Revista Cubana de Ciencias Informáticas
Vol. 9, No. 1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 78-93

Tipo de artículo: Artículo original
Temática: Técnicas de programación
Recibido: 14/11/2014 | Aceptado: 19/01/2015

Selecting frameworks for multi-agent systems development for the oil industry

Selección de frameworks para desarrollo de sistemas multi-agente para la industria petrolera

J. Antão B. Moura [1*], Pryscilla Dóra [2], Ana Cristina Oliveira [3]

[1] Systems and Computing Department, Federal University of Campina Grande (UFCG). CP: 10.106. E-mail: copin@dsc.ufcg.edu.br
[2] Unipetech, University Center of João Pessoa (UNIPÊ). CP: 318. E-mail: ubs@unipe.br
[3] Convergent Networks Research Group (GPRC), Federal Institute of Paraíba (IFPB). E-mail: campus_cg@ifpb.edu.br

[*] Autor para la correspondencia: {pryscilla.dora, cristina.alves, antaomoura}@gmail.com

Abstract
Throughout the years, the development of multi-agent systems (MAS) has evolved and many frameworks to support such development were proposed in the literature. Some frameworks did not advance; others have evolved according to the type of usage in either academy or industry. Given the large number of frameworks for developing multi-agent systems, it is important to evaluate these frameworks in order to select which one best suits the development project at hand. There seems to be scanty information and recommendations in the specialized bibliography on development frameworks for MAS to be used in support of supply chain management (SCM) by the petroleum industry in particular. We present in this work a methodology for comparing and choosing frameworks to be used for developing MAS for the oil industry. The methodology includes theoretical and practical aspects. Application of the methodology is carried out in a real case, supply chain management scenario offered by Petrobrás, the Brazilian petroleum company, in automating its planning for draining petroleum products.
Keywords: Evaluation Methodology, MAS Benchmark, Multi-agent Systems Framework.

Resumen
A lo largo de los años, el desarrollo de sistemas multiagente (MAS) ha evolucionado y se han propuesto muchos frameworks para apoyar ese desarrollo en la literatura. Algunos frameworks no avanzaron; otros han evolucionado de acuerdo con el tipo de uso, ya sea en la academia o la industria. Dado el gran número de frameworks para el desarrollo de sistemas multi-agente es importante evaluar estos con el fin de seleccionar el que mejor se adapte al proyecto de desarrollo en cuestión. No parece haber información y recomendaciones escasas en la bibliografía especializada sobre los frameworks de desarrollo para el MAS que se utilizará para apoyar la Gestión de la Cadena de Suministro (SCM) por la industria del petróleo en particular. Se presenta en este trabajo una metodología para comparar y elegir los frameworks que se utilizarán para el desarrollo de MAS para la industria petrolera. La metodología incluye aspectos teóricos y prácticos. La aplicación de la metodología se lleva a cabo en un escenario de gestión de la cadena de suministro de caso real ofrecido por Petrobrás, la empresa petrolera brasileña, en la automatización de la planificación para el drenaje de los productos del petróleo.
Palabras clave: *frameworks de sistemas multiagente, MAS Benchmark, metodología de evaluación.*

Introduction

Development frameworks allow the reutilization of software's codes and projects (or part of them). They are commonly described as a set of abstract classes, and they define how the objects of these classes communicate (ROBERTS, 1996). By allowing the reutilization of code and project, a good framework cuts costs of system development.

Multi-agent systems are especial types of distributed systems, formed by autonomous agents that have specific characteristics that function independently, but that compose together the distributed system's as a whole. They can be very complex and their development is not a trivial activity.

Multi-agent systems present certain common characteristics that can be mapped into a framework, leaving to the programmers only the effort to organize and develop the specific business logic according to each application.

One common characteristic of all multi-agent systems is the proper communication between the agents. Agents are autonomous entities that need to communicate in order to exchange, negotiate and deliberate information. Another common characteristic is the agent's abstraction. Since there is not a single definition of agent with which all researchers of the area agree upon, each framework brings its own way of representing an agent. Frameworks can provide interesting services, such as names' service and automatic agents' discovery.

Throughout the years, many frameworks for multi-agent systems development have been proposed in the literature – see for instance, (CHEYER, 1999), (CHEYER, 2001), (POSLAD, 2000), (AVANCINI, 2000), (HOWDEN, 2000), (HÜBNER, 2000), (BELLIFEMINE, 2003) and (BELLIFEMINE, 2008). Those frameworks were implemented in an isolated manner, with particular features for each specific context. These implementation differences make it difficult to choose one framework for multi-agent systems development for a given context. The context of interest here is that of the supply chain for the oil industry in Brazil.

The Brazilian oil supply chain is made up of oil refineries, consumer markets, terminals for intermediary storage and several transportation modals, such as pipelines, ships, trucks and trains. The transportation planning of oil products in this multimodal network is a complex problem that is being automated and whose development is based on team´s expertise. The lack of a Decision Support System (DSS) that considers the entire complexity of the problem, planning usually lasts for a period of three months. Due to the characteristics of the problem in study where the full network needs to be taken into account as well as the negotiations among the different entities involved, using multi-agent systems for automation offers a possibility Worth exploring. However, choosing a framework that offers a good fit to the context is not trivial – it is in fact, the motivating aspect for the comparative work presented in this paper.

The contribution of this work is to structure the evaluation of frameworks for multi-agent system development with the objective of selecting the one that best suits the systems' requirements.

The remainder of the contents of the paper is organized into five additional sections. In Section II, we present the methodology proposed to benchmark the frameworks. In Section III, we present results obtained from the evaluation of several frameworks regarding theoretical and practical aspects. Section IV contemplates the development of multi-agent systems using the two "best evaluated" frameworks to come out of the previous section. The framework of choice is finally selected in Section V. Related work is discussed in Section VI. Conclusions and brief comments on further work are offered in Section VII.

Materials and Methods

The frameworks' evaluation was conducted in two phases: (i) selection process; and (ii) analysis of selected frameworks.

Selection process – identifying potential frameworks for the context of interest

The research for identifying frameworks for multi-agent systems development was conducted in two ways:

- Revision of gray literature: informal research (e.g. *Wiki, blog, webpages*, etc.);

- Using automatic search engines: ACM (Advanced Search); IEEE (Advanced Search), *Google Scholar*, and *Isiknowledge*.
- After classifying the most frequently mentioned and used frameworks, an analysis was conducted to select and prioritize the most relevant ones for our purposes. Section III lists the initial set of identified, relevant frameworks of interest here.

Qualification process - criteria for analyzing the identified frameworks

Qualifying criteria break up into elimination and classification (ranking) criteria. The following elimination criteria were adopted:

- *Compatibility with the Foundations for Intelligent Physical Agents (FIPA)* (FIPA, 2012) *standards*: FIPA is a standardizing organization of the IEEE Computer Society, whose function is to promote the technology based on agents and the inter-operability with other technologies (FIPA, 2012a) and (FIPA, 2012b).
- *Compatibility with the JAVA Language*: Largely disseminated and used development language.
- *Tooling with Tech Support*: The level of support offered is important, because it indicates how easy it is to receive feedback from the framework development team.
- *Tooling with Maintainability*: The framework must have continuous maintenance and be in constant evolution.

After applying the elimination criteria, we chose the following criteria for classification (ranking) purposes:

1) Agents Representation: how an agent can be created, what is its life cycle, what are its characteristics, and how it is used.
2) Partners Identification: number of teams using the framework.
3) License Type: Identifying what kind of license is associated with the framework.
4) Architecture: Indicating how the main classes communicate and where the project standards are used
5) Communication Infra-structure: communication protocol that must be used among agents, and how to represent the data.
6) Intelligent Characteristics: characteristics that are encapsulated by frameworks to facilitate the project of a new system.

Qualified frameworks (i.e., those classified or ranked frameworks that pass the elimination process) were then applied to the implementation of a simple example of a multi-agent system in the form of a Trading Agent Competition (TAC) (TAC, 2009). The purpose of this implementation was to evaluate practical aspects such as: usability, functionalities, development environment, libraries, and support tools.

TAC Competition

The TAC competition was conceived to capture some of the challenges concerning the support to the dynamic practices of the supply chain, keeping the game rules simple enough in order to attract a great number of competitors. The game was created in collaboration with researchers from Carnegie Mellon University and the Swedish Institute of Computer Science (SICS) (TAC, 2009).

We have submitted multi-agent systems developed using the selected frameworks by direct competition among the agents. In this sense, it was possible to evaluate not only overall development aspects, but also the intelligence quality of each system.

The motivation for choosing the TAC-SCM competition was based on its similarity with the real environment in the supply chain for the oil industry in Brazil. Besides, the acquired knowledge will help develop a simulator with use cases based on Petrobras Supply Chain, in order to support the decision-making process for the distribution of oil-derived products through the transportation system.

Results and Discussion

Research (Identification) Results

After executing the methodology for identifying the most referenced frameworks, was obtain the following list:

- JADE – *Java Agent Development framework*;
- FIPA-OS – *Foundation for Intelligent Physical Agents Operating System*;
- JACK – *Intelligent Agents*;
- OAA – *Open Agent Architecture* (CHEYER, 1999) (CHEYER, 2001);
- SACI – *Simple Agent Communication Infrastructure*.

After identifying the frameworks of interest, the next step was to evaluate them according to the theoretical, qualifying criteria for.

Qualification results for the selected Frameworks

The theoretical evaluation is subdivided in two sets of criteria, eliminatory and classificatory. The eliminatory set identifies which frameworks are not compatible with the macro-needs of the project. The second set of criteria (classificatory) aims at identifying other important characteristics to support the final choice for the framework to be adopted in the development project.

Table 1 – Comparison of frameworks according to the four elimination criteria.

Criteria	**JADE**	**FIPA**	**JACK**	**OAA**	**SACI**
FIPA *compliant*	Yes	Yes	Can use	No	No
Language	Java	Java	Java Extension	Support Java	Java
Support	Yes	No	Yes	Yes	No
Maintainability	Yes	No	Yes	Undefined	No
Overall Result	Apt	Failed	Apt	Failed	Failed

According to the data presented in Table 1, only two frameworks were approved, JADE and OPENAGENT. (Any "No" entry in the table eliminates the corresponding framework.) FIPA-OS was eliminated because it did not provide support, and was not frequently maintained. OAA was eliminated because it was not FIPA-Compliant, besides not offering much insight into its maintainability (OPENAGENT, 2000), (OPENAGENT, 2001), (OPENAGENT, 2005), (OPENAGENT, 2006) and (OPENAGENT, 2009). Finally, SACI was eliminated because it was not considered a framework by its developers, and it offered no support nor was it FIPA-Compliant. Thus, the only frameworks that are apt to go to the next stage of evaluation (practical evaluation) are JADE and JACK.

Practical Evaluation

Practical evaluation was carried out in two steps:

- Framework Recognition – recognition of the environment and available resources for development;
- Agent Implementation – implementation of the MAS to compete at the TAC-SCM.

A. *JADE Agent Development Framework – JADE*

The JADE framework usability, which may also be understood as "*programmability*", is considered simple for those who have knowledge of the Java language. The JADE library and all of its add-ons can be imported to Eclipse, one of the most well-known development environments among Java programmers. Besides that, Jade implements the FIPA communication standard, which is currently the most used and most referenced standard for agents' communication.

The JADE framework documentation is quite complete, clearly describing how to create the agents, how to perform communication between agents, and how to define all the agents' behaviors.

The execution of a JADE agent can be initiated using the name of the agent that one wish to initiate followed by the path of its code and passing them as arguments to the *jade.Boot* class (j*ava jade.Boot <agent name>: <code path>).*

We had a problem with the initiation of Jade agents within a program, i.e., inside *themain* class. It is not possible to create an isolated instance for a JADE agent, because it always executes inside the *container* associated to Jade. However, we wanted to create agents able to connect to the game framework, and not only to the JADE manager. This way, in order to execute a JADE agent from any program, we used the following code within the main method of such program:

```
    String[] param = new String[2];
    param[0] = "-gui";
    param[1]= "<agent name>:<code path >";
   Boot.main( param );
```

The first parameter is to initiate the graphic interface and the second is to initiate the agent of name <agent name> that is implemented by the class correspondent to the specified path.

