MISIA: Middleware Infrastructure to Simulate Intelligent Agents

Elena García, Sara Rodríguez, Beatriz Martín, Carolina Zato, and Belén Pérez

Abstract. Nowadays there is a clear trend towards using methods and tools that can help to develop multiagent systems (MAS). This study presents a multiagent based middleware for the agents behavior simulation. The main challenge of this work is the design and development of a new infrastructure that can act as a middleware to communicate the current technology in charge of the development of the multiagent system and the technology in charge of the simulation, visualization and analysis of the behavior of the agents. It is a key element when considering that MAS are autonomous, adaptive and complex systems and provides advances abilities for visualization. The proposed middleware infrastructure makes it possible to visualize the emergent agent behaviour and the entity agent. It also allows visualization of the interaction between the agent and the environment.

Keywords: Multiagent systems, Simulation, JADE, Repast.

1 Introduction

Nowadays, simulation is used for several purposes ranging from work flow to system's procedures representation. Simulation can be defined as the representation of the operation or features of one process or system through the use of another. The current research lines that attempt to answer the question of which technique is best for different purposes of simulation are gaining importance. The contribution from agent based computing to the field of computer simulation me-diated by ABS (Agent Based Simulation) is a new paradigm for the simulation of complex systems that require a high level of interaction between the entities of the system. Possible benefits of agent based computing for computer simulation include methods for evaluation of multi agent systems or for training future users of the system [5]. Many new technical systems are distributed systems and involve complex interaction between humans and machines, which notably reduce their usability. The properties of ABS makes it especially suitable for simulating this kind of systems. The idea is to model the behaviour of the human users in terms of software agents.

Elena García · Sara Rodríguez · Beatriz Martín · Carolina Zato · Belén Pérez
Computers and Automation Department, University of Salamanca, Salamanca, Spain
e-mail: {elegar,srg,eureka,carol_zato,lancho}@usal.es

A. Abraham et al. (Eds.): International Symposium on DCAI, AISC 91, pp. 107–116.
springerlink.com

However, it is necessary to define new middleware solutions that allow the connection on ABS a simulation software.

This paper describes the results achieved towards a multiagent-based middleware for the agents' behavior simulation. The middleware, called MISIA (*Middleware Infrastructure to Simulate Intelligent Agents*), allows simulation, visualization and analysis of the agent' behavior. The main contribution of this paper is the design of a new infrastructure that make it possible to provide these capabilities. MISIA makes use of technologies for the development of multiagent systems known and widely used, and combines them so that it is possible to use their capabilities to build highly complex and dynamic systems. On one hand, it is JADE [9],the most widely used platform for based software agents middleware. On the other hand, it is Repast (Recursive Porous Agent Simulation Toolkit) [10], a free and open-source agent-based modeling and simulation toolkit.

Our second contribution is the reformulation of the FIPA protocol used in JADE [9], achieving several advantages: (i) development of a new framework that provides independence between the model and visualization components; (ii) improvement on the visualization component that makes it possible to use the concept of "time", essential for simulation and analysis of the behavior of agents; (iii) and improvements to the user capabilities to which several tools were added, such as message visualization, 2D (and future 3D agents), analysis behavioral, statistics, etc.

Both contributions resulted in the first middleware infrastructure to simulate intelligent agents with visualization, simulation and analysis capabilities.

The article is structured as follows: Section 2 makes a review of agent-modeling toolkits and presents the challenges for simulated multiagent systems. Sections 3 introduces a description of the middleware specifically adapted to the simulation of multiagent systems within dynamic environments (MISIA). Finally, some results conclusions are given in Sections 4 and 5.

2 Background

Agents and multiagent systems are adequate for developing applications in dynamic, flexible environments. Autonomy, learning and reasoning are especially important aspects for an agent. These capabilities can be modelled in different ways and with different tools [15]. Open MAS should allow the participation of heterogeneous agents with different architectures and even different languages [16][4]. The development of open MAS is still a recent field of the multiagent system paradigm and its development will allow applying the agent technology in new and more complex application domains. Thanks to the contribution from agent based computing to the field of computer simulation mediated by ABS is obtained benefits like methods for evaluation and visualization of multi agent systems or for training future users of the system [4]. There are existing work on agent simulation study using commercial-offthe- shelf simulation packages with built-in agent-based modeling and BDI (Belief-Desire-Intention) behaviour architecture [11], modeling detailed complex human behaviours.

