The Journal of Systems and Software 83 (2010) 467–484

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A modern approach to multiagent development

D. Vallejo *, J. Albusac, J.A. Mateos, C. Glez-Morcillo, L. Jimenez

School of Computer Science, University of Castilla-La Mancha, Paseo de la Universidad, 4, 13071 Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 28 January 2009
Received in revised form 18 September 2009
Accepted 18 September 2009
Available online 27 September 2009

Keywords:
Agent technology
Multiagent architecture
Distributed artificial intelligence

ABSTRACT

Multiagent systems (MAS) development frameworks aim at facilitating the development and administration of agent-based applications. Currently relevant tools, such as JADE, offer huge possibilities but they are generally linked to a specific technology (commonly Java). This fact may limit some application domains when deploying MAS, such as low efficiency or programming language restrictions. To contribute to the evolution of multiagent development tools and to overcome these constraints, we introduce a multiagent platform based on the FIPA standards and built on top of a modern object-oriented middleware. Experimental results prove the scalability and the short response-time of the proposal and justify the design and development of modern tools to contribute the multiagent technology.

1. Introduction

Agent-Oriented Programming (AOP) (Jennings, 2000) is defined as a software development paradigm that combines Artificial Intelligence and Distributed Systems. Basically, AOP conceives an application as a set of software components named agents (Wooldridge and Jennings, 1995), which are autonomous and proactive entities that are able to communicate with one another. The architectural model of agent-based systems is inherently distributed and is based on a peer-to-peer scheme, where each agent cooperates and communicates with others when it is needed.

Currently, multiagent systems (MAS) are being applied as a solution to a wide range of problems (Weiss, 1999), such as planning, scheduling systems, real-time control, robotics, and more industrial fields. This expansion in the use of MAS is also captured in software engineering models (Jennings, 2000) based on the use of autonomous agents to solve complex and distributed problems, the definition of methodologies (Wooldridge and Ciancarini, 2001), the use of languages (Bordini et al., 2006), or even the adoption of standards (Foundation for Intelligent Physical Agents, 2002a). In other words, AOP is used to deal with the design of problems in which other approaches are insufficient or incomplete.

This evolution can also be appreciated in MAS software development platforms (see Table 1). Thanks to these tools, agent-oriented system developers can focus their work on the multiagent application instead of dealing with administrative issues. For the time being, JADE (Java Agent DEvelopment framework) (Bellifemine et al., 2008) is probably the most widespread agent-oriented middleware. JADE can be defined as a modular and distributed framework that facilitates the development of agent-based applications. This agent framework implements the agents' life cycle and the management logic by providing administrative tools to deploy, monitor, and debug multiagent systems (Bellifemine et al., 2007).

Developers commonly use some existing framework in order to do the final deployment and not to spend too time in management and communication issues. The main advantages of this choice are an easy configuration and the administrative facilities derived from the agent middleware. Nevertheless, the main drawback is that the developer will be conditioned by the characteristics of the chosen framework.

Most current agent-oriented middlewares (see Table 1 in Section 2.2), although developed with the idea of providing general-purpose tools applicable to a wide range of domains, have several limitations that are directly related to their particular developments:

- Most of them are linked to a technology that limits their expansion to some platforms (due to hardware or resource consumption constraints).
- Most of them use technologies based on Java, which decreases the performance due to the Java Virtual Machine technology (see the experimental results in Section 4.2).

Within this context, several application domains, such as the systems that need short response-time, require solutions that overcome the previous limitations. To address these issues, the agent community needs to provide solutions that contribute to

* Corresponding author. Fax: +34 926 295 354.
E-mail address: David.Vallejo@uclm.es (D. Vallejo).
URL: http://personal.oreto.inf-cr.uclm.es/dvallejo (D. Vallejo).

0164-1212/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2009.09.036

Table 1
Currently relevant agent-oriented software development platforms.

Name	Language	Standard	Free	Relevant paper
JADE	Java	FIPA	Yes	Bellifemine et al. (2008)
Jadex	Java	FIPA	Yes	Pokahr et al. (2005)
Cougaar	Java	No	Yes	Helsinger and Wright (2005)
Agent factory	Java	FIPA	Yes	Ross et al. (2004)
JACK	Java	No	No	Winikoff (2006)

the evolution of platforms for developing multiagent systems and solve the limitations of existing frameworks. Therefore, one of our goals is to use a modern object-oriented communication middleware (Henning, 2004) that allows us to extend the platform proposed in this work to different programming languages, operating systems, and hardware environments.

Another critical design topic to take into account in this evolution is the adoption of standards to promote the interoperability between MAS. In this context, the FIPA (Foundation for Intelligent Physical Agents) committee defines the most widespread proposal within the multiagent field. In this way, another of the central themes of our approach consists in adopting the set of FIPA specifications for the development of MAS. In fact, the last goal is to keep spreading the agent technology by providing an agent platform that facilitates the agent management and answers the challenge posed in (Ricci and Nguyen, 2007) regarding the deployment of MAS in real environments.