This form of initiating agents allows us to initiate JADE agents in any *Main* class, which is necessary for implementing the agent for the TAC SCM. Despite those code line being not explicit in the JADE documentation, it was easy to find them in the forums and lists available at the framework website.

To study the framework, we implemented some examples found in the documentation. One of the implemented examples reunites the framework's main characteristics, such as *behaviours* and message exchange. The first agent to be implemented in the following example is the *SenderAgent*, which can be seen bellow:

```
public class SenderAgent extends Agent {
   public void setup() {
      sendMessage();
   }
   private void sendMessage() {
      AID r = new AID "jack@"+getHap(),
         AID.ISGUID);
      ACLMessage aclMessage = new
         ACLMessage(ACLMessage.REQUEST);
      aclMessage.addReceiver(r);
      aclMessage.setContent("Hello! How
         are you?");
      this.send(aclMessage);
   }
}
```

In this class we have the setup method, which is responsible for initializing the moment that the agent must start to execute. It is similar to the run method of a thread. This agent initiates calling the *sendMessage*() method, which is responsible for sending a message to the JACK agent.

In the *sendMessage*() method, we have the creation of the agent id in the first line and then the construction of the *ACLMessage* (Access Control List Message), FIPA standard message, finalizing with the message dispatch.

The agent that receives this message is implemented by the code bellow:

```
public class ReceiverAgent extend Agent{
   public void setup() {
      System.out.println("Hello. My name
              is "+this.getLocalName());
      addBehaviour(new ResponderBehaviour
               (this));
   }
}
```

In that class, we also have the setup method, which is initiating the agent action through a *Behaviour*. The *Behaviour* represents an action (behavior) that the agent is adding, and is implemented by the code bellow:

```
public class ResponderBehaviour extends
SimpleBehaviour {
 private static final MessageTemplate mt =
      MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
public ResponderBehaviour(Agent agent) {
            super(agent);
}
 public void action() {
          ACLMessage aclMessage = myAgent.receive(mt);
      if (aclMessage!=null)
            System.out.println(myAgent.          getLocalName()+":          I          receive
message.\n"+aclMessage.getContent());
          } else {
             this.block();
           }
}
public boolean done() {
            return false;
}
}
```

In the prior code, we have the association between message type and the receiver which is waiting for it. The receiver only receives messages of this type. The *Behaviour* needs to keep executing constantly, so in order for it to execute only when it receives a message, there is a *block* in case it has not received any messages. The *done* method is responsible for verifying if the *Behaviour* has already completed its task, or if it needs to continue executing. In this case, it always returns *false*, which indicates that the behavior continues associated with the agent.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 78-93

A - Agent Implementation

For the implementation of the MAS, our development team was divided into two groups, one responsible for the MAS implementation using the JACK framework, and another responsible for the MAS implementation using the JADE framework. This division was conceived with the purpose of evaluating the frameworks for further choosing, which one would be used in the MAS SCM.

The implementation started with a TAC TEX analysis (the basic TAC TEX agent made available in this agent's website). By analyzing this agent, we noticed that all of its strategies were implemented basically in two classes: *SimpleSupplyManager* and *SimpleDemandManager*.

After overcoming the obstacles mentioned in the previous section, we had no difficulties in mapping the TAC TEX for implementation with the Jade framework. We used the *Main* class of the TAC SCM, the class that starts the competition, and in it we also initiated the two agents that would be implemented by the two basic TAC TEX classes. This can be seen in the code bellow:

```
String[] param = new String[2];
param[0] = "-gui";
param[1] = "comprador:SimpleSupplyManager     escalonador:SimpleDemandManager";
Boot.main( param );
```

After creating the JADE agents, we started mapping its methods for *Behaviours*, which are the actions (behaviors) an agent has. An example of this can be seen bellow:

The *predictSalesPrices* method of the *SimpleDemandManager* class, which can be seen in the code bellow:

```
private void predictSalesPrices(){
    for (int i = 0; i < numComputers;i++){
        if (date > 0 &&  agentInfo.computers.lowPrice[i][date-1] > 0)
        salesPrices[i] = agentInfo.computers.lowPrice[i][date-1];
        else
            salesPrices[i] = 0.75 agentInfo.computers.basePrices[i];
       }
}
```

It was mapped to a *Behaviour* using the following code:

```
public class PredictSalesPrices extends Behaviour {
        public void action(){
           for (int i = 0; i < numComputers; i++){          if       (date       >       0       &&
agentInfo.computers.lowPrice[i][date-1] > 0)
salesPrices[i] = agentInfo.computers.lowPrice [i] [date-1];
             else
```

```
                salesPrices[i] = 0.75 agentInfo.computers.basePrices[i];
                 }
              }
              public boolean done() {
                 return false;
              }
         }
}
```

In the *Behaviour* implementation, we have the *action*() method, which has the same implementation of the original TacTex method, and, shortly after, the *done*() method, which is responsible for indicating if a *Behaviour* has already finished what it was supposed to do, or if it needs to remain active, and repeat this behavior. In this case, *done*() always returns a *false* response, meaning that the *Behaviour* must remain active and be repeated.

Within the JADE agent, the *callback* to the behavior is done as follows.

```
public void predictSalesPrices() {
      new PredictSalesPrices().action();
}
```

The method was modified to public in order to be called outside the class, and inside it there is only the *Behaviour* instantiation alongside the *action*() method call, which is responsible for the initialization of the behavior.

This mapping was done with all methods of the two main TAC TEX classes already mentioned.

Regarding the communication, messages exchange was not necessary, since those classes did not communicate with one another, and there were the only JADE agents in the system. This way, the communication was always done via callback methods.

B - Intelligent Agent – JACK

The JACK documentation is very complete, containing various exercises that guide the development of an agent/a multi-agent system and its interactions. Following those exercises step–by–step, it is possible to build a simple MAS with no additional difficulties.

The complete MAS development is obtained through the resolution of eleven consecutive exercises, where each one represents one part of the complete system. Each exercise goal is obtained after approximately 10 (ten) work hours. We will describe how the development of the exercises was conducted.

Exercise 1: Building a painter robot agent, with the Java method, in order to auto-paint oneself in any given part of the body.

Exercise 2: Enlarging the robot in order to allow it to use a Jack *Plan* (action plan to insert some intelligent capacity in the agent).

Exercise 3: Enabling the robot to select, among multiple plans through the relevance (action to know which plan is more adequate). The relevance is divided into two methods, *relevant* (method to check the event's additional independent information), and *context* (method to check the stored information as part of the agent's point of view).

Exercise 4: Enabling the robot to select among multiple plans through the *context* and the *relevance*.

Exercise 5: Illustrating the events repository when a plan fails.

Exercise 6: Providing the *Painting* capability.

Exercise 7: Creating a MAS consisting of a robot agent and a portion of the robot agent.

Exercise 8: Developing the communication protocol (messages' exchange) between the robot and its portion, and demonstrating the use of the Jack Interaction Diagram.

Exercise 9: Modifying the robot agent behavior so that the painting does not exceed a certain time limit.

Exercise 10: Synchronizing the communication protocol between the robot and its portion.

Exercise 11: Using a *semaphore* (synchronization resource that can be used to establish mutual exclusion of processing regions from Jack's plans and *threads*).

In this context, we observe that the architecture of a JACK agent/multi-agent system is very rich. It is possible to build an agent with its events, plans, capabilities, and so on that is possible or necessary for the agent development. When creating the agent in *Design Views*, the system sets the agent in a specific class, as well as its plans, events, etc., by adding *links* between the agent and its components, then, whole relationship structure is created.

C- Agent Implementation

Both JACK and JADE development teams built the MAS using the TAC TEX as the underlying base (the basic agent available in the TAC SCM game website).

The implementation was done via a TAC TEX analysis (PARDOE, 2009), in which we noticed that the TAC TEX is composed by two agents, *SimpleSupplyManager* and *SimpleDemandManager*. Those same agents were mapped into our MAS and, consequently, adapted for the JACK framework. This mapping was more complex than expected, because of the difficulty of translating the TAC TEX Java code to the extension used by JACK.

We will describe some of the difficulties faced during the development process:

- Non-friendly IDE and poor usability;

- It was not possible to integrate JACK's library with the Eclipse IDE;
- Because of the first item, routine activities became tedious, such as compilation errors and troubleshooting;
- It was not possible to separate each class in its proper *package* – it was necessary to group them in a single *package*, procedure that increased the programmers' efforts and diminished the performance during the system compilation;
- Using JACK's IDE, it was not possible to use the autocomplete command, which made the typing subject to errors and increased the development time;

The main methods for executing certain tasks had to be transformed into events and plans, such type of changes demanded more effort, time, and dexterity from the programmer's side.

D - Framework choice

After concluding the MASs, TAC TEX JACK (MAS created with JACK) and TAC TEX JADE (SMA created with JADE), the internal competition between the MAS-SCM project members started. Two matches were played, whose results can be seen in Table 2 (first match) and Table 3 (second match). Based on these results, we will analyze the matches' winner, as well the ease of programming, and come up will the winner framework.

Table 2 : Matches between JACK MAS versus JADE MAS

Player	Revenue	Interest	Costs				Margin		Results
			Material	Storage	Penaltry		1	2	
TACTEXJADE	136.235.790	473.877	94.464.057	1.355.561	5.275	0%	30%	30%	40.884.774
TACTEXJACK	133.288.950	384.128	94.909.627	1.940.540	1.242.871	1%	27%	27%	35.580.040
Dummy-4	52.588.441	182.522	39.071.320	389.635	1.091.118	3%	25%	23%	12.218.890
Dummy	53.944.916	177.290	40.535.547	393.268	1.280.554	3%	24%	22%	11.912.837
Dummy-3	54.749.289	174.013	41.349.936	402.099	1.396.651	3%	24%	22%	11.774.616
Dummy-2	53.468.074	172.962	40.052.639	396.668	1.588.978	4%	24%	22%	11.602.751

Table 3: Matches between JACK MAS versus JADE MAS and System Dummys

Player	Orders	Utilization	Deliveries (on time/late/missed)		Dperf
TACTEXJADE	8.215	96%		8212 / 3 / 0	100%
TACTEXJACK	7.103	96%		6908 / 115 / 80	97%
Dummy-4	2.211	31%		2038 / 168 / 5	92%
Dummy	2.240	31%		2065 / 166 / 9	92%

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 78-93

Dummy-3	2.342	32%		2147 / 186 / 9	92%
Dummy-2	2.263	31%		2053 / 200 / 10	91%

Table 2 shows one of the matches between TACTEXJADE and TACTEXJACK. Many other matches were played. In all matches the TACTEXJADE was superior than the TACTEXJACK in every analyzed aspect. The most important aspects are mentioned bellow:

- **Revenue** (second column): TACTEXJADE was 2% superior in regard to TACTEXJACK, and in average 250% superior concerning *Dummys* (agents made available by the server).
- **Interest** (third column): 473,877 for TACTEXJADE and 384.128 for TACTEXJACK; difference of 23%.
- **Costs** (fourth, fifth and sixth columns): although the costs with piece's purchases were compatible, TACTEXJADE presented lower costs with storage, indicating a bigger production, less storage time, and, consequently, lower penalty.
- **Total profit margin** (seventh and eighth columns): the total profit, with and without bank taxes and penalties, was of 30% for TACTEXJADE and 27% for TACTEXJACK.

  In table 3, we can verify the superiority of the TACTEXJADE results, according to the statistics provided by the game:
- **Clients' orders**: it obtained 8215.15% more orders when compared to TACTEXJACK;
- **Industry utilization**: both used an average of 96% of the industry;
- **Deliveries**: it is important to highlight that TACTEXJADE delivered only three late orders, while the other agents delivered more than one hundred late orders;
- **Delivery performance**: TACTEXJADE scored 100% in its deliveries, while the others scored an average of 92,8%.

Because of the learning difficulties of the JACK framework, (by using a not so friendly IDE and presenting poor usability), the agent development was more expensive. On the other hand, the development of the Jade framework agent was faster and allowed the development of more elaborate game strategies. The JADE framework popularity is higher, which makes it easier to find needed information via search engines. Since the JADE users' community is bigger and active, they frequently share tips about problem solutions in websites and forums.

E - Related Work

Multi-Agent System (MAS) development is one of the areas where researchers are intensively pursuing new solutions by comparisons of frameworks to build new systems up more effectively. Regarding MAS comparisons, there are many solutions for modeling and simulating complex MAS (MOYA, 2007). There are authors who have compared the metrics of software engineering techniques for developing complex system with multi-agent approaches (GARCIA, 2011), the development of oriented robots using MAS (BLASCO, 2012), or tools for multi-agent development techniques (GARCIA, 2010) or then a new way of tackling complex problems based on the social metaphor (ARAUJO, 2014). Casare et al. presented a comparison about MAS situational methods, as well as the main components of a MAS application, such as agents, environments, interactions, organizations, and when to use each method or component (CASARE, 2014). However, in this work, we compared framework solutions applied to the supply chain management (LONG, 2014) with focus on draining petroleum products. As such this work can be seen as complementing these related works.

Conclusions

This paper proposed a methodology and theoretical and practical criteria for comparing and choosing frameworks for the development of multi-agent systems (MAS). The proposal was applied to the development of agents in a supply chain management (SCM) context for the Brazilian oil industry.

Five frameworks were initially selected. Elimination criteria reduced this number to two: JACK and JADE. According to all evidences, criteria and analysis, we opted to use JADE as MAS development framework. Apart from being very popular among MAS developers', JADE also provided the best results in our analysis. Implementation was easier because of the development environment that JADE provides to developers: assistance tools and simplicity. JADE is also one of the main MAS development frameworks for the purpose of developing a Decision Support Systems.

By comparing MAS development frameworks within an SCM context for the oil industry, this paper complements related works that offer insight into such type of comparisons but for other applications and industry sectors.

References

- ARAUJO, P., B. RODRIGUES, S., A. “Enfoque para a la validación sintáctica de modelos organizacionales de Sistemas Multiagentes” Ciencia y Tecnología. 2014, pp. 123-144 ISSN 1850-0870.
- AVANCINI, H., AMANDI, A. “A Java framework for Multi-agent Systems”. SADIO Electronic Journal of Informatics and Operations Research. 2000, vol. 3, no. 1, pp. 1-12.

- BELLIFEMINE, F., CAIRE, G., POGGI, A., RIMASSA, G. "JADE: a White Paper. EXP". In search of innovation Journal. September 2003, vol. 3, no. 3, pp. 6-14.
- BELLIFEMINE, F., CAIRE, G., POGGI, A., RIMASSA, G. JADE: "A software framework for developing multi-agent applications". Lessons learned. Information and Software Technology. 2008, vol. 50, pp. 10-21.
- BLASCO, P. I., RIO, F. D., TERNERO, M. C. R., MUÑIZ, D. C., DIAZ, S. V. "Robotics software frameworks for multi-agent robotic systems development". In: ACM Robotics and Autonomous Systems. June 2012, vol. 60, Issues 6, pp. 803-821.Amsterdam, Netherlands. DOI: 10.1016/j.robot.2012.02.004.
- CASARE, S. J., BRANDÃO, A. A., GUESSOUM, Z., SICHMAN, J. S. "Medee Method Framework: a situational approach for organization-centered MAS". In: Autonomous Agents and Multi-Agent Systems. May 2014, vol. 28, Issue 3, pp 1-44. Hingham, MA, USA. DOI: 10.1007/s10458-013-9228-y.
- CHEYER, A., MARTIN D., and MORGAN, D. "The Open Agent Architecture: A framework for Building Distributed Software Systems". In: Artificial Intelligence. January 1999, vol. 13, no. 1-2, pp. 91-128 [cited 2014-02-23]. Available from: http://www.ai.sri.com/pubs/files/415.ps.gz
- CHEYER, A. and MARTIN, D. "The Open Agent Architecture". Journal of Autonomous Agents and Multi-Agent Systems. March 2001, vol. 4, no. 1, pp. 143-148.
- Foundations for Intelligent Physical Agents (FIPA). "Design Process Document Template". IEEE Foundation for Intelligent Physical Agents, 2012. [Cited: 2014-11-23]. Available from: http://www.fipa.org/
- Foundations for Intelligent Physical Agents (FIPA). "Abstract Architecture Specification". IEEE Foundation for Intelligent Physical Agents, 2012b. [Cited 2014-11-28]. Available from: http://www.fipa.org/specs/fipa00001/ SC00001L.pdf
- GARCIA, E., GIRET, A., BOTTI, V. "An evaluation tool for multiagent development techniques". In: AAMAS '10 Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems. 2010. vol. 1, pp. 1625-1626. ISBN: 978-0-9826571-1-9. 2010.
- GARCIA, E., GIRET, A., BOTTI, V. "Evaluating software engineering techniques for developing complex systems with multiagent approaches".Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems. May 2011. pp. 1625-1626. ISBN: 0-98265-710-0/978-0-9826571-1. DOI: 10.1016/j.infsof.2010.12.012.
- HOWDEN, N., RONNQUIST, R, HODGSON, A., LUCAS, A. "Jack Intelligent Agent – Summary of an Agent Infrastructure". In 5th International Conference on Autonomous Agent. 2000. [2014-08-08] Available from: http://www.ece.byu.edu/faculty/jka/class/522rf06/ howden.pdf

– HÜBNER, F., SICHMAN, S. SACI: "Uma ferramenta para implementação e monitoração da comunicação entre agentes". Anais do International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI (Open Discussion Track), PP. 47-56, São Carlos, 2000. ICMC/USP. Available from: http://www.inf.furb.br/~jomi/pubs/2000/Hubner-iberamia2000.pdf.
– LONG, Q. "An agent-based distributed computational experiment framework for virtual supply chain network development". In: ACM Expert Systems with Applications: An International Journal, Volume 41. Issues 9, July, 2014. Tarrytown, NY, USA. DOI: 10.1016/j.eswa.2014.01.001.
– MOYA, L. J., TOLK, A. "Towards a taxonomy of agents and multi-agent systems". In: Proceedings of the 2007 spring simulation multiconference. 2007, vol. 2, pages 11-18.
– OPENAGENT. "Architecture (OAA®) v2.x FAQ". [Letter]. 2001. Available from: http://www.ai.sri.com/~oaa/distribution/doc/oaa-faq-v2.html
– OPENAGENT. "Documentation. OAA® 2.3.2" [Letter]. 2000. Available from. http://www.ai.sri.com/~oaa/distribution/v2.3/2.3.2/
– OPENAGENT. "Overview Presentation. OAA®" [Letter]. 2006. Available from: http://www.ai.sri.com/~oaa/oaaslides/newoaa.ppt
– OPENAGENT. "The Open Agent ArchitectureTM". [Letter]. 2005. Available from http://www.ai.sri.com/~oaa-lite.html/
– PARDOE, D., STONE, P., VAMMIDDLESWORTH, M. "TacTex-05: An adaptive agent for TAC SCM". In AAMAS 2009 Workshop on Trading Agent Design and Analysis / Agent Mediated Electronic Commerce. 2009, vol. 4452, pp. 46-61, Springer.
– POSLAD, S., BUCKLE, P., HADINGHAM, R. "The FIPA-OS agent platform: Open Source for Open Standards". In Proceedings of PAAM 2000. Manchester. UK. 2000. Available from http://fipa-os.sourceforge.net/docs/papers/FIPAOS.pdf.
– ROBERTS, D., JOHNSON, R. "Evolving frameworks: A Pattern Language for Developing Object-Oriented frameworks". In Proceedings of the Third Conference on Languages and Programming. University of Illions, 1996. Available from http://st-www.cs.uiuc.edu/users/droberts/evolve.html.
– TAC. Trading Agent Competition – Supply Chain Management - Official competition site. Competition 2009. Available from: http://www.sics.se/tac/