Mainly there are two ways for visualizing multiagent systems simulation: the agents interaction protocol and the agent entity. In the former, it is visualized a sequence of messages between agents and the constraints on the content of those messages. On the other hand, the latter method visualizes the entity agent and its iteration with the environment. Most software programs, such as JADE platform [2][9] and Zeus toolkit [3], provide graphical tools that allow the visualization of the messages exchanged between agents.

The toolkits MASON [6], Repast [7][10] and Swarm [12] provide the visualization of the entity agent and its interaction with the environment. The Swarm [12] is a library of object-oriented classes that implements the Swarm conceptual framework for agent-based models and provides many tools for implementing, observing, and conducting experiments on ABS. The MASON [6] is multiagent simulation library core developed in Java. It provides both a model library and an optional suite of visualization tools in 2D and 3D. The Repast [10] is a free and open-source agent-based modelling and simulation toolkit. There are other works like Vizzari et al. [14] where is developed a framework supporting the development of MAS-based simulations based on the Multilayered Multiagent Situated System model provided with a 3D visualization. We did not adopt this framework because would add a non-desired complexity to our system. We chose the Repast toolkit because, when the project started, it was one of the few to offer 3D visualization feature, as well as being simple and having good documentation. Moreover, the Repast system, including the source code and is available directly from the web. Repast seeks to support the development of extremely flexible models of living social agents, but is not limited to modelling living social entities alone. Repast is differentiated from other systems since it has multiple pure implementations in several languages and built-in adaptive features such as genetic algorithms and regression [8].

3 MISIA Middleware

The most well-known agent platforms (like Jade [9]) offer basic functionalities for the agents, such as AMS (Agent Management System) and DF (Directory. Facilitator) services; but designers must implement nearly all organizational features by themselves, like simulation constraints imposed by the MAS topology. In order to model open and adaptive simulated systems, it becomes necessary to have an infrastructure than can use agent technology in the development of simulation environments.

The framework presented in this paper is called MISIA (*Middleware Infrastructure to Simulate Intelligent Agents)*. It is a middleware infrastructure that allows to model JADE multiagent systems with the possibility of being represented in Repast. The main concept introduced in this environment is the notion of time in JADE, which means it is possible to render in real time the events into Repast. One of the main differences between JADE and Repast is that in JADE, there not exists the concept of time as such, and the agents interact each other based on changes or events that occur in the execution environment. However, Repast has a time unit : the tick, which is what sets the pace and allows simulations.

On the other hand, agents in the JADE context are implemented based on FIPA standards. This allows to create multiagent systems in open environments, which is not possible within Repast. These differences are what MISIA solved, integrating these two environments and achieving a working environment for creation and simulation of multiagent systems more powerful and versatile.