To face these challenges, this work introduces the design and development of an agent platform that follows the FIPA specifications and facilitates the development of agent-oriented applications. The main goal of our approach is to provide an alternative agent development framework that overcomes the previously discussed limitations. The design of this agent platform is heavily based on the use of well-known software engineering issues, such as design patterns and templates, and supported by ZeroC ICE (Henning, 2004), a modern object-oriented communication middleware that provides tools, APIs, and libraries to build object-oriented client–server distributed applications. ICE represents a modern alternative to develop distributed systems, which is based on CORBA's principles but with a lower complexity and a higher cohesion to make easy its use and learning (Henning, 2006).

The justification of using an object-oriented approach is due to two main reasons: (i) agents are actually an evolution of objects but with new characteristics such as autonomy and proactiveness (Wooldridge, 2001), and (ii) FIPA semantics shows object-oriented notions to describe concepts and ontologies (Foundation for Intelligent Physical Agents, 2004). This choice facilitates both the development of the agent platform itself and the development of agent-based applications on behalf of external developers.

In order to validate the agent platform, we have run two series of tests that measure its response-time and efficiency. The first one is related to an environment with multiple computers in which the agents send a variable number of messages with a determined size. Within this context, we evaluate the total traffic, the time spent in spamming messages, and the processing time spent in receiving messages depending on the number of messages sent and the message size. The second one refers to the deployment of agents by means of the named agent factories. The goal of this set of tests is evaluating the impact of hibernating and activating agents implemented in different programming languages by considering the agent creation time and the agent reactivation time. Finally, we briefly describe two scenarios where the agent platform proposed in this work has been used to successfully deploy multiagent applications.

The rest of the paper is organised as follows. Section 2 positions our work in the context of existing standards and multiagent platforms. Section 3 describes and discusses in depth the design and the development of the different components of the proposed platform. Section 4 presents the experiments carried out to validate the work and its application in two real problems. Finally, Section 5 concludes the paper and suggests future research lines.

2. Related work

In the last decade there has been an important number of software developments in different MAS related fields, ranging from declarative and imperative languages, integrated development environments, to agent platforms and frameworks (see Bordini et al., 2006 for a recent survey). Since this paper introduces a framework for the development of FIPA related MAS, we position our work in the context of existing standards and platforms.

2.1. MAS standards

2.1.1. Foundation for Intelligent Physical Agents (FIPA)

The Foundation for Intelligent Physical Agents (FIPA) is an IEEE committee aimed at promoting agent-based technology and interoperability between agent-based applications. FIPA specifications provide a set of standards for agents to interoperate at different levels. One of the more relevant documents is the FIPA Abstract Architecture Specification (Foundation for Intelligent Physical Agents, 2002a). This document and its derived specifications define the abstract architecture proposed by FIPA for the development of MAS (see Fig. 1). The main advantage of adopting the set of FIPA standards is the maturity obtained after more than 12 years of producing specifications for heterogeneous and interacting agents and agent-based systems.

The FIPA Agent Management Specification (Foundation for Intelligent Physical Agents, 2004) establishes the agent management model of the agent platform, including the basic FIPA management services, the management ontologies, and the message transport model. The first relevant service is the Directory Facilitator (DF), which is responsible for providing agents with a yellow-pages service. Next, the Agent Management System acts as the platform manager and maintains a directory with the valid agent identifiers (AIDs) of the agents registered with the platform. Besides providing the agents with a white pages service, the AMS also manages the agent life cycle. Finally, the Message Transport Service (MTS) provides support to send and receive Agent Communication Language (ACL) messages.

2.1.2. OMG's agent PSIG

The mission of the Agent Platform Special Interest Group (OMG) (PSIG) is to work with the OMG (Object Management Group) platform in order to stimulate the creation of agent specifications directly related to OMG technology (Odell, 2000). The relevant goals are to recommend OMG extensions to deal with agent tech-

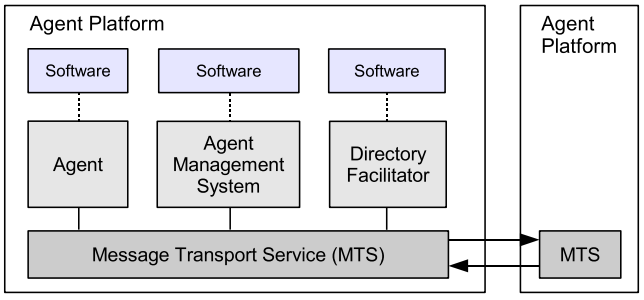

Fig. 1. FIPA abstract architecture.