Platform for Building Large-Scale Agent-Based Systems

Carolina Zato, Alejandro Sánchez, Gabriel Villarrubia,
Sara Rodríguez and Juan Manuel Corchado
Department of Computer Science and Automation
University of Salamanca
Salamanca, Spain
{carol_zato, asanchezyu, gvg, srg, corchado}@usal.es

Javier Bajo
Department of Computing Science
Pontifical University of Salamanca
Salamanca, Spain
jbajope@upsa.es

***Abstract*—This paper presents an agent platform called PANGEA (Platform for Automatic coNstruction of orGanizations of intElligent Agents). This platform allows to developed multiagent systems modeled as Virtual Organizations. The concepts of roles, organizations and norms are fully supported by the platform assuring flexibility and scalability. Moreover, a communication protocol based on IRC gives high performance and reliability to this kind of distributed systems.**

Keywords-component; agent platform, distributed systems, virtual organizations, IRC protocol.

I. INTRODUCTION

Distributed multi-agent systems have become increasingly sophisticated in recent years, with the growing potential to handle large volumes of data and coordinate the operations of many organizations [24]. Distributed intelligent systems are intelligent systems built on a distributed computer system. They are based on the use of cooperative agents, and are organized in hardware or software components. In the system, each agent independently handles a small set of specialized tasks and cooperates to achieve the system-level goals and a high degree of flexibility [26]. Multiagent systems have become the most effective and widely used form of developing this type of application in which communication among various devices must be both reliable and efficient. One of the problems related to distributed computing is message passing, which is in turn related to the interaction and coordination among intelligent agents. Consequently, a multiagent architecture must necessarily provide a robust communication platform and mechanisms to control and correct any problems.

There is additionally a rapidly growing use of mobile devices with limitless connection possibilities and computing power. This creates another problem for distributed systems, which must now be able to accept requests from different devices. From a practical perspective, multiagent systems face yet another challenge, that of obtaining light intelligent agents that can be deployed in this type of device without relinquishing their capabilities.

This article presents a multiagent platform based on a Virtual Organization paradigm. The platform includes a robust communication model that allows intelligent agents to connect from a variety of devices. These agents can be developed both quickly and in any language using the tools provided by the platform.

The remainder of the paper is structured as follows: the next section introduces some existing platforms. Section 3 presents an overview of the main characteristics of the platform. Finally, section 4 explains some results and conclusions.

II. RELATED WORKS

As previously mentioned, the multiagent system was based on a Virtual Agent Organization [7]. Consequently, the platform makes it possible to create open systems that resolve the inflexibility of a multiagent system whose characteristics must be modeled during the design and implementation phases, but are difficult to adapt during the execution phase. This can be achieved by virtual agent organizations, which provide communities, roles, norms and other organizational and institutional concepts [6] that are modeled in way that allows the architecture to offer flexibility for the configuration of its use during the execution phase.

This is a step forward with regards to flexibility, although it entails greater effort in the coordination and interaction among agents, which in turn implies a greater number of messages, itself a key factor in distributed systems. This usually requires a common language, an Agent Communication Language, or ACL. Much work has been done in developing ACLs that are declarative, syntactically simple, and readable by people. KQML [11] and FIPA-ACL [13] are two of the most widely used ACLs in multi-agent systems [10].

The platforms for the development of the most successful multiagent architectures to date, including FIPA-OS [9], JADE (Java Agent Development Framework) [12], Jadex [15], use FIPA-ACL. Other platforms such as Cougaar [25] use KQML language or JASON [27][28], which uses SACI (Simple Agent Communication Infrastructure) [14] based on KQML.

Focusing on platforms that create Virtual Organizations we find the same scenario. MadKit [20], Jack Teams [23], J-Moise+ [22], THOMAS [21] use one of the previous implementations.

PANGEA will not pretend to present a new communication protocol; instead it will introduce the IRC protocol within multiagent systems. This protocol is widely used in other

distributed environments and has already demonstrated its reliability and robustness. What is proposed is its use within the platform, providing advantages, such as ease of implementation and reliability, given that it has been widely used in online communities with good functionality.

III. Description Of The Platform

In general terms, the proposed platform includes the following characteristics:

- Different models of agents, including a BDI and CBR-BDI architecture.
- Control the life cycle of agents with graphic tools.
- A communication protocol that allows broadcast communication, multicast according to the roles or suborganizations, or agent to agent.
- A debugging tool.
- Module for interacting with FIPA-ACL agents.
- Service management and tools for discovering services.
- Web services.
- Allow organizations with any topology.
- Organization management.
- Services for dynamically reorganizing the organization.
- Services for distributing tasks and balancing the workload.
- A business rules engine to ensure compliance with the standards established for the proper operation of the organization.
- Programmed in Java and easily extensible.
- Possibility of having agents in various platforms (Windows, Linux, MaccOS, Android and IOS)
- Interface to oversee the organizations.

