declarative definition of components. All of this within a multi-agent system execution container with distributed management tools and development utilities to facilitate the implementation and deployment of new applications. Thus, the paper is organized as follows. Section 2 deals with the different extensions introduced on the communications. Next, in section 3, a declarative component model for AmI MAS systems is presented along with its related tools. Finally some conclusions are extracted.

2 Improving Agent Communication

Even though JADE provides good basic capabilities for inter-agent communication there is a need to introduce some features that would help to adapt them to the special characteristics of AmI environments. Here we will describe three areas where extensions have been created: Inter-agent message interchange, multi-agent conversations and a publish/subscribe communications system.

In terms of Inter-agent Message Interchange, JADE provides a basic system for asynchronous message emission and reception that invites developers to code agents communications mixed with control logic in an ad-hoc manner. Due to this fact it is expensive to develop high level communication capabilities like agents with the ability to automatically know how to interact with other agents. One way to alleviate this problem could be to upgrade agent capabilities with a mechanism to encourage the decoupling between communications and control logic code.

Thus, we start by introducing the concept of Message Processor. A Message Processor is the most basic way of receiving messages in this extended platform. It is a behaviour that is associated to one or more message templates (they specify characteristics a message must fulfill) and is specialized on processing received messages. When a message is received that fulfills some message template, its associated Message Processor is woken up and the message is forwarded to it for processing. Message Processor elements will be in charge of applying any required preprocessing to the messages, decoupling it from the control logic to make it reusable, and finally, deliver the message to an adequate control behavior.

The message templates associated to each Message Processor completely specify the communications interface of an agent in any instant of time (Message Processors and templates can be added or removed dynamically). They thus provide a straightforward procedure to decouple communications code from the agent's functional logic and, when combined with the declarative agent model that will be presented later, provides basic support for declarative communications.

On top of the Message Processor concept, a priority based scheduling system for message reception was introduced. It provides developers with a mechanism to prioritize some agent interaction capabilities over others in order to focus the agent's attention on its most critical tasks.

Regarding Multi-Agent conversations, the JADE agent platform provides support for some predefined agent communications (even 1-to-N communications) based on the interaction protocols proposed by the FIPA standard. However, in a complex and broad environment like AmI, where there will be a large number of heterogeneous agents from multiple vendors, with complex behaviours and their own goals, it seems difficult to establish every complex interaction in a predefined way, especially when one of the goals of AmI is to build systems that autonomously evolve and adapt to the environment and its users.

One way to cope with those complex interactions could be to organize agents constituting societies and to provide them with mechanisms to participate in

conversations with multiple agents through generic models that abstract them from the management logic of the interaction. Thus, the concept of conversation was introduced in the extended platform as a way to support multi-agent (N-to-M) interactions. Every agent has the integrated capability of being a member of a conversation. Therefore, (see Fig. 1) an agent is inherently capable of creating new conversations, of inviting new agents to a conversation, of sending/receiving messages within a conversation and of being informed about agents that enter or leave a conversation. As interactions between multiple agents can become really complex, the system is designed to easily support multiple types of conversations through the concept of Conversation Model.

