Analysis of Jade and JATLite Frameworks in Web Domains

David Camacho, Ricardo Aler, Javier Hernando, and José M. Molina

Universidad Carlos III de Madrid, Computer Science Department, Avenida de la Universidad n° 30, CP 28911, Leganés, Madrid, Spain
dcamacho@ia.uc3m.es

Abstract. Nowadays, there is an intensive research in different computer science fields that are involved in the development of intelligent systems that should work with the information stored in the World Wide Web. Specially, and due to the characteristics of this domain, the Intelligent Software Agent field is very active. This has produced a lot of different tools, frameworks, reusable libraries, etc ... that can be used in a free way to implement agent-based systems that will be applied over different kind of domains. The previous situation has originated several problems that could be summarized as: what are the agent technology most appropriate to build an Internet Agent or Multi-Agent based system?, which technology has the best scalability when the number of agents grows?, what are the main technological characteristics of each tool?, what kind of computer-aided design and development tools different frameworks provide?,... Some of the previous questions will be analyzed in this paper. Due to the number of agent-based tools, we will analyze two of those tools that are widely used by different researchers in the fields of Software Agents and Multi-Agent Systems research. This tools will be JADE and JATLITE. The paper will show an empirical evaluation of these tools, specially in the software reutilization, and in the scalability problems.

1 Introduction

In recent years, there has been a lot of work in Web-based information technologies, a fast development of Internet, and many posibilities of managing the stored knowledge. Because of this, Intelligent agents or Multi-Agent Systems have become an important focus of interest [10]. Those research fields apply different techniques to implement adaptive and autonomous systems whose main goal is to obtain, reason and manage the knowledge stored in the Web. There is a wide range of posibilities to achieve the previous goals, from systems that gather information from different electronic information sources and then other agents integrate it into a possible solution [2, 6, 8], to systems that provide a customizable answer to the user using his/her learned preferences [7].

Due to the increasing complexity of Internet, the related complexity of the specialized systems that work retrieving and managing information from this

source are increasing too. This generates several problems, that could be summarized as:

- It is more difficult to build specialized Web-based agents.
- It is more complex to mantain and to reuse the deployed agents, and the whole Multi-Agent System.
- When anything changes in the electronic sources (format, knowledge representation or access information methods: CGI, servlets, asp...) the agents need to adapt to these changes.

To solve, or to smooth, previous problems it have been developed tools, or frameworks, that allow engineers to:

- Design the roles and functionalities for each agent.
- Define the ontologies, or knowledge that will be shared, between them.
- Define and implement the communication process.
- Help to implement the designed agents, and the whole Multi-Agent System.
- Reuse software libraries to facilitate the software development phase.
- Test and debug the implemented system.

The previous points have been achieved by different frameworks providing to engineers programming facilities like visual programming toolkits, documentation, reusable software libraries, etc... Actually it is possible to find about half a hundred of those tools[1]. There is a wide range of possibilities from commercial products like: AgentBuilder, IBM Aglets Workbench, Grasshopper, JACK Intelligent Agents,... to research products like: Jade, JATLite, MadKit, Zeus,... When any designer needs to build his/her own Multi-Agent System, it would be useful to have guidelines to select the most appropriate, to do that several characteristics should be measured, like software reusability or computational performance for the implemented agents. The main goal of this paper is to evaluate the performance of two of those frameworks in a specific domain.

The paper is structured as follows. Section 2 describes the main characteristics of the two analyzed frameworks: JADE and JATLITE. Section 3 explains in detail a Multi-Agent System that will be implemented with the two analyzed frameworks and will be used to evaluate important aspects of both architectures. Section 4 evaluates empirically the performance of both tools. Finally, Section 5 summarizes the conclusions and future lines of work.

2 Multi-Agents Frameworks Description

¿From all the possible frameworks we have selected two popular tools used by different engineers and research centers to develop its agent-based applications:

[1] http://www.multiagent.com/index.html

1. JADE (**J**ava **A**gent **DE**velopment tool): Developed by Multimedia Technologies and servicies of Cselt (http://sharon.cselt.it/projects/jade).
2. JATLite (**J**ava **A**gent **T**emplate, **Lite**): Developed by Stanford Center for Design Research (http://java.stanford.edu/).

Common characteristics could be found for both frameworks like: they use Java as main programming language to provide portability to the implemented agents; both frameworks provide a set of packages of reusable java classes to aid and to facilitate the implementation of the software agents; and finally, the frameworks provide to engineers some graphical user utilities that will help to build and test the Multi-Agent System.