When launching the main container of execution, the communication system is initiated; the agent platform then automatically provides the following agents to facilitate the control of the organization:

- OrganizationManager: the agent responsible for the actual management of organizations and suborganizations. It is responsible for verifying the entry and exit of agents, and for assigning roles. To carry out these tasks, it works with the OrganizationAgent, which is a specialized version of this agent.
- InformationAgent: the agent responsible for accessing the database containing all pertinent system information.
- ServiceAgent: the agent responsible for recording and controlling the operation of services offered by the agents.
- NormAgent: the agent that ensures compliance with all the refined norms in the organization.
- CommunicationAgent: the agent responsible for controlling communication among agents, and for recording the interaction between agents and organizations.
- Sniffer: manages the message history and filters information by controlling communication initiated by queries.

We want to create a service oriented platform that can take maximum advantage of the distribution of resources. To this end, all services are implemented as web services. This makes it possible for the platform to include both a service provider agent and a consumer agent, thus emulating a client-server architecture. The provider agent knows how to contact the web service; once the client agent's request has been received, the provider agent extracts the required parameters and establishes the contact. Once received, the results are sent to the client agent.

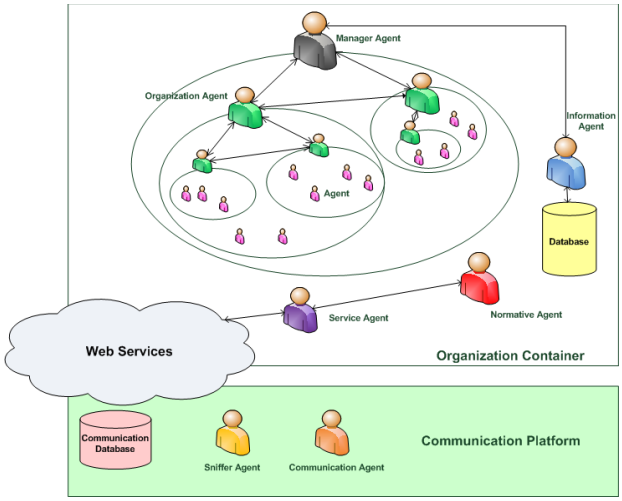

Figure 1. Platform Overview

A. Communication Platform

This section will focus on describing the communication platform and protocol. As observed in Figure 1, the communication platform includes two main agents: the CommunicationAgent and the Sniffer. The first is in charge of checking the connections to confirm that the agents are online and see which ones have disconnected. It is also in continual communication with the NormAgent to ensure that the agents respect the lines of communication and comply with the standards. The Sniffer is in charge of recording all communication, offers services so that other agents can obtain history information, and facilitates the control of information flow for programmers and users.

The IRC protocol was used to implement communication. Internet Relay Chat (IRC) is a Real Time Internet Protocol for

simultaneous text messaging or conferencing. This protocol is regulated by 5 standards: RFC1459 [1], RFC2810 [5], RFC2811 [4], RFC2812 [2] y RFC2813 [3]. It is designed primarily for group conversations in discussion forums and channel calls, but also allows private messaging for one on one communications, and data transfers, including file exchanges [25]. The protocol in the OSI model is located on the application layer and uses TCP or alternatively TLS [29]. An IRC server can connect with other IRC servers to expand the user network. Users access the IRC networks by connecting a client to a server. There have been many implementations of clients, including mIRC or XChat. The original protocol is based on flat text (although it was subsequently expanded), and used TCP port 6667 as its primary port, or other nearby ports (for example TCP ports 6660-6669, 7000) [26]. The standard structure for an IRC server network is a tree configuration. The messages are routed only through those nodes that are strictly necessary; however, the network status is sent to all servers. When a message must be sent to multiple recipients, it is sent similar to a multidiffusion; that is, each message is sent to a network link only once [29]. This is a strong point in its favor compared to the no-multicast protocols such as SimpleMail Transfer Protocol (SMTP) or the Extensible Messaging and Presence Protocol (XMPP).

One of the most important features that characterize the platform is the use of the IRC protocol for communication among agents. This allows for the use of a protocol that is easy to implement, flexible and robust. The open standard protocol enables its continuous evolution. There are also IRC clients for all operating systems, including mobile devices.

All messages include the following format: *prefix command command-parameters\r\n*. The prefix may be optional in some messages, and required only for entering messages; the command is one of the originals from the IRC standard.