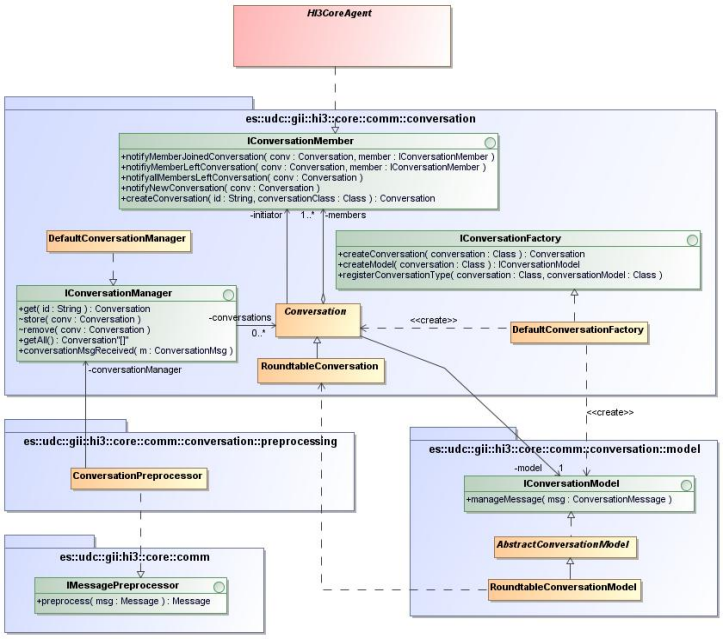

Fig. 1. Class diagram of the conversation management subsystem

Fig. 1 shows a simplified class diagram of the conversation subsystem design. It shows how conversations are controlled by a Conversation Manager that tracks messages of every known conversation by using an adequate Conversation Model implementation. These models contain the logic to drive a particular type of conversation, including how to invite new members to a conversation, how to leave a conversation or how to notify members of message reception.

In addition to the previously described communication models, we have added an extension for publish/subscribe communications. In this model, agents that wish to send some kind of information publish it as events, whereas agents interested in receiving certain types of events subscribe to those events.

A publish/subscribe model is a loosely coupled communication mechanism in which a publisher of an event is not necessarily aware of the recipients interested

in that event or even of their existence. This communications model fits perfectly with typical interactions present in AmI environments like notifications about state changes in sensors, access to information about the lifecycle of other components or the platform itself, or sharing data between components.

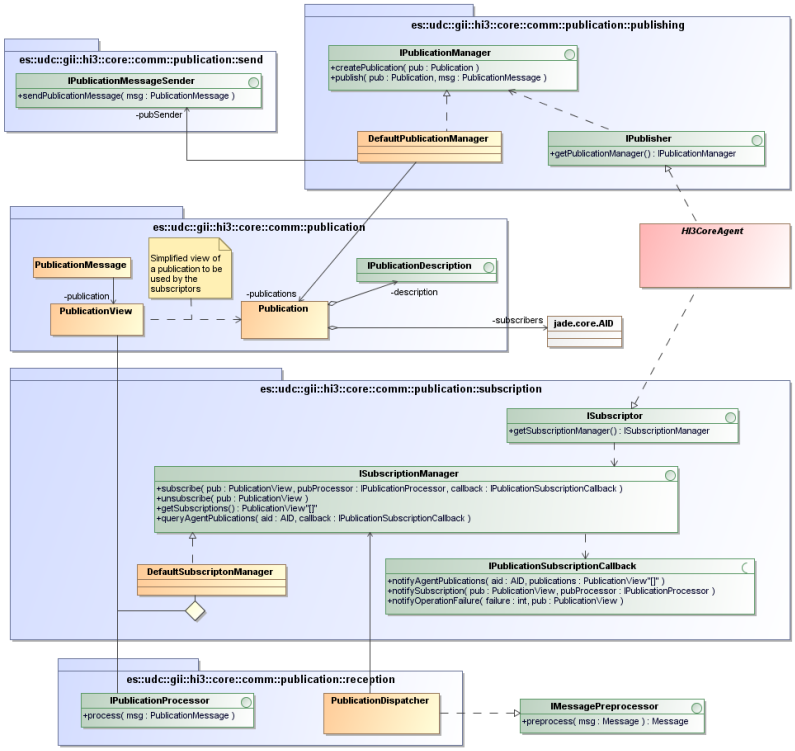

Fig. 2. Simplified class diagram of the publish/subscribe subsystem

As JADE does not provide a communication model of this nature, we have developed a publish/subscribe system on top of the communications extensions previously described. Within this system, agents inherently have the ability to create publications, publish content on those publications and subscribe to publications from other agents.