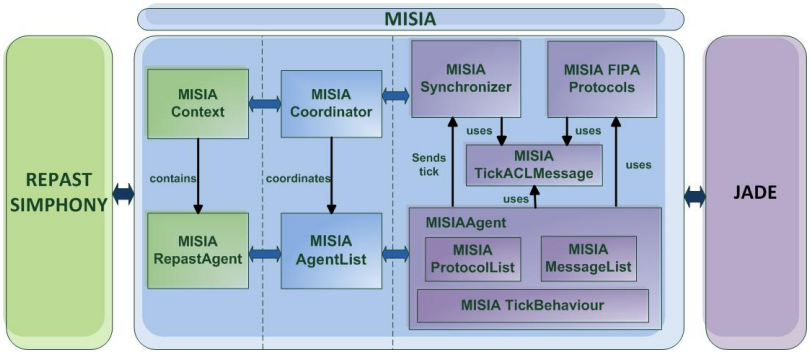

Fig. 1 Functional structure of MISIA

MISIA consists of three principle components or layers: the upper layer is the contact with Repast, the intermediate layer, whose main goal is the interconnection of the two platforms (JADE and Repast), and the bottom layer, which enables JADE supports the notion of time. Next, we will proceed to explain each functional block independently, but to better understand the overall operation, it is necessary to take in mind the following ideas: It is neccesary to syncronize JADE to work simultaneously to Repast. This is achieved by keeping the JADE agents informed about the tick of the simulation they are involved. Moreover, agents are informed when a tick is elapsed. To obtain versatile simulations, it is necessary that all events occurring in JADE are rendered instantly on Repast. The minimum unit of time is the tick, thus, the idea is that every JADE agent can perform functions in a tick (must be simple actions, such as sending a message, receiving or reestablishment of their state) and once finished, they can be updated in Repast. This must occur during the course of all ticks, which are getting updated in real time all events.

The bottom layer of the framework is which connects JADE, and is divided into four functional blocks: (i) **MISIAAgent,** is the extension of JADE agent. Performs the same functions, but adapting them to the presence of ticks. It consists of a number of features to manage the time in JADE. These functions are detailed in the following subsection. (ii) **MISIATickACLMessage.** JADE messages are used for communication between agents. MISIAAgent agents communicate between them with MISIATickACLMessage messages. MISIATickACLMessage is the extension of JADE ACL message that incorporates the concept of time. It includes aspects such as the tick where to send the message, and the delay that the message has when achieves its destination. In JADE, the messages exchanged between agents are sent and arrive instantly, but in real life, that is not the case. It aims to

simulate and view the evaluation of the system as time passes, and to achieve this, it is necessary that messages are not instant, but must have a shipping time and a different reception time. (iii) **MISIAFIPAProtocols.** As discussed above, JADE implements FIPA standards, which, among other things, specify multiple communication protocols. These define a series of patterns that respond to different types of communication that two or more agents can perform. The objective is to adapt FIPA protocols defined in JADE with Repast ticks. Features of this module will be detailed in Subsection 3.2. (iv) **MISIASynchronizer** is a JADE agent that acts of notificator. It is responsible for notifying the MISIAAgent when a tick goes by. Is the system clock synchronization. When a tick goes by, MISIASynchronizer is notified in order to notify MISIAAgents. It is made through MISIATickACLMessage messages with a special performative.

The top layer is the contact with Repast. Contains two functional blocks, which are: (i) **MISIARepastAgent**. Each MISIAAgent existing in the system will be represented by a MISIARepastAgent in the context of Repast. This means that for every agent that we want to have on the system actually have to create two: a MISIAAgent agent running on JADE , and its respective MISIARepastAgent released on Repast. It can be seen as follows: a logical agent, and two physical agents. MISIARepastAgents have an important role: they cannot updated their status until their respective MISIAAgents does not end with all the work they need to perform during that tick. This is a very important aspect, since it is the characteristic of the framework as a system in real time. (ii) **MISIAContext** has two important objectives. One is to establish the synchronism in the execution. When a tick goes by, lets know MISIASynchronizer agent that it is necessary to notify MISIAAgent agents that following tick happened. The other goal of this module is to incorporate new agents MISIARepastAgent that entry in the context of the Repast simulation. For each new MISIAAgent that appears in the system, MISIAContext will create their respective MISIARepastAgent and will added it to the simulation environment.

Finally, the intermediate layer is divided into two functional blocks, and its goal is to join adjacent layers. These modules are: (i) **MISIAAgentList**, as its name implies, stores all agents in the system at a given time. It plays an important role because it enables communication between a MISIAAgent and their respective MISIARepastAgent, and vice versa. The diagram shows two-way information flows ranging from MISIARepastAgent to MISIAAgentList and MISIAAgentList to MISIAAgent. These flows are representing that communication, that union between the two physical agents, to confine a logical agent. (ii) **MISIACoordinator** coordinates communication between the two adjacent layers. It is necessary the presence of a coordinator to maintain synchronism between both layers. Thanks to MISIACoordinator, MISIAContext can notify the occurrence of a tick to MISIASynchronizer, and MISIASynchronizer can assure that its purpose is served to MISIAContext, reporting that all MISIAAgent received tick. This kind of communication is necessary to maintain full synchronization between the two platforms. Also shown in the diagram two flows between MISIAContext and MISIACoordinator, between MISIACoordinator and MISIASynchronizer, and represent the flow of ticks and the synchronism.