Another advantage in using IRC involves the ease in implementing communication. The platform's code generating tool makes it possible to easily create an outline of an agent, with the communication code requiring few lines of code. The Figure 2 displays the code for an agent in C#. It is clear that the functionality of the code consists in associating different events to the *OnqueryMessage* method, intercepting when an agent receives a message or enters an organization, and effectively handling that action from the *OnqueryMessage* method. The Connect method specified the host and the communication server port, which is responsible for all agents to connect and communicate. The *OnRawMessage* event is responsible for intercepting all server responses.

```
private void conect (Object sender, EventArgs e){
irc.OnJoin += new JoinEventHandler(OnQueryMessage);
irc.OnQueryMessage += new IrcEventHandler(OnQueryMessage);
irc.OnRawMessage += new IrcEventHandler(OnRawMessage):
irc.Connect(host.Text, 6667);
irc.Login(agent,Text, null);
irc.Listen()
}
```

Figure 2. Example of the connection code for an C# agent

IV. Results and Conclusions

In order to test communication among the agents deployed in the platform, a test case was designed, as shown in Figure 2. The first building contains a machine with an Intel Core 2 CPU 6600 2.4GHz processor with 4Gb RAM and a 64 bit operating system connected to a network with a symmetrical speed of 2 Mb. The communication server is installed on this machine. Located in the same building but on a different network, there are 4 PCs each containing 15 agents. In another building located 4 kms away, 2 laptops, each containing 30 agents, and a PC with 600 agents are connected. In a third building, 10 agents join the platform from a connected PC; 4 agents are connected from Android Smartphones, and 2 from iPhones. The purpose of the study is to test communication. Consequently, the 736 agents involved do not carry out complex computational tasks; instead they simply request information from web-based news services or other basic services configured within the platform.

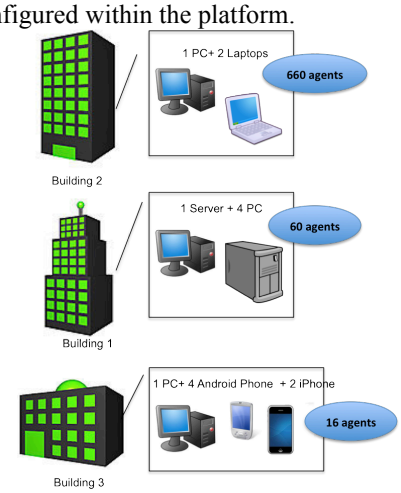

Figure 3. Test Case

Two tests were carried out. The first was performed on a Monday with agents active during a 24 hour period, while the second was performed on a Friday with agents functioning during an 18 hour period, so that different network conditions could be analyzed. As previously explained, the agents can be developed in any language that uses sockets to enable communication. The following table displays the number of agents according to the language implemented, and the number of disconnected agents. The third column lists the number of resent messages. Since all messages should arrive to the destination agent, the platform configures time-outs to resend the messages. This column, therefore, represents the number of messages that were unable to be sent on the first attempt, and have remained in the server's time-out system waiting to be sent and receive confirmation.

TABLE I. Results

Agent Type	Number of deployed agents	Number of down agents		Number of resend messages	
		Test 1	*Test 2*	*Test 1*	*Test 2*
.NET	210	2	4	21	18

Agent Type	Number of deployed agents	Number of down agents		Number of resend messages	
		Test 1	*Test 2*	*Test 1*	*Test 2*
C#	200	3	3	24	20
Objective C	2	0	0	2	5
Phyton	140	1	0	13	15
Java	180	1	1	18	11
Java Android	4	0	0	3	4

The evolution of the number of messages that the server transmitted during the first test can be seen in Figure 4. For each hour, the average number of messages is shown.

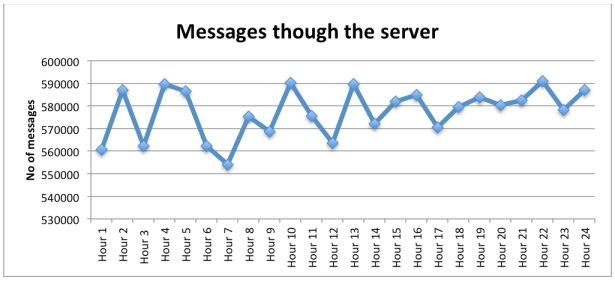

Figure 4. Message though the server

We can conclude that the PANGEA agent platform that we are developing has great potential to create open systems, and more specifically, virtual agent organizations. This platform includes various tools that make it easy for the end user to create, manage and control these systems. One of the greatest advantages of this system is the communication platform that, by using the IRC standard, offers a robust and widely tested system that can handle a large number of connections and ensure scalability.

ACKNOWLEDGMENT

This work has been partially supported by the MICINN project TIN 2009-13839-C03-03

REFERENCES

[1] Oikarinen, J. and D. Reed, "Internet Relay Chat Protocol", RFC 1459, May 1993.

[2] Kalt, C., "Internet Relay Chat: Client Protocol", RFC 2812, April 2000.

[3] Kalt, C., "Internet Relay Chat: Server Protocol", RFC 2813, April 2000.