A simplified class diagram of the publish/subscribe subsystem is shown In Fig. 2. The management of the publications and subscriptions is decoupled from the agents by the use of a publication and a subscription manager that encapsulates the logic and data required to receive publications and to send content to the subscribed components. These managers rely on the message interchange subsystem of the extended platform described previously. Furthermore, developers can, in a declarative manner, define and describe the set of publications for a given agent. Then, at run-time, other heterogeneous agents can discover publications and dynamically subscribe and unsubscribe to them.

3 Multi-Agent Declarative Model

As mentioned before, AmI systems tend to be really large and complex systems but, at the same time, they usually share a lot of common functionalities. One possible option to alleviate this complexity is to promote the division of the system into highly decoupled components. This is one of the main purposes of our extended platform from a conceptual point of view and it is supported in the software platform by what we call the Multi-agent declarative model.

The Multi-agent declarative model is, on one hand, a model to describe a multi-agent system and its components and, on the other, a set of tools to declaratively define these components. As a model it brings together all the different elements the platform leaves in the hands of the developers to build new applications. The top level concept of the platform is the multi-agent system (MAS). The platform is prepared to support various types of MAS with different high level characteristics but, in its basic form, they are a collection of instances of different types of agents that work together to perform a task.

Directly below the MAS concept is the agent-type, which specifies the components that define an agent in the platform, providing a way to declaratively build new agent-types by reusing preexisting components and mixing them with new ones. Agent-types are defined according to a set of concepts that also have independent and declarative definitions:

- Behaviours. They contain the implementation of the agent's functionality.
- Message Processors. Define the communications interface of an agent.
- Publications. Specify the publications that the agent offers for subscription.
- Arguments. Allow the parameterization of agents at instantiation time.
- Functionalities. Describe the functionalities that an agent wants to publicly advertise to other agents. When an agent instance is launched, the platform automatically announces its functionalities on the platform agent registry.

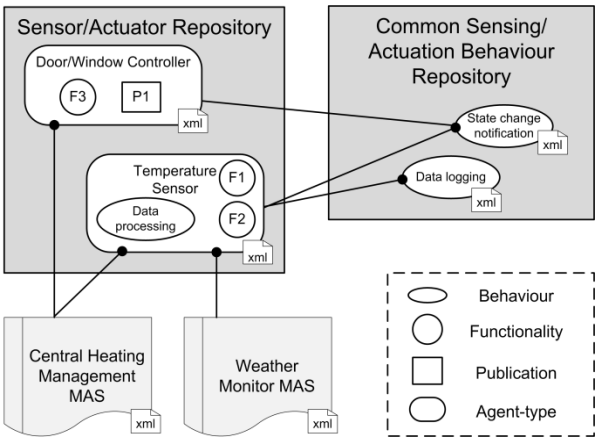

Fig. 3. Declarative model usage and component reutilization example

Furthermore, Agent-types support inheritance. An Agent-type can inherit the components specified for its parent Agent-type, so that the instances of the child Agent-type will have the behaviours, message processors, etc. of the parent.

This conceptual model is backed by a set of tools that provide declarative support to define and use these concepts. These tools are mainly made up of an XML based language for component description, and utilities to create and manage these description files.

Every major component has its own definition in an XML file, making them independent and reusable. To support that reusability, an URL based system was designed for component referencing. This provides a flexible and transparent way to distribute and reference reusable components.

An example of using the Multi-agent declarative model is shown in Fig. 3. It is easy to see how every component has its own independent definition, except those that are internal to other components, like the "Data processing" behavior of the "Temperature Sensor". It also shows the use of component repositories, from which new components can be created by the aggregation of preexisting ones.

4 Conclusions

JADE has been revealed as a suitable MAS platform in many contexts, but, in order to be really usable in the context of AmI it requires of a set of extensions that enhance its functionalities and adapts them to the problems it will really find. In this paper we have argued for and described some of these enhancements.

