SOFTWARE AND PERFORMANCE MEASURES FOR EVALUATING MULTI-AGENT FRAMEWORKS 

DAVID CAMACHO and RICARDO ALER

Universidad Carlos III de Madrid, Computer Science Department, Avenida de la Universidad nº 30, 28911, Leganés (Madrid), Spain

email: dcamacho@ia.uc3m.es, aler@inf.uc3m.es

Abstract. *Nowadays, multi-agent frameworks allow to implement very complex systems by means of agent technology. However, this complexity makes more difficult to evaluate software and runtime characteristics of Multi-Agent Systems (MAS). Our aim is to define and study some quantitative measures to measure MAS aspects like development time, reusability, scalability, etc. These measures could be used by engineers to guide the selection among several MAS frameworks. Our study has been carried out in several MAS frameworks like* JADE, JATLITE, SKELETONAGENT, *and* ZEUS, *which have been used to build a MAS application for news retrieval.*

1 Introduction

The evolution of the Web, the Semantic Web [Berners-Lee *et al.*2001], and new computing environments like the Grid have made available a huge amount of

data and the possibility to use multiple interconnected computers to solve a problem A popular approach to build distributed systems is the agent-based oriented programming approach, taking advantage of characteristics of agents, such as robustness, adaptability, scalability, and autonomy Currently, there are many frameworks that help engineers to build multiagent systems However, there is little guidance to choose among them because of the lack of empirical evaluations

The main contribution of this paper is to empirically study several software and performance measures for MAS (and MAS frameworks) These measures are development time and reusability in the MAS framework, and performance and scalability of the MAS itself Development time and reusability measure how easy it is to build a MAS in a particular framework We have studied performance with respect to the complexity of the task given to the agents In order to focus our empirical study, we have used several MAS frameworks (JADE, JATLITE, SKELETONAGENT, and ZEUS) to build a particular application (news retrieval), which has been subsequently tested We believe that the results obtained in this study will be useful to other researchers that use these frameworks to build other applications

The paper is structured as follows Section 2 describes the software and performance characteristics that will be considered to obtain an evaluation for a particular Multi-Agent System Section 3 describes briefly the main characteristics of MAS frameworks we have used Section 4 explains in detail the Multi-Agent System that has been implemented with these previous frameworks and

that will be used to evaluate the characteristics described in Section 2 Section 5 provides the empirical evaluation of the implemented MAS Finally, Section 6 summarizes the conclusions of the paper

2 Analysed MAS Characteristics

The purpose of this section is to define several characteristics that will be used to evaluate empirically the behaviour of a particular agent-based system, and the related framework used to build it Two different characteristics have been taken into account On the one hand, the software building characteristics - development time and reusability- of the MAS framework, measure how easy it is to build an application within the framework They also depend on programmer's experience On the other hand, the runtime characteristics, performance and scalability, measure how well a MAS reacts to increased task complexity and increased number of agents, respectively Our purpose is to study system performance degradation with respect to these parameters

2.1 Software building characteristics

They are related with the software engineering phases and with the tools, or frameworks, used to implement the system Therefore, these parameters evaluate the implementation effort required from the designers and programmers The selected characteristics are:

- **Development Time** This parameter includes both, the design and implementation effort, for the development of the first agent, and for the whole

MAS For the First Agent, the next phases will be considered (and measured):

- *Analysis and Design* Several characteristics are measured in this phase, like the number of hours used by the engineers to analyze and design the desired MAS for each technology This phase includes the time necessary to analyse, and understand correctly the documentation provided by the frameworks The time necessary to analyze other framework facilities like programming API, or CAD tools are included too
- *Code generation for the first agent* It measures the time needed by engineers to implement the new agents characteristics, or roles, that have been defined in the previous phase
- *Integration with external code* It evaluates the effort to integrate the previous code (that implements a specific ability) in the agent Therefore, the time required to modify the (generic) agents provided by the frameworks to build the specialized agents is measured
- *Tests and debugging* Time necessary to test and debug the problems that could appear until the system works correctly (using the minimum set of agents necessary to work)
- For the rest of the agents, the time to achieve the next phases will be considered:
  - To generate the code (specific skills or abilities) for the new agents
  - To integrate, test, and debug the new code

- **Software Reusability** Three software parameters will be measured The *LOC* (lines of code), the *new Java classes* and the *number of reused classes* that are necessary to build the new agents in the system These characteristics measure the programming effort for the technologies evaluated

2.2 Runtime characteristics

Once a Multi-Agent system is fully deployed, several experimental tests can be defined to measure important characteristics like robustness or scalability in the system This subsection analyzes the selected characteristics to measure the behavior of any MAS

- **Performance evaluation** This performance evaluation is used to test a simple MAS with the minimum set of agents that allow the system to work correctly The following parameters will be measured for each framework:
  - *Response (request) time* This is the time the user has to wait until the MAS has finished its task This depends on the complexity of the task given to the agents
  - *Number of Messages* The number of exchanged messages between the agents during the run This is an important characteristic in MAS, because the behavior the whole system could be affected if the number of messages grows quickly, or if these messages are not correctly managed
- **Scalability evaluation** The number of agents involved in the implemented MAS will be increased to test how scalable is the system when a particular MAS framework is used From the same experimental sets used in previous

performance evaluation, the number of agents will be increased Only the response time will be measured to obtain a quantitative measure about the scalability of the system

3 Multi-Agent Frameworks Description

Nowadays, there are many MAS frameworks that help in all the different design and implementation phases of this type of systems Several design decisions, like the agent architecture, the communication technology used, or how to define the behavior of those agents, will finally affect the behavior of the whole system

This section describes briefly the main characteristics of several MAS frameworks All of them provide the programmer with Java classes to be reused and also have some graphical capabilities

3.1 Jade

JADE (Java Agent DEvelopment framework) [Bellifemine *et al.*2001], [Bellifemine *et al.*1999], developed by Multimedia Technologies and services of Cselt (http://sharon cselt it/ projects/ jade), is a software framework that simplifies the implementation of MAS through a middle-ware that complies with the **FIPA** specifications (http://www fipa org/ specifications/ index html) The JADE agent platform tries to keep high the performance of a distributed agent system implemented with the Java language 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"* includes next mandatories agents (to be FIPA compliant) 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

The architecture of a Jade Agent platform is built using a set of *"agent containers"*, each agent container is an RMI server object that locally manages a set of agents It controls the life cycle of agents by creating, suspending, resuming and killing them When this tool is selected to build the MAS, 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 for debugging
- One of the best characteristics in Jade is that it has an excellent documentation, and a good API to reuse the libraries

3.2 JATLite

JATLite [Jeon *et al.*2000,Petrie1996] (Java Agent Template Lite: java stanford edu/ java_agent/html), developed by Stanford Center for Design Research (http://

java stanford edu/), is a tool for creating agent-based systems JATLite is designed to allow users to quickly create new software agents that can communicate robustly over the Internet 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 in which agents register with an Agent Message Router (AMR) using a name and a password JATLite provides a *"template"* for building agents that utilize a common high-level language (KQML [Finin *et al.*1994]) 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) These layers build the JATLite API, 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; and finally, a single address should be assigned to each agent

3.3 SkeletonAgent

SkeletonAgent [Camacho *et al.*2005], developed by the Systems Complex and Adaptive Laboratory (http:// scalab uc3m es/agente), is a software framework toolkit that tries to wrap the "agent concept" into a set of reusable libraries The architecture of SkeletonAgent has been designed to allow the integration, and cooperation, of classical AI techniques like planning, or machine learning,

into distributed and heterogeneous agent-based systems SkeletonAgent requires a set of predefined agents:

- Control agents: the SkeletonAgent architecture is based in the "team" concept All the agents in the system belong to one team, which is managed by a particular *CoachAgent* All of the CoachAgents are managed by a *Manager-Agent* These agents manage different problems, like the insertion or deletion of them in the system
- Execution agents: these agents are involved in solving the task It is possible to define different kinds of agents, like UserAgents (that deals with the users), WebAgents (specialized in retrieving information from the Web), Learning agents, or Planner Agents

The main disadvantages of this framework coud be summarized as:

- The framework does not have a graphical programming interfaces, and the debugging of the agents is a hard task
- There is a poor documentation about the reusable libraries

However, the concept of agent is well encapsulated and it is possible to reuse a lot of code The inter-agent communication in Skeleton implements a reduced version of KQML This language provides to the system agents flexible protocols that allows them to coordinate and to share information and tasks in a cooperative way

3.4 The ZEUS Agent Building Toolkit

The main goal of ZEUS Toolkit [Azarami and Thompson2000,Collis *et al.*1998], developed by BT Laboratories in the Advanced Applications & Technology Department (http://www labs bt com/ projects/ agents/ zeus/ index htm), is to facilitate the rapid development of new multi-agent applications by abstracting into a toolkit the common principles and components underlying some existing multi-agent systems Different assumptions are made when any software agent is build using ZEUS, this assumptions trying to facilitate, and also to describe the typical application domains of these agents ZEUS makes several assumptions about the agents to be built The main assumptions made regarding the agent behavior are that the agents are: deliberative, goal-directed and rational; always truthful when dealing with other agents; versatile, i e they can have may goals and can engage in a variety of tasks; and finally, temporally continuous The MAS architecture defined by ZEUS uses the following type of agents:

- Agent Name Server (ANS) Any agent in the system can request to the ANS for the address of other agents
- Facilitator Agent These kind of agents receive and reply to queries from agents about the abilities of other agents They work by periodically querying all the agents in the society about their abilities and storing the returned information in their Acquaintance Database
- Generic Agents These agents perform different tasks and allow the system to achieve the designed goals

The inter-agent communication in ZEUS language is used to communicate with the Agent Name Server, the Facilitator and other agents The communication requires a shared representation and understanding of common domain concepts, i e , a common ontology (ZEUS provides tools to create new ontologies) The communication protocol used by the agents is TCP/IP and the language supported is FIPA-ACL

4 SIMPLENEWS: A News Meta-Search Engine

This section describes the architecture of SIMPLENEWS [Camacho *et al.*2002] SIMPLENEWS is a MAS implemented using several specialized agents that are able to retrieve information from several Web sites This system has been implemented using everyone of the frameworks described in the previous section, and it will be used as a case study for the proposed software and performance parameters of Section 2

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 simple topology has been used (see Figure 1), where all of the Web agents solve the queries sent by the UserAgent These agents can retrieve from the selected electronic sources the requested news, filter the different answers from the Web sites, and show them to the user As Figure 1 shows, SIMPLENEWS architecture can be structured in several layers:

– *UserAgent Interface* This agent only provides a simple Graphical User Interfaces to allow users to request for news from the selected electronic papers

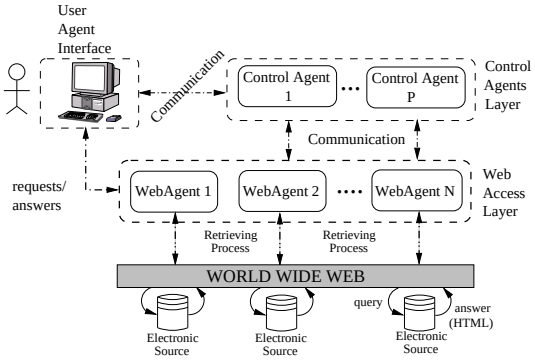

Fig. 1. SIMPLENEWS Architecture.

The interface used by this agent allows to the user to know: The current state of the agents (active, suspended, searching, finished) and the messages them

- *Control Access Layer* Jade, JATLite, SkeletonAgent, or ZEUS, need to use specific agents to manage, running or controlling the whole system (i e *AMS, ACC ,DF* in Jade; *AMR* in JATLite; *ANServer, Facilitator, Agent Viewer, Society Viewer,...* agents in ZEUS; or Manager and CoachAgent in SkeletonAgent), this level represents the set of necessary agents (for each architecture analyzed) that will be used by SIMPLENEWS to work correctly
- *Web Access Layer* This layer represents the specialized WebAgents which retrieve information from the specific electronic sources in the Web The main differences in this layer will be performed by the particular agent (internal) structure provided by each MAS framework

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

1 Financial Information: specialized agents for the financial newspapers *Expansion* (http:// www expansion es) and *CincoDias* (http://www cincodias es)