2.1 Multi-Agent Framework: Jade

JADE (Java Agent DEvelopment framework) is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications [4] and through a set of tools that supports the debugging and deployment phase [1]. JADE agent platform tries to keep high the performance of a distributed agent system implemented with the Java language. In particular, its communication architecture tries to offer flexible and efficient messaging, transparently choosing the best transport available and leveraging state-of-the-art distributed object technology embedded within Java runtime environment. JADE uses an agent model and a Java implementation that offer a good runtime efficiency and software reuse. This agent model is more "primitive" than the agent models offered by other systems, but such models can be implemented on the top of our " primitive" agent model [1]. This framework is built using the combination of two main products: a FIPA-compliant agent platform and a package to develop Java agents.

Any application implemented using Jade uses the *platform* concept, all the agents are run inside a set of *containers* that provides the communication between them. The *"Jade Agent Platform"* complies wiht FIPA97 specifications and includes all those mandatories agents that manage the platform, that is:

- The Agent Management System (**AMS**), is the agent that exerts supervisory control over access to and used of the platform (it is responsible for authentication of resident agents and control of registrations).
- The Agent Communication Channel (**ACC**) is the agent that provides the path for basic contact between agents inside and outside the platform (it is the default communication method).
- The Directory Facilitator (**DF**) is the agent that provides a yellow page service to the agent platform.

All of those agents are automatically activated at the agent platform start-up. Figure 1 shows the software architecture of one Jade agent platform. These agents platforms are distributed and can be split on several hosts. Any software agent in Jade is implemented as one Java thread. The agent platform

provides a Graphical User Interface (GUI) for the remote management, monitoring and controlling status of the agents. This GUI has been implemented as an agent, called RMA (Remote Monitoring Agent). The agent communication is performed through message passing, where FIPA-ACL is the language to represent messages.

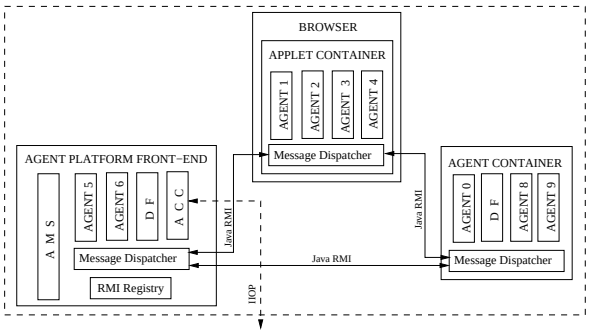

Fig. 1. Jade Agent Platform Software Architecture[1].

The architecture of a Jade Agent platform is built using a set of *"agent containers"*, each agent container is an RMI server object[2] that locally manages a set of agents. It controls the life cycle of agents by creating, suspending, resuming and killing them. Several agent special containers are implemented for each platform:

- the *front-end role container*: running management agents and representing the whole platform to the outside world.
- a complete *agent platform* (AP): composed of several agent container.
- a special *light-weigth* container (Browser): which allows the execution of agents within a Web Browser.

When this tool is selected to build the Multi-Agent System, it has both advantages and disadvantages that could be summarized as:

- Jade does not have a powerful programming environment, this framework only provides to the user a set of interfaces that allow him to debug the implemented agents.
- One of the best characteristics in Jade is that it has an excellent documentation, a good API to reuse the provided libraries to build new agents.
- Using Jade, a set of communication libraries (or packages) is provided to the software engineers, those libraries allow them to isolate the communication problem.

[2] RMI or Remote Method Invocation is the Distributed Object Model provides by Java language.

2.2 Multi-Agent Framework: JATLite

JATLite (Java Agent Template, Lite) is a tool for creating agent-based systems. JATLite is a package of programs written in Java language that allow users to quickly create new **software agents** that communicate robustly over the Internet [5, 9]. This framework includes a message router that supports name and password mechanism that lets agents move freely between hosts. JATLite also provides a basic infrastructure, shown in Figure 2, in which agents register with an Agent Message Router (AMR) using a name and a password.

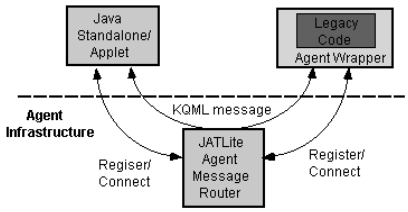

Fig. 2. JATLIite Agent Message Router and agent communication infrastructure[5].

This architecture allows the agents to connect or disconnet from the Internet, send and receive messages, transfer files (with the FTP protocol) and generally exchange information with other agents running in different computers. The JATLite Agent Message Router is a specialized application which receives a message from the registered agents and routes to the correct receiver.

JATLite provides a *"template"* for building agents that utilize a common high-level language (KQML [3]) and protocol. This template provides the user with a set of predefined Java classes that facilitate agent construction. Those classes are provided in *layers* (Protocol Layer, Router Layer, KQML Layer, Base Layer, Abstract Layer)[3], so the developer can decide what classes are needed for the agent. General operations assumptions for a JATLite agent are:

- TCP/IP based connections.
- Agent communication through message passing, using the standard KQML language.
- One live connection for each connected agent.
- A single address should be assigned to each agent.

3 SIMPLENEWS: A Meta-Searh News Engine

The aim of this section is to present the description of a MAS, called SIMPLENEWS. This MAS will be implemented using the two frameworks so that they

[3] These layers are the JATLite API.

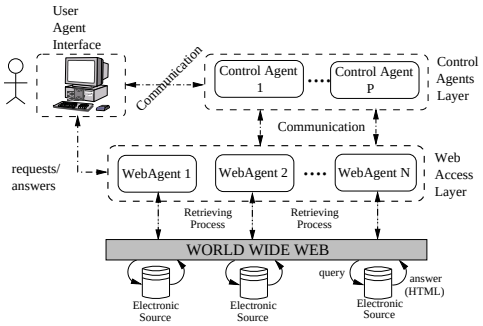

Fig. 3. SIMPLENEWS Architecture.

can be compared. It is important to remark that the goal of this system is not to build a MAS that works with several WEB sources. The main goal, is to implement a MAS that can work in serveral WEB domains for different Multi-Agent toolkits.

SIMPLENEWS is a meta-search engine that allows, by means of a UserAgent, to search for news in a set of electronic newspapers. In this paper a very simple topology was used (see Figure 3), where all of the Web agents solve the queries sent by the UserAgent. The SIMPLENEWS engine is built using a set of specialized agents that are able to retrieve information from a particular electronic newspaper. SIMPLENEWS can retrieve from the selected electronic sources, filter the different answers from the specialized agents and show them to the user. As Figure 3 shows, SIMPLENEWS architecture can be structured in serveral interconnected layers:

- *UserAgent Interface.* This agent only provides a simple Graphical User Interfaces to allow users to requests for news from the selected electronic papers. SIMPLENEWS uses a UserAgent that provides to the users a simple graphical user interface for making queries. The number of solutions requested, and the agents that will be consulted. The interface used by this agent allows to the user to know: The actual state of the agents (active, suspended, searching, finished), and the messages and contents sent between the agents. Finally the entire requests retrieved by the agents are analyzed (only different requests are taken into account) and the UserAgent builds an HTML file, which is subsequently displayed to the user.
- *Control Access Layer.* Jade, JATLite (or any Multi-Agent architecture) need to use specific agents to manage, running or controling the whole system (*AMS, ACC ,DF* in Jade or *AMR* in JATLite), this level represents the set of necessary agents (for the architecture analyzed) that will be used by SIMPLENEWS to work correctly. This layer performs the differences from the two versions of SIMPLENEWS implemented (from Jade framework and from JATLite).
- *Web Access Layer.* Finally, this layer represents the specialized WebAgents which retrieve information from the specific electronic sources in the Web.

The meta-search engine includes a UserAgent and six specialized WebAgents. The Web specialized agent can be classified in the next categories:

1. Financial Information. Two WebAgents was implemented and they will specialized in *Expansion* [4] and *CincoDias* [5] financial newspapers.
2. Sports information. Other two WebAgents specialized in *Marca* [6] and *Futvol.com* [7] sportive papers.
3. General information. Finally other two WebAgents was implemented to retrieve information from; *El Pais* [8] and *El Mundo* [9] newspapers.

The selected electronic sources are Spanish to allow a better evaluation in the retrieving process, because is difficult to evaluate the performance of a particular WebAgent if it works using a query in a different language. Other reason to select previous sources was that most of those sources are widely used in Spain, so the information stored in them should be enough to test a MAS built with those WebAgents.

4 Experimental Evaluation

In this section, we report the results of several experiments, using both versions of SIMPLENEWS to evaluate the *scalability* and the *performance* (in time response) for each framework when the number of those agents increases.

4.1 Experimental Setup

Various empirical tests to evaluate the general performance of the MAS have been implemented. A set of 432 queries to each architecture were made, so 1728 requests were made to obtain the empirical results shown in the next section. The following variables were measured:

- Independent variables:
  - Number of Web agents used: from only one Web agent, to six specialized Web agents.
  - Number of requested documents: from only one document to fifty documents: 1, 5, 10, 15, 20, 30, 40 and 50 news articles requested.
  - The user query that will be searched by the agents.
- Dependent (measured) variables:
  - Response time that the UserAgent spent to answer the question.
  - Number of news articles retrieved.

[4] *http://www.expansion.es*
[5] *http://www.cincodias.es*
[6] *http://www.marca.es*
[7] *http://www.futvol.com*
[8] *http://www.elpais.es*
[9] *http://www.elmundo.es*

The same questions were made to each configuration. The following tests were made, using the following configurations in SIMPLENEWS:

- Only one Web agent (better Web agent in the retrieving news process).
- Two Web agents specialized in different electronic newspapers. It was used an agent specialized in general information (*ElPais-WebAgent*), and other agent specialized in financial information (*Expansion-WebAgent*).
- Three Web agents: *ElPais-WebAgent, Expansion-WebAgent, Marca-WebAgent* (the best specialized agent in the different information sources).
- Finally, all the Web agents developed will be used to measure the performance of SIMPLENEWS.

4.2 Scalability and Performance Evaluation in Jade and JATLite

The experiments in this section display the average time to answer questions and the number of articles retrieved for each architecture.

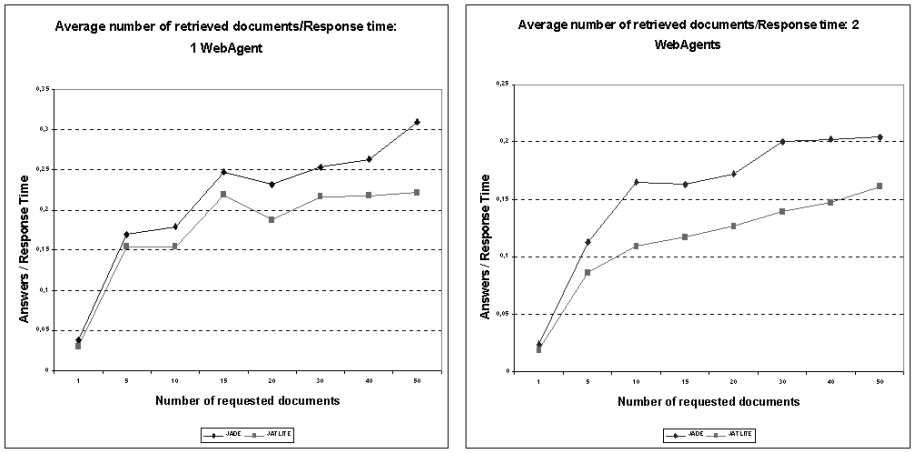

Fig. 4. Average number of retrieved documents/ response time for each architecture using: (a) One and (b) Two specialized Web agents.

Figure 4 (a) shows the performance for each architecture when the best Web agent (in the searching and retrieving process) is used. It can be seen how Jade has a good performance. Therefore, only two agents are considered in the system, *ElPais-WebAgent* and the UserAgent (we are not considering the *Control Agents Layer* in the System). When two WebAgents (*ElPais-WebAgent*): general information, and *Expansion-WebAgent*: financial information) were used, the performance of Jade SIMPLENEWS version is still better, Figure 4 (b) shows the reply time obtained for the different questions asked by the user.

The empirical results shown in Figure 5 (a) were obtained using *ElPais-WebAgent, Expansion-WebAgent*,and *Marca-WebAgent*. Figure 5 (a) shows how

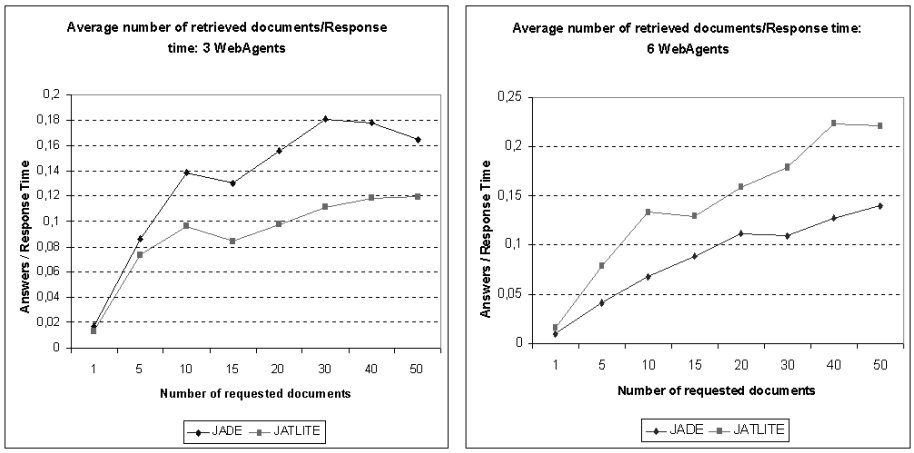

Fig. 5. Average number of retrieved documents/ response time for each architecture using: (a) Three and (b) Six specialized Web agents.

the behavior for the JATLite and JADE architectures are still very linear, and how the best performance is achieved again by the Jade architecture. However, Figure 5 (b) shows the obtained performance when all the possible WebAgents are used, and how, in this case, JATLite MAS has a better performance with this configurations of SimpleNews .

Finally, it can be observed a linear behavior for the two versions of SimpleNews implemented, and the performance is similar. The Jade architecture has a good performance, but when the number of agents increases to seven specialized agents (one UserAgent and six WebAgents) the performance of the MAS is 9.7% under the peformance of JATLite architecture (in the worst situation for 40 news articles requested).

5 Conclusions and Future Work

Although there are many different frameworks to build multi-agent systems, not much effort has been spent in comparing them empirically. In this paper, we present a simple Web domain which is used as a common domain to test two different MAS frameworks: Jade and JATLite. In summary, our conclusions are:

- The Jade architecture allows a quicker implementation of MAS and provides excellent documentation if simple agents are implemented (this framework provides a good API to help to programming software agents). However the underlying technologies (RMI, IIOP, etc...) are difficult to understand and to use.
- JATLite has a similar performance to Jade, however, currently development is slower because it has a poor documentation and development support tools.

– The two frameworks scalate well, but when the number of agents grows, Jade complex architecture penalizes the performance of the whole multi-agent system (shown in the last set of experiments).

In the future, we will use the SIMPLENEWS domain to compare other architectures [10], and will increase the number of agents to check scalability. We encourage other researchers to test their architectures in this domain to have comparative results in a real task.

Acknowledgements

The research reported here was carried out as part of the research project funded by CICYT TAP-99-0535-C02.

References

1. Bellifemine, F., Poggi, A., Rimassa, G. JADE-A FIPA-Compliant agent framework. Proceedings of the 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-99). pp. 97–108, 1999.
2. Fan, Y., Gauch, S.:Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources. Proceedings of 1999 AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University, March 1999.
3. Finin, T., Fritzson, R., Mackay, D., McEntire, R.: KQML as an Agent Communication Language. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94), pages 456–463. New York: Association of Computing Machinery (1994).
4. Foundation for Intelligent Agents (FIPA). Specifications 1998. Available from http://www.fipa.org/specifications.
5. Jeon H., Petrie, C., Cutkosky, K.R. JATLite: A Java Agent Infrastructure with Message Routing. IEEE Internet Computing. pp. 87–96, March/April 2000.
6. Lambrecht, E., Kambhampati, S.: Planning for Information Gathering: A tutorial Survey. ASU CSE Techincal Report 96-017. May (1997).
7. Lieberman, H.: Letizia: An Agent That Assists Web Browsing. International Joint Conference on Artificial Intelligence (IJCAI95), August (1995) 924–929.
8. Knoblock, C.A., Minton, S., Ambite, J.L., Muslea, M., Oh, J., Frank, M.: Mixed-Initiative, Multi-source Information Assistants. The Tenth International World Wide Web Conference (WWW10). ACM Press. May 1-5. (2001).
9. Petrie, C. Agent-based Engineering, the Web, and Intelligence. IEEE Expert, 11 (6), pp. 24–29, December, 1996. Also available online at http://cdr.stanford.edu/NextLink/Expert.html.
10. Sycara, K., Decker, K. Distributed Intelligent Agents. IEEE Expert. 11(6), pp. 36–46, December, 1996.

[10] We have already started evaluating the ZEUS and SkeletonAgent frameworks.