Here a new subsystem for message reception and processing was implemented with support to decouple communications code from functional logic. This subsystem provides the basics to build systems in which components can clearly specify their communications interface to others. In addition, two high level communications models that provide integrated support for typical interactions of the AmI world, like dynamically changing interactions between multiple components, through the conversation concept, or highly decoupled event-based communications through a publish/subscribe model have been created. Finally, a component conceptual model has been introduced on top of JADE agents and behaviours. It decouples a system into a set of multiple independent components that can be easily shared between multiple systems. This model is supported by tools like an XML based language to describe the components and an URL system for transparent component referencing and aggregation. As a whole, it provides a standardized way to divide a system into components and an easy way to reutilize logic between systems.

Summarizing, to adapt JADE to the realm of AmI, some of its functionalities must be extended and adapted to this context. This paper is a first step in this direction and we expect to build new functionalities in a structured way to make the design and deployment of AmI systems simpler and more reliable.

Acknowledgments This work was partially funded by the Ministerio de Educación y Ciencia of Spain through project DEP2006-56158-C03-02.

References

1. P. Remagnino and G. L. Foresti, "Ambient Intelligence: A New Multidisciplinary Paradigm", IEEE Trans. SMC- Part A, Vol. 35, No. 1, Jan. 2005, pp.1-6
2. O. Brdiczka, J. L. Crowley and P. Reignier, "Learning situation models in a smart home". IEEE Trans. SMC - Part B, Vol. 39(1), 56-63, 2009.
3. PERSONA, "PERceptive Spaces promoting iNdependent Aging", http://www.aal-persona.org/deliverables.php
4. AMIGO, "Ambient intelligence for the networked home environment", http://www.hitech-projects.com/euprojects/amigo/software.htm
5. Chao-Lin Wu, Chun-Feng Liao, Li-Chen Fu, "Service-Oriented Smart-Home Architecture Based on OSGi and Mobile Agent Technology", IEEE Trans. on SMC, 2007, vol. 37,no. 2, pp 193-205.
6. S. Peters, G. Look, K. Quigley, H. Shrobe and K. Gajos, "Hyperglue: Designing High-Level Agent Communication for Distributed Applications", Proceedings of AAMAS'03, 2003.
7. J. Soldatos, N. Dimakis, K. Stamatis and L. Polymenakos, "A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications", Personal and Ubiquitous Computing, Vol 11, Issue 3, 193-212, 2007.
8. G. Varela, A. Paz-Lopez, S. Vazquez-Rodriguez and R. J. Duro, "HI3 Project: Desing and Implementation of the Lower Level Layers", Proc..IEEE VECIMS, 2007, pp. 1-6.
9. A. Paz-Lopez, G. Varela, J. Monroy, S. Vazquez-Rodriguez and R. J. Duro, "HI3 Project: General Purpose Ambient Intelligence Architecture", Proc. AITAmI'08, 2008, pp. 77-81
10. G. M. P. O'Hare, M. J. O'Grady, S. Keegan, D. O'Kane, R. Tynan and D. Marsh, "Intelligent Agile Agents: Active Enablers for Ambient Intelligence", Proc. of the Ambient Intelligence for Scientific Discovery Conference, 2004.
11. N. I. Spanoudakis and P. Moraitis, "Agent Based Architecture in an Ambient Intelligence Context", EUMAS, 2006.
12. F. Doctor, H. Hagras, and V. Callaghan, "A Fuzzy Embedded Agent-Based Approach for Realizing Ambient Intelligence in Intelligent Inhabited Environments", IEEE Trans. SMC- Part A, Vol. 35, No. 1, January 2005, pp 55-65.
13. JADE. Java Agent DEvelopment Framework, http://jade.tilab.com
14. FIPA Agent Com. Lang. Spec., http://www.fipa.org/repository/aclspecs.html.
15. P. Moraitis, E. Petraki and N. Spanoudakis, "An Agent-Based System for Infomobility Services", EUMAS2005, Brussels, Belgium, 2005.