2 Sports information: *Marca* (http://www marca es) and *Futvol.com* (http://www futvol com) sport newspapers

3 General information: *El Pais* (http://www elpais es) and *El Mundo* (http://www elmundo es) newspapers

5 Experimental Evaluation

This section evaluates the four MAS frameworks (JADE version 2 5, JATLite version 0 4, and ZEUS version 1 0 3 will be used) using the measures described in Section 2 On the one hand, we show the development time and reusability for implementing SIMPLENEWS with each of the frameworks On the other hand, we evaluate the performance and scalability of the resulting MAS's

5.1 Experimental Results: Software Characteristics

Table 1 provides the evaluation for the software characteristics Parameters: $Analysis\&Design$ to $CodeIntegration(MAS)$ are expressed in hours, whereas parameters $LinesofCode$ to $N^{o}ReusedClasses$ are expressed in lines of code, and number of implemented/reused classes respectively Some initial conclusions can be drawn from Table 1, SkeletonAgent obtains the maximum values for most of the evaluated characteristics On the other hand, JADE and ZEUS get the best values (in bold)

Table 1. Experimental results for qualitative parameters.

Parameter	JADE	JATLite	Skeleton	ZEUS
$Analysis\&Design$	28	25	**10**	63
$CodeGeneration(1^{st}agent)$	**12.5**	25	30	**12.5**
$CodeIntegration(1^{st}agent)$	12.5	15	30	14.5
$Test\&Debug$	**6**	10	15	7
$CodeGeneration(MAS)$	**2.5**	3	3.5	2
$CodeIntegration(MAS)$	**2**	**2**	2.5	**2**
$LinesOfCode$	1897	3199	1447.5	**1354**
$NewJavaClasses$	**9**	11	20	13
$N^{o}ReusedClasses$	**11**	13	14	**11**

5.2 Experimental Results: Runtime Characteristics

Two different experiments will be carried out to evaluate the runtime characteristics in SIMPLENEWS On the one hand, to evaluate the **performance** for each version of the system ($RequestTime$ and $N^{o}Messages$ paremeters), a set of 50 queries to each architecture will be made (modifying the number of requested news, from only one document to fifty documents), and the following variables will be measured:

- Number of Web agents used: from only one Web agent, to six specialized Web agents
- Response time that the UserAgent spent to answer the query ($RequestTime$ quantitative parameter)

– Number of messages exchanged among the agents ($N^{a}Messages$ quantitative parameter)

The same queries will be made for every configuration The following tests have been made, using the following configurations in SIMPLENEWS:

– Only one Web agent (the best news retriever)
– Two Web agents specialized in different electronic newspapers It has been 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)
– All the Web agents

Table 2 shows the results for both parameters As in the previous section, this table shows two values: [current value/normalized value] for each evaluated parameter The normalization of these parameters have been made as dependent parameters Bold results appear in bold

Some conclusions can be drawn from Table 2 Minimum values correspond to the Jade and SkeletonAgent frameworks whose implementations of SIMPLENEWS obtain the better request time for the same experimental sets Both parameters, $RequestTime$ and $N^{o}Messages$, are highly related, because when any framework obtains the best evaluation in one of them, usually has the best evaluation in the other ones Only for the last configuration (6 Web agents) this is not true, although the values between Jade and SkeletonAgent frameworks are very closer

Table 2. Performance results for the runtime characteristics ($RequestTime/N^{o}Messages$).

N° agents	Jade	JATLite	SkeletonAgent	ZEUS
1	(**113.8/35.2**)	(225.1/38.9)	(136.3/35.7)	(146.7/39.3)
2	(378.7/77.3)	(468/75.4)	(**312.8/65.2**)	(383.6/74.8)
3	(659.6/108.4)	(946.3/113)	(**412.6/98**)	(716.7/144.2)
6	(1323.9/**185.1**)	(1093.1/253.3)	(**989.7**/190)	(1272.2/203.8)

On the other hand, the second experiment has been carried out to evaluate the **scalability** of the four versions of SIMPLENEWS In this experiment the same 50 queries have been used Also, only the best Web agent will be present, although it will be cloned up to 100 times for the scalability study The set of agents will be distributed in several computers (up to 8 Pentium 700MHz) Ten agents is the maximum number of (Web) agents that will be executed on a single computer In the last (and more complex) experiment, an average of fifteen (Web) agents per computer will be used, until one hundred of Web agents are working

In this experiment the same query is sent to all the agents, no selection of news is made by the user agent (because all the agents obtain the same results), therefore only the request time is measured It is shown in Table 3

In this experiment we can see how the best results are obtained by JADE, JATLite and SkeletonAgent However, the most important question is the problem about the scalability of those systems Only two frameworks, JADE, and SkeletonAgent, are able to execute the maximum number of agents JATLite

Table 3. Experimental results for the runtime characteristic: *RequestTime*.

N° agents	Jade	JATLite	SkeletonAgent	ZEUS
1	**113.8**	225.1	136.3	146.7
6	1323.9	1093.1	**989.7**	1272.2
10	**879.67**	1126	881.34	898.7
20	1607	**1544.7**	1558.2	1585.7
30	2603.7	1762.3	**1689.6**	1763.3
40	2379.23	2292.3	**2197.4**	–
50	1934.5	**1374.7**	1462.2	–
100	**110.3**	–	1201.4	–

obtains generally better results and only in the last experiment the AMR agents is not able to manage all the messages, this problem could be solved using more than one AMR agent because this is allowed in the architecture (to avoid the communication overload in one AMR agent) However, in this test only a simple configuration were used for all the frameworks Finally, the ZEUS framework is not able to execute more than thirty agents Also, the agent architecture provided by this framework, and specially some of the provided libraries (used to parse the contents of the messages), presents some problems when special characters are detected Usually, when the number of messages is increased, or special characters are retrieved from Web pages, the system goes down and this affects directly to the robustness of the ZEUS-based system

6 Conclusions

In this paper, we have tested several MAS frameworks (JADE, JATLITE, SKELETONAGENT, and ZEUS) by measuring software and performance characteristics in a news Web retrieval domain The software characteristics refer to the building of the MAS itself, and are related with the development time and reuse facilities provided by the MAS framework The performance (and scalability) characteristics measure how well the MAS reacts to increased task complexity and increased number of agents Our study highlights the differences between the frameworks used

Some frameworks like JATLite or SkeletonAgent have been designed and implemented to work in a particular type of domains (i e Internet), this characteristic could affect the results if the domain is changed Also, it would be interesting to test more domains, using the same measures, so that engineers can choose the right framework for their applications In the future we would like to extend this study to more frameworks and domains

References

[Azarami and Thompson2000] N. Azarami and S. Thompson. Zeus: A toolkit for building multi-agent systems. In *Proceedings of Fifth annual Embracing Complexity Conference*, April 2000.

[Bellifemine *et al.*1999] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade - a fipa-compliant agent framework. In *Proceedings of the Conference on Practical Applications of Agents and Multi-Agents (PAAM'99)*, pages 97–108, April 1999.

[Bellifemine *et al.*2001] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software - Practice And Experience*, (31):103–128, 2001.

[Berners-Lee *et al.*2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

[Camacho *et al.*2002] David Camacho, Ricardo Aler, César Castro, and José M. Molina. Performance evaluation of Zeus, Jade and SkeletonAgent frameworks. In *Proceedings of the IEEE Systems, Man, and Cybernectics Conference (SMC-2002)*, Hammamet, Tunisia, October 2002. IEEE.

[Camacho *et al.*2005] David Camacho, Ricardo Aler, Daniel Borrajo, and José M. Molina. A multi-agent architecture for intelligent gathering systems. *AI Communications. The European Journal on Artificial Intelligence. Ed. by IOS Press.*, To appear in 18(1), 2005.

[Collis *et al.*1998] J. Collis, D. Ndumu, Hyacinth S. Nwana, and L. Lee. The zeus agent building tool-kit. *BT Technology Journal*, 16(3):60–68, 1998.

[Finin *et al.*1994] Tim Finin, R. Fritzson, D. Mackay, and R. McEntire. Kqml as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94)*, pages 456–463, Gaithersburg, Maryland, 1994. New York: Association of Computing Machinery, ACM Press.

[Jeon *et al.*2000] Heecheol Jeon, Charles Petrie, and Mark R. Cutkosky. Jatlite: a java agent infrastructure with message routing. *IEEE Internet Computing*, 4(2):87–96, April 2000.

[Petrie1996] Charles Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, 11(6):24–29, December 1996.