3.1 JADE Adapted to the Notion of Time

The adaptation of JADE to support the notion of time is the most important and complex feature of our proposal. It was necessary to redefine a series of classes of JADE agents. The new capacity autonomously manages the receipt of ticks and maintains synchronism with Repast, so that it abstracts all these aspects and provides flexibility to the final programmer who uses this framework. Broadly, the sequence of steps that occurs for a tick is: (i) The occurrence of a tick is generated on Repast, as it is the platform that has this ability. MISIAContext is notified of this and, through MISIACoordinator can notify MISIASynchronizer of its tasks. (ii) When MISIASynchronizer receives the notice of MISIAContext, alerts all MISIAAgent about the new tick. As MISIASynchronizer is an agent, it can communicate with MISIAAgent agents through MISIATickACLMessage messages. That's how it notifies them the new tick, sending a message with special semantics to each agent. (iii) Once MISIASynchronizer has sent all the special messages notification of tick, it must wait until all MISIAAgent answer to ensure that all received the notification. Moreover,it sends a ACK message to maintain a strong synchronism between two sides. (iv) When a MISIAAgent receives a tick by MISIASynchronizer carries out various actions. First it sends the messages has to ship in this tick. MISIAAgent agents have a special queue for sending messages. Below is a comparative picture of how is shipping in JADE, and what was the change made in the framework for adaptation over time.

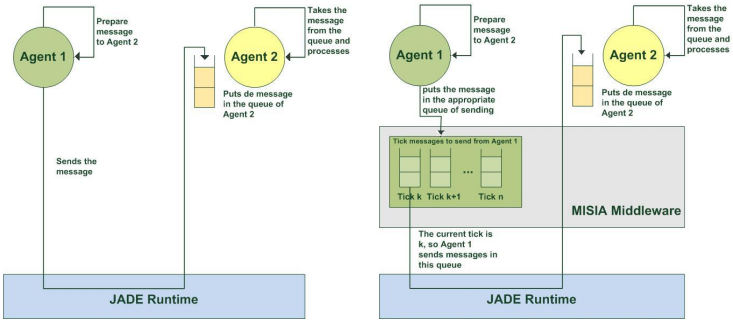

Fig. 2 (a) Sending a message in JADE. (b) Sending a message incorporating MISIA

In JADE, sending a message is instant, the programmer gives the order, and the message is sent ipso facto. MISIAAgent agent does not it in this way. It has a messaging queue MISIATickACLMessage, so that when a programmer gives the order to send a message, the message is automatically inserted in the scheduled queue with the tick which it must be sent. Thus, when a tick goes by, MISIAAgent immediately sends messages that are queued labeled with the current tick. Subsequently, it adds to the agent the protocols needed in this tick. This idea will be further detailed in the following subsection. After performing this task, executes a method designed to make the final programmer overwritten. The aim of this method is to have a function similar to *step ()* Repast method in JADE, which is

automatically executed when a tick went by. This function is very convenient to the developers, because, being called on every tick, it is possible separate the actions of the agent in order to carry out them on a specific tick. Finally, after performing all tasks, notifies its agent Repast counterpart, MISIARepastAgent, to make the necessary changes in the simulation environment. Once all MISIARepastAgent agents have been updated in the context of the simulation, this tick went by, and Repast proceed with the change to the next tick.

3.2 Redefinition of FIPA Protocols

JADE has a number of implemented FIPA protocols, which help the programmer. With these classes, it abstracts the developer from having to prepare messages to be sent, sending, or to manage the reception of them, among other things. In this framework has been re-implemented FIPA protocols defined in JADE to support the notion of time. These communication protocols JADE defines two roles, which starts the conversation (Initiator role) and which is envolved in the conversation (Responder role). The Initiator agent role will begin by the conversation by sending a message to the recipient. Therefore, it follows the logic developed with the message queue. When a MISIAAgent agent wishes to follow a communication protocol in a given tick, just add the protocol of communication to the agent in the tick established. Therefore, one of the functions of MISIAAgent agent after receiving a tick is to add communication protocols. The rest of communication for sending and receiving messages is re-implementing, recording different behaviors that make the different functions of the protocols. The novelty is that these new behaviors support MISIA modules redefined for JADE , such as support MISIATickACLMessage messages or the ability to respond to a message in a certain tick, without being immediately.

An example reimplemented is the FIPA-Request protocol, which is like follows: the agent with Initiator role sends a request to agent with Responder role. Responder replies, accepting or rejecting the request, and immediately returns to answer the agent with Initiator role informing the result (if the request was made correctly, or there was a problem). With the new definition by MISIA of this protocol, it is possible to send messages during the tick chosen. In this case, MISIA only redefines the role Responder. The Initiator is not necessary because it only sends a message to the beginning.

In the case of the Responder role, must send two messages, as discussed above. So, MISIA provides to programmers two handles, like JADE; one to send the first message, and another to send the second one, abstracting from all the system logic that is to managing ticks.

Below is a fragment of code in Java where it shown how a behavior is reimplemented to manage the arrival of the request by the agent with Initiator role. In this example, *handleMISIARequest* is the procedure that the final developer overwrites to provide the message he want to send in response.

```
registerPrepareResponse(new OneShotBehaviour(){
   public void action() {
          //Get DataStore to obtain the request message
      DataStore ds = getDataStore();
      ACLMessage requestMessage = (ACLMessage) ds.get(REQUEST_KEY);
      TickACLMessage agreeMessage = null;
      try {  agreeMessage = handleMISIARequest(requestMessage);
       } catch (Exception e) {}
    //If the message isn't null, send
    if (agreeMessage != null)     jadeAgent.MISIASend(agreeMessage);   }
```

4 Experimental Results

It has been developed a case study using this middleware to create a multiagent system aimed at facilitating the employment of people with disabilities, so it is possible to simulate the behavior of the agents in the work environment and observe the agents actions graphically in Repast. This is a simple example that defines four jobs, which are occupied by four people with certain disabilities. Every job is composed of a series of tasks. Agents representing the workers have to do them, and according to their capabilities, carry out the assignment with varying degrees of success. Performing various simulations, and seeing the evolution in time, the results can be assessed to determine what would be the most suitable job for each employee. In addition, taking into account the capabilities of Repast, it is possible to make a collection of data generated from each simulation and exported to external applications such as MatLab, for different studies on them. Below is an example of the execution of this case study.There are two ways for visualizing multiagent systems simulation: the agents interaction protocol and the agent entity. MISIA provides the capabilities visualize the sequence of messages between agents and the entity agent and its iteration with the environment. The union of these two platforms involves having a highly efficient environment for the creation of multiagent systems, getting the benefits of JADE to create the systems, as is the use of FIPA standards; and also the visual representation and extraction of simulation data to different applications provided by Repast.

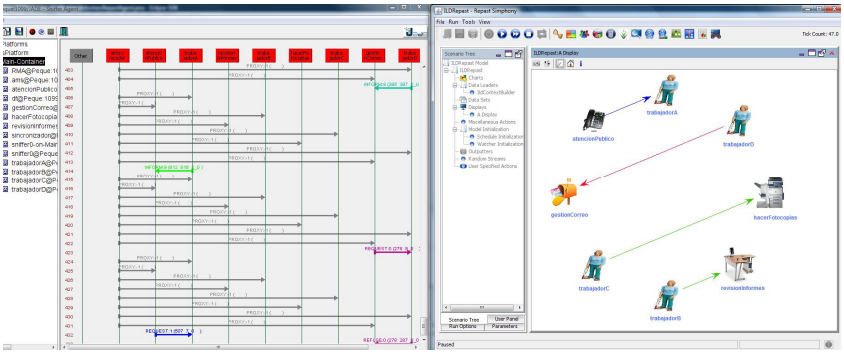

Fig. 3 Case Study MISIA

4 Conclusions and Future Works

Simulation is a helpful tool for understanding complex problems. Therefore, the simulation of multiagent systems in several levels of details and the emergent behavior is fundamental for analyzing the systems processes. In this study, a list of basic concepts and advances is presented for the development of simulated multiagent systems. We showed in detail the visualization and simulation infrastructure for developing the MAS behavior simulators. MISIA allows simulation, visualization and analysis of the behavior of agents. With the MAS behavior simulator it is possible to visualize the emergent phenomenon that arises from the agents' interactions.

The proposed visualization system also suggests further developments. One of them is make the agent representation more photo-realistic. A 3D agent visualization in more levels of details showing the interaction them would make the system complete and realistic. Another future work is to improve interactivity with the user. We would like to allow the user to visualize the agent state and its simulation individually. We also want to improve the interactivity by means of allowing the interaction of the specialists with the live execution besides the basic functionalities such as play, pause, stop and increase/decrease the speed, by means of putting some substances in the position and observing the emergent behavior. It would allow the self-organization optimization and the proposal of new hypotheses. Even more: generation of reports about the information visualized during the simulation process in several levels of detail, which could increase the comprehension about the process. MISIA is the ideal framework for this purpose.

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

References

[1] Agent Oriented Software, Ltd., ACK Intelligent Agents-Agent Practicals, 4 (2004)
[2] Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: Jade a white paper. EXP in Search of Innovation 3(3), 06–19 (2003)
[3] Collis, J.C., Ndumu, D.T., Nwana, H.S., Lee, L.C.: The zeus agent building tool-kit. BT Technol Journal 16(3) (1998)
[4] Corchado, E., Pellicer, M.A., Borrajo, M.L.: A MLHL Based Method to an Agent-Based Architecture. International Journal of Computer Mathematics 86(10, 11), 1760–1768 (2008)
[5] Davidsson, P.: Multi agent based simulation: Beyond social simulation. In: Moss, S., Davidsson, P. (eds.) MABS 2000. LNCS (LNAI), vol. 1979, pp. 97–107. Springer, Heidelberg (2001)
[6] Luke, S., Cioffi-Revilla, C., Panait, L., Mason, S.K.: A new multiagent simulation toolkit. In: Proceedings of the 2004 SwarmFest Workshop (2004)
[7] North, M.J., Howe, T.R., Collier, N.T., Vos, J.R.: The repast symphony runtime system. In: Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms (2005)

[8] North, M.J., Collier Nicholson, T., Vos Jerry, R.: Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit. ACM Transactions on Modeling and Computer Simulation 16(1), 1–25 (2006)
[9] JADE, Java Agent Development Platform, http://JADE.tilab.com
[10] Repast, http://repast.sourceforge.net/repast_3/index.html
[11] Shendarkar, A., Vasudevan, K., Lee, S., Son, Y.-J.: Crowd Simulation for Emergency Response using BDI Agent based on Virtual Reality. In: Proceedings of the 2006 Winter Simulation Conference, pp. 545–553 (2006)
[12] Swarm, http://www.swarm.org
[13] Foundation for Inteligent Physical Agents. FIPA Agent Management Specification. Disponible en,
http://www.fipa.org/specs/fipa00001/SC00001L.html
[14] Vizzari, G., Pizzi, G., da Silva, F.S.C.: A framework for execution and visualization of situated agents based virtual environments. In: Workshop Dagli Oggetti Agli Agenti, pp. 22–25 (2007)
[15] Wooldridge, M., Jennings, N.R.: Agent Theories, Architectures, and Languages: a Survey. In: Wooldridge, M.J., Jennings, N.R. (eds.) ECAI 1994 and ATAL 1994. LNCS, vol. 890, pp. 1–22. Springer, Heidelberg (1995)
[16] Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology. ACM Transactions on Software Engineering and Methodology 12, 317–370 (2003)