[4] Kalt, C., "Internet Relay Chat: Channel Management", RFC 2811, April 2000.

[5] Kalt, C., "Internet Relay Chat: Architecture", RFC 2811, April 2000.

[6] Ferber, O. Gutknecht, F. Michel. 2004. From Agents to Organizations: an Organizational View of Multi-Agent Systems, in: P. Giorgini, J. Muller, J. Odell (Eds.), Agent-Oriented Software Engineering VI, Vol. LNCS 2935 of Lecture Notes in Computer Science, Springer-Verlag: 214–230.

[7] Foster, I., Kesselman, C. and Tuecke, S. 2001. The anatomy of the grid: Enabling scalable virtual organizations, Int. J. High Perform. Comput. Appl, vol. 15, no.~3: 200--222

[8] O'Brien, P. D. and Nicol, R. C. FIPA , Towards a Standard for Software Agents. BT Technology Journal, vol. 13, issue 3, pp. 51-59. Springer Netherlands, 1998.

[9] Emorphia, FIPA-OS. http://fipa-os.sourceforge.net/

[10] Berna-Koes, M.; Nourbakhsh, I.; Sycara, K. "Communication efficiency in multi-agent systems," Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on , vol.3, no., pp. 2129-2134 Vol.3, 26 April-1 May 2004 doi: 10.1109/ROBOT.2004.1307377

[11] Finin, T., Fritzson, R. and McEntire, R. KQML as an agent communication language. In Proceedings of the 3rd International Conference on Information and Knowledge Management, November 1994.

[12] Dale, J., Knottenbelt, J. and Fujisu Labo. April Agent Platform. http://designstudio.lookin.at/research/relate%20survey/Survey%20Agent%20Platform/April%20Agent%20Platform.htm (accessed 02 01 2012)

[13] (1997) Foundation for intelligent physical agents. [Online]. Available: http://www.fipa.org.

[14] SACI – Simple Agent Communication Infrastructure. http://www.lti.pcs.usp.br/saci/

[15] Braubach, L., Pokahr, A. and Lamersdorf, W. Jadex: A Short Overview. In Procceding Main Conference Net.ObjectDays 2004, pp. 195-207, 2004.

[16] O. Gutknecht and J. Ferber. MadKit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical Report R.R.LIRMM 9718, LIRM, December 1997.

[17] Busetta, P., Rönnquist, R., Hodgson, A., and Lucas, A. (1998). JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia

[18] Jomi Fred Hubner, Jaime Simao Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In Guilherme Bittencourt and Geber L. Ramalho, editors, Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), volume 2507 of LNAI, pages 118-128, Berlin, 2002. Springer.

[19] Hübner, J.F., Bordini, R.H., Picard, G.: Using Jason and MOISE+ to develop a team of cowboys. In: Hindriks, K., Pokahr, A., Sardina, S. (eds.) Proceedings of the Seventh International Workshop on Programming Multi-Agent Systems (ProMAS 08), Agent Contest, held with The Seventh International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2008), LNAI, vol. 5442, pp. 238–242. Springer, Heidelberg (2009)

[20] Gutknecht O. and Ferber, J. MadKit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical Report R.R.LIRMM 9718, LIRM, December 1997.

[21] Giret, A. An open architecture for Service-Oriented Virtual Organizations. Programming Multi-Agent Systems: 7th International Workshop, ProMAS 2009

[22] Hübner, J.F. J -Moise+ Programming organisational agents with Moise+ & Jason. Technical Fora Group at EUMAS'07.

[23] Agent Oriented Software Pty Ltd. JACK™ Intelligent Agents Teams Manual. s.l. : Agent Oriented Software Pty Ltd, 2005

[24] Helsinger, A.; Thome, M.; Wright, T. "Cougaar: a scalable, distributed multi-agent architecture," Systems, Man and Cybernetics, 2004 IEEE International Conference on , vol.2, no., pp. 1910- 1917 vol.2, 10-13 Oct. 2004 doi: 10.1109/ICSMC.2004.1399959

[25] Helsinger, A.; Wright, T.; , "Cougaar: A Robust Configurable Multi Agent Platform," Aerospace Conference, 2005 IEEE , vol., no., pp.1-10, 5-12 March 2005 doi: 10.1109/AERO.2005.1559614

[26] Gruver, W., "Technologies and Applications of Distributed Intelligent Systems", IEEE MTTChapter Presentation, Waterloo, Canada, 2004

[27] Bordini, R. H., Hübner, J. F., and Vieira, R. 2005. Jason and the Golden Fleece of agent-oriented programming. In Bordini, R. H., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., eds., Multi-Agent Programming: Languages, Platforms and Applications. Springer-Verlag. chapter 1, 3-37.

[28] Bordini, R. H., Hübner, J. F., and Wooldridge, M. 2007. Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd.