Journal of Intelligent & Fuzzy Systems 28 (2015) 29–41
DOI:10.3233/IFS-141219
IOS Press

Infrastructure to simulate intelligent agents in cloud environments

Fernando De la Prieta[a,*], Sara Rodríguez[a], Juan M. Corchado[a] and Javier Bajo[b]
[a]*Computers and Automation Control Department, Faculty of Science, University of Salamanca, Plaza de la Merced S/N, Salamanca, Spain*
[b]*Artificial Intelligent Department, Faculty of Computer Science, Technical University of Salamanca, Boadilla del Monte, Madrid, Spain*

Abstract. Nowadays there is a clear trend towards using methods and tools that can help the development of Simulation Systems. In this regard, Multiagent System (MAS) is a key technology that allows simulating user behavior. When the user is the focus of the simulation, Ambient Intelligent (AmI) increases in importance. AmI is an emerging multidisciplinary area based on ubiquitous computing that aims to adapt the environment to the needs of the user. Moreover, Cloud Computing is revolutionizing the services provided through the Internet, continually adapting itself in order to maintain the quality of its services. This study presents a multiagent based middleware for the simulation of agent behavior in cloud environments, and uses this information to adapt to the environment as AmI proposed. The main challenge of this work is the design and development of a new middleware that makes possible the communication between the technology in charge of the development of MAS and the technology in charge of the simulation, visualization and analysis of the behavior of the agents using the potential of cloud computing. The platform makes the computation and communication possible by following the principles proposed by AmI.

Keywords: Multiagent systems, simulation, agent based simulation, cloud computing, distributed systems

1. Introduction

Many new technical systems are distributed systems involving complex interaction between humans and machines, which notably reduces their usability. Ambient Intelligence (AmI) is an emerging multidisciplinary area based on ubiquitous computing. It influences the design of protocols, communications, systems, devices and so on, proposing new ways of interaction between people and technology, adapting them to the needs of individuals and their environment. Nowadays, these systems are evolving into a new paradigm known as Cloud Computing, which has emerged as a key component of the Future Internet. The key advantage of this new Computing framework lies in the provision of the services: it tries to reduce user requirements and provide a dynamic resource management, where the resources increase only when there is a rise in service demand [20, 21]. An environment based on cloud computing must readjust its resources by taking the demand of the provided services into account.

Cloud Computing and AmI are complementary technologies. At a functional level, AmI focuses on adapting the environment to the user. However the underlying technological level on which it is based is the computing and communication services. These computing needs provide a Cloud Computing environment, in addition to the demand for user needs. The Cloud Computing environment provides these computational services.

Simulation can be defined as the representation of the operation or features of one process or system through

*Corresponding author. Fernando De la Prieta, Computers and Automation Control Department, Faculty of Science, University of Salamanca, Plaza de la Merced S/N, 37008 Salamanca, Spain. Tel.: +34 923 294 400/Ext. 1525; Fax: +34 923 294 514; E-mail: fer@usal.es.

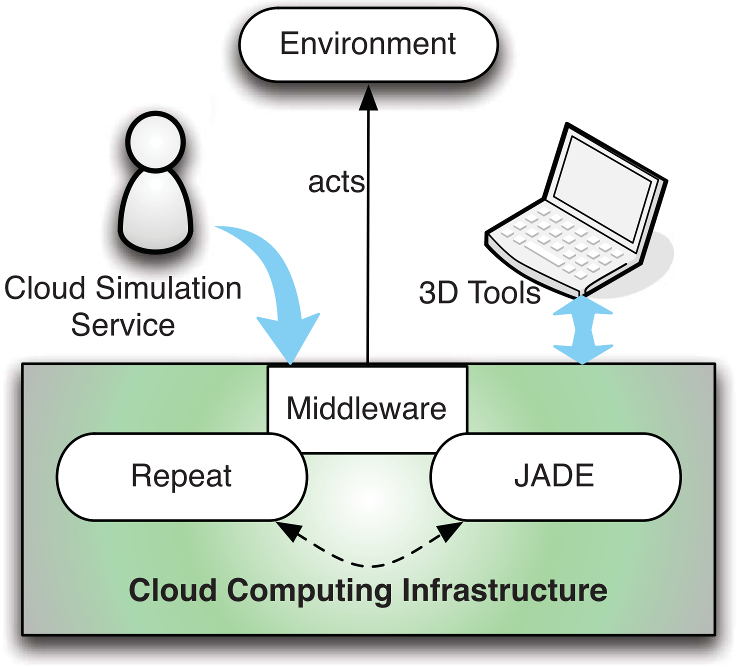

Fig. 1. Simulation middleware overview.

the use of another [16]. Current research lines that attempt to determine which technique is best for different purposes of simulation are gaining importance. 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, including humans. The benefits of agent based computing for computer simulation include various methods for evaluating multi agent systems or for training future users of the system [16]. The properties of ABS make it especially suitable for simulating this kind of system. The idea is to model the behavior of the human users in terms of software agents. Within this context, MAS are adequate for developing applications in dynamic and flexible environments. This is primarily due to their capabilities, which can be modeled in different ways and with different tools [46]. Moreover, the use of both Cloud Computing and AmI technologies in the context of user behavior simulation is an advantage because it makes it possible to anticipate the reallocation of the resources.

Open MAS should allow the participation of heterogeneous agents with different architectures and even different languages [13, 47]. However, it is necessary to define new solutions that allow the connection on ABS simulation software.

This study describes the results achieved by a multiagent-based middleware for the simulation of agent behavior. The middleware allows the simulation, visualization and analysis of the agents' behavior. The main contribution of this paper is the design of a new infrastructure that makes it possible to provide these capabilities, see Figure 1. The simulation middleware makes use of technologies for the development of widely used MAS, and combines them so that it is possible to use their capabilities to build highly complex and dynamic systems. This includes, for example, JADE [23] that is the most widely used platform for software agents middleware, and Repast (Recursive Porous Agent Simulation Toolkit) [34], a free and open-source agent-based modeling and simulation toolkit. This middleware and its dependent component were deployed into a Cloud Infrastructure in order to provide the system as a cloud service.

The next contribution is the reformulation of the FIPA protocol used in JADE [23] This has achieved several advantages: (i) development of a new framework that provides independence between the model and visualization components; (ii) improvement of the visualization component that makes it possible to use the concept of "time", essential for the simulation and analysis of the behavior of agents; (iii) and improvements to the user capabilities to which several tools were added, including message visualization. A 3D tool was developed in order to provide a graphical representation of the information provided by the simulation through the visualization messages. This tool provides an interface to model buildings and visualization of the simulation.

Smart environments must take context information into account, which can be collected by sensor networks [4]. To do so, the system is complemented with a module that permits the actuation in the environment, adapting itself to the needs of the end user. In this regard, the simulation makes it possible to know the user's needs in advance.

This study simulates a work environment where there are disabled people with special needs in their workstation. As a result of the information provided by the simulation, user needs can be prepared in advance so that the disabled people do not have to wait to work.

All contributions resulted in what is, to the best of our knowledge, the first middleware infrastructure to simulate intelligent agents with 3D visualization, simulation and analysis capabilities in a cloud platform.

The article is structured as follows: Section 2 reviews the state of art of agent-based simulation and cloud computing technology. Section 3 introduces a description of the infrastructure specifically adapted to the simulation of multiagent systems within dynamic environments. Finally, some results and conclusions are given in Sections 4.

2. Background

2.1. *Agent-based simulation*

The main characteristics of agents and MAS are autonomy, learning and reasoning. These capabilities can be implemented with different tools [46]. Open MAS should allow the participation of heterogeneous agents with different architectures and even different languages [13, 47]. The development of open MAS is still a recent field of the MAS paradigm and its development will allow applying the agent technology in new and more complex application domains. There are existing studies on agent simulation using commercial off-the-shelf-simulation packages with built-in agent-based modeling and BDI (Belief-Desire-Intention) behavior architecture [38], modeling detailed complex human behaviors.

There are mainly two ways to visualize MAS simulation: the agent's interaction protocol, and the agent entity. With the former, it is possible to visualize a sequence of messages between agents and the constraints on the content of those messages. The latter method visualizes the entity agent and its iteration with the environment. Most software programs, such as JADE platform [6, 23] and Zeus toolkit [12], provide graphical tools that allow the visualization of the messages exchanged between agents.

The MASON [27], Repast [31, 34] and Swarm [40] toolkits provide the visualization of the entity agent and its interaction with the environment. Swarm [40] is a library of object-oriented classes that implements the conceptual framework of Swarm for agent-based models, and provides many tools for implementing, observing, and conducting experiments on ABS. MASON [27] 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. Mason has a distributed version (DMASON). Finally, Repast [34] is a free and open-source agent-based modelling and simulation toolkit. Nowadays, there are also distributed versions of these simulations frameworks: DMASON [18], Repast HPC [1], AGLOBE [2] and so on.

There are also other studies such as Vizzari et al. [43] which have 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 it would add a non-desired complexity to our system. We chose the Repast toolkit because, when the project first started, it was one of the few to offer a 3D visualization feature, as well as being simple and having good documentation. Moreover, the Repast system, which includes the source code, 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 [30].

2.2. *Cloud computing paradigm*

The technology industry is presently making great strides in the development of the Cloud Computing paradigm. As a result, the number of both closed and open source platforms has been rapidly increasing.

They all have a similar architecture. From an external point of view, the three most widely known services are Software, Platform and Infrastructure. From an internal point of view, the services generally offered are considered elastic services due to the high number of underlying technologies (virtualization, server farms, web services, web portals, among others) which have reached their prime.

NIST [29] propose the following definition, which comes very close to what is understood today by Cloud Computing: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

While at first sight it may be considered a merely technological paradigm, reality demonstrates that its rapid progress is primarily motivated by the economic interests that surround its purely computational or technological characteristics. This type of platform will allow Cloud Computing providers to offer computational services following a model comparable to the concept proposed by Utility Computing [36] in that computational services are offered in a way similar to how traditional public utilities (e.g. electricity, telephone, water, etc.) are offered. As a result, the service providers will be able to offer their demanded services through the Internet with a pay-per-use model [9, 19]. Cloud service users, on the other hand, will not be required to adopt additional computational resources to handle peak demand of their products or services, nor will they be required to have the infrastructure needed to provide them, thus turning their capital expenses into operational expenses [5].

There are many cloud providers such as VMWare [44], Citrix [11], Amazon [3] and so on. Furthermore, from an academic and research scope, Cordeiro et al. [15] propose the Euronet platform to interconnect multiple virtual laboratories using a Cloud environment. Nurmin et al. [32] present EUCALYPTUS, an open source framework for cloud computing that implements IaaS functions. Another proposal is provided in [28], which presents a 3C model (Cooperative Cloud Computing) for research centers and universities. This model is based on the Virtual Cloud model and expects to generate a vast repository of computational resources for research centers. Given that the majority of Cloud platforms are proprietary, the underlying infrastructure is invisible to researchers.

2.2.1. +Cloud platform

For the deployment of the simulation middleware and its associated software components, our own cloud-based platform, +Cloud [22], was used. +Cloud is a platform based on the Cloud Computing paradigm. This platform allows services to be offered at the PaaS (Platform as a Service) and SaaS (Software as a Service) levels. The SaaS services are offered to end users in terms of web applications, while the PaaS services are offered as web services. Both PaaS and SaaS layers are deployed using an internal layer, which provides a virtual hosting service with automatic scaling and functions for balancing workload. Therefore, this platform does not offer an IaaS (Infrastructure as a Service) layer.

This platform is multipurpose and allows the deployment not only of components specially developed for cloud environment, but also of software components that are not originally designed to use within a cloud infrastructure, such as with our case. As a result, it is possible to take advantage of Cloud computing and, at the same time, to continue using the traditional server-side components.

The internal layer of +Cloud is formed by the physical environment which allows the abstraction of resources shaped as virtual machines. Thus, since this layer is internal to the system, +Cloud does not offer this kind of service to its users. A more detailed description of each layer is provided below:

– SaaS Layer. This layer hosts a wide set of Cloud applications. It offers a set of native applications to manage the complete Cloud environment, where users have a personalized virtual desktop from which they can access their applications in the Cloud environment, and other more general third party applications.
– PaaS Layer. This layer is oriented to offer services to the upper layer, and is supported by the internal infrastructure layer. The PaaS layer provides services through RESTful web services [33] in an API format. The components of this layer are: (i) the IdentityManager, which is the module in charge of offering authentication services to clients and applications; (ii) the File Storage Service (FSS), which provides an interface for a container of files, emulating a directory structure in which the files are stored; and, finally, (iii) the Object Storage Service (OSS), which provides a

simple and flexible schema-less database service oriented towards documents.

The virtual and physical resources are managed dynamically. To this end, a virtual organization of intelligent agents that monitor and manage the platform resources is used [22]. In +Cloud, the elastic management of the available resources is performed by a MAS based on VO, which eases the design of the overall system [20]. In the +Cloud VO there is a set of agents that are especially involved in the adaptation of the system resources in view of the changing demand of the offered services. In addition, one of the most innovative aspects of the +Cloud framework is its ability to provide the possibility of reaching agreements at two different levels: at the external level, whereby +Cloud allows temporal SLA's parameter negotiation between the system and its customers using multiagent interaction protocols; and at the internal level, whereby +Cloud includes a computational case-based argumentation framework that agents can use to engage in an agreement process to make a decision about the best resource redistribution to meet a SLA during peak service demand.

2.3. Agent-based cloud computing and simulations

There are only a limited number of studies on the state of the art that relate Cloud Computing with agent technology [41]. In general terms, a Cloud system may use MAS applications in a Cloud environment for deployment; there are also Cloud environments that use agent technology to manage their resources. Some of those applications include:

– Agents using Cloud. Within this group, the main state of the art applications use computational resources from the Cloud environment.
– Cloud using Agents. Within this subgroup, the range of possibilities is even further extended. Mong Sim [39] highlights three subgroups of applications: (i) combination of resources among Cloud providers; (ii) planning and coordination of shared resources; (iii) establishing contracts between users and Cloud service providers.

Obviously, a simulation using Cloud and MAS would correspond to the first application, Agents using Cloud. The research in this group is based on the use of computation resources (storage and computation) that the Cloud environment provides. With regards to the computational capacity of the cloud, there are more examples [10, 17] that use the computational strength of the environment to perform simulations in different fields. All of them use the computing environment as a grid environment [8].

The approach of this paper is different; the simulation not only uses the computational resources of the cloud platform, but also offers a specific cloud service of simulation to the end users. It is also important to note that the simulation does not use any kind of algorithm to distribute the complexity of the simulation over the cloud (such as previous examples that use a Grid computing approach). In the case of this paper, the middleware only grants control to the +Cloud platform that is in charge of providing the resources depending on the needs of the simulation.

2.4. Ambient intelligent

Ambient Intelligence offers a great potential to improve quality of life and to simplify the use of technology by offering a wider range of personalized services and providing users with easier and more efficient ways to communicate and interact with other people and systems [14].

The characteristics of the agents make them appropriate for developing dynamic and distributed systems based on AmI, as they possess the capability of adapting themselves to the users and environmental characteristics [24].

The integration and interoperability of agents and MAS with SOA. Web services approaches has been explored in order to provide an intelligent environment. Some developments are centered on communication between these models, while others are centered on the integration of distributed services into the structure of the agents [7, 25, 26, 35, 37, 45].

Cloud Computing facilitates integration due to the ease of use of services that it provides, besides the large range of them (communication, computational service and simulation) and the simulation makes possible to advance the adaption of the system to the user needs.

3. Infrastructure to simulate environments

The most well-known agent platforms (like Jade [23]) 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, such as simulation constraints imposed by the MAS topology, by themselves.

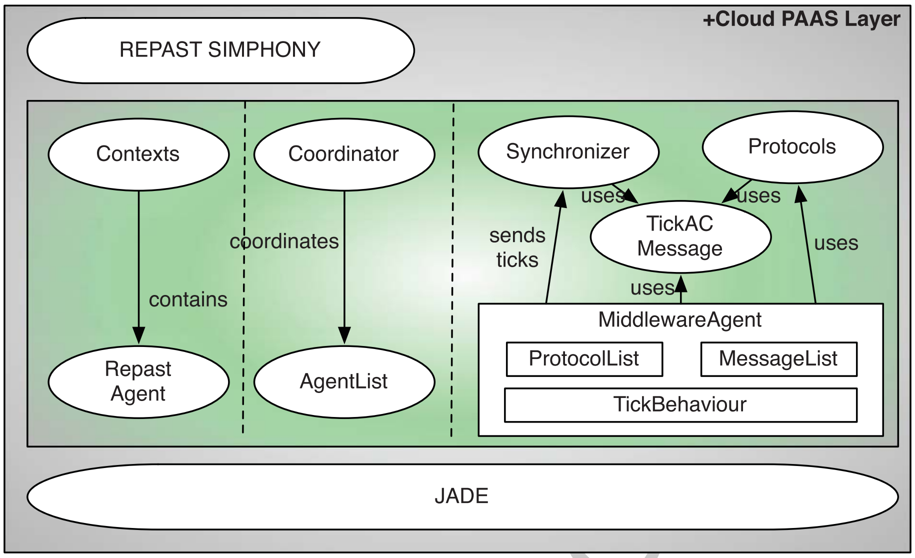

Fig. 2. Middleware deployed in +Cloud Platform.

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. To this end, this study presents a middleware infrastructure that allows modeling JADE MAS with the possibility of being represented in Repast.

In order to better understand the overall operation, it is necessary to bear in mind the following idea: *it is necessary to synchronize JADE to work simultaneously to Repast.* The main concept introduced in this environment is the notion of time in JADE, which means it is possible to render the events into Repast in real time. One of the main differences between JADE and Repast is that in JADE, there does not exist a concept of time as such; the agents interact with 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. Agents in the JADE context are implemented based on FIPA standards. This allows creating MAS in open environments, which is not possible within Repast. These differences are what this middleware has been able to resolve, integrating the two environments and achieving a working environment for the creation and simulation of a more powerful and versatile MAS and Middleware components.

Furthermore, the deployment of this new middleware and its related components in a Cloud platform makes it easier to underlay the infrastructure management. With this approach, the responsibilities are clearly defined; the middleware only has to manage the simulation process, while the cloud takes care of the computational resources.

The middleware consists of three principle components or layers as shown in Fig. 2:

– **The upper layer is the contact with Repast**. It contains two functional blocks, which are:
  - *RepastAgent*. For every agent that we want to have on the system we must actually create two: a MiddlewareAgent agent running on JADE, and its respective RepastAgent released on Repast. It can be seen as follows: a logical agent, and two physical agents. RepastAgents have an important role: they cannot update their status until their respective MiddlewareAgent has finished with all the work they need to perform during that unit time (tick). This is a very important aspect, since it is the characteristic of the framework as a system in real time.
  - *Context.* It has two important objectives. One is to establish the synchronism in the execution. When a tick goes by, it lets the Synchronizer agent know that it is necessary to notify Middleware Agent agents that the tick has ocurred. The other goal of this module is to incorporate new RepastAgent that enter in the context of the Repast simulation. For each new Middle-

wareAgent that appears in the system, Context will create its respective RepastAgent and add it to the simulation environment.

– **The bottom layer, which enables JADE, supports the notion of time**. It is divided into four functional blocks:
  • *Middleware Agent*. It is the extension of the 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.
  • *TickACLMessage*. JADE messages are used for communication between agents. MiddlewareAgent agents communicate among themselves with TickACLMessage messages. TickACLMessage is the extension of the JADE ACL message, incorporating the concept of time. It includes aspects such as the tick identifying where to send the message, and the delay that the message has when it has reached its destination. In JADE, the messages exchanged between agents are sent and arrive instantly; however, that is not the case in real life. JADE aims to simulate and view the evaluation of the system as time passes. To achieve this, messages cannot be instant, but must have a shipping time and a different reception time.
  • *FIPA Protocols*. 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 detailedin a subsequent section.
  • *Synchronizer*. This is a JADE agent that acts as notifier. It is responsible for notifying the Middleware Agent when a tick goes by. It is the system clock synchronization. When a tick goes by, Synchronizer is notified in order to notify MiddlewareAgents. Messages with a special performance or function are made through TickACLMessage.

– **The intermediate layer, whose main goal is to interconnect the two platforms (JADE and Repast)**. It is divided into two functional blocks, and its goal is to join adjacent layers. These modules are:
  • *AgentList*, as its name implies, it stores all agents in the system at a given time. It plays an important role because it enables communication between a MiddlewareAgent and its respective RepastAgent, and vice versa. The diagram shows two-way information flows from RepastAgent to AgentList, and from AgentList to MiddlewareAgent. These flows represent the communication between two physical agents.
  • *Coordinator* coordinates communication between the two adjacent layers. The presence of a coordinator to maintain synchronism between both layers is fundamentally necessary. Because of Coordinator, Context can notify the occurrence of a tick to Synchronizer, and Synchronizer can ensure Context that its purpose has been carried out, reporting that all MiddlewareAgent have received the tick. This kind of communication is necessary to maintain full synchronization between the two platforms. Also shown in the diagram are two flows: between Context and Coordinator, and between Coordinator and Synchronizer; they represent the flow of ticks and the synchronism.

3.1. Notion of time in JADE

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, as it is shown in Figure 3:

– The occurrence of a tick is generated on Repast, as it is the platform that has this ability. Context is notified of this and, through Coordinator, can notify Synchronizer of its tasks.
– When Synchronizer receives the notice from Context, it alerts all Agents about the new tick. As Synchronizer is an agent, it can communicate with MiddlewareAgent through TickACLMessage messages. That is how it notifies them about the new tick, sending a message with special semantics to each agent.
– Once Synchronizer has sent all the special messages notifications of tick, it must wait until all MiddlewareAgents answer, to ensure that all have received

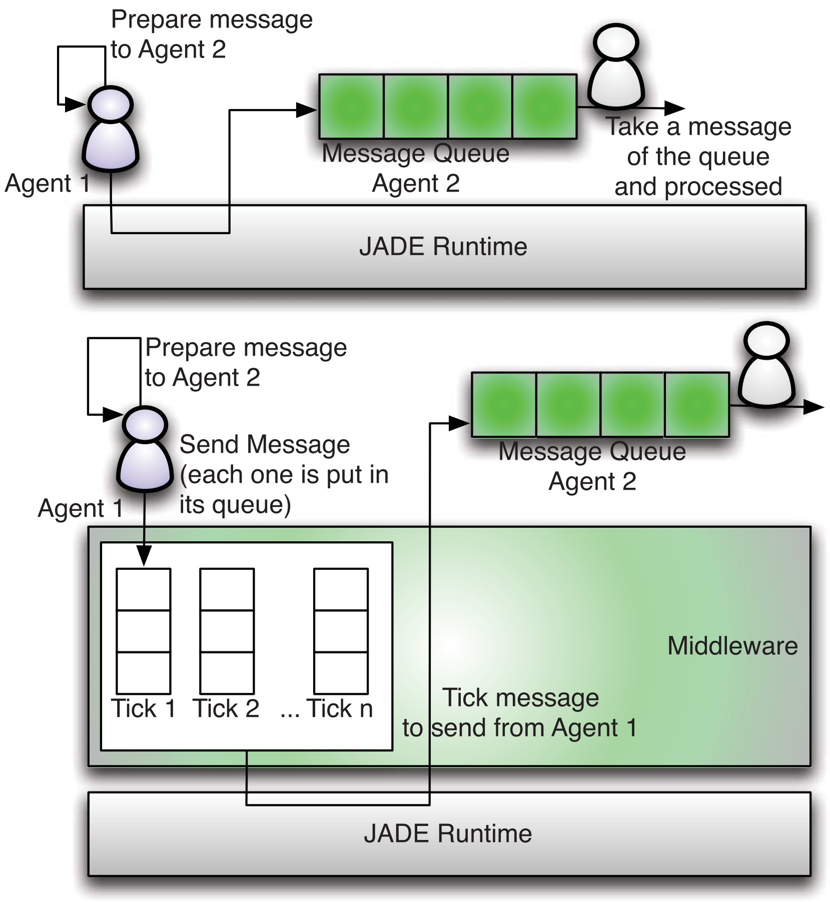

Fig. 3. Sending a message in JADE and with the framework.

the notification. Moreover, it sends an ACK message to maintain strong synchronism between both sides.

– When a MiddlewareAgent receives a tick by Synchronizer, it carries out various actions. First, it sends the message that it has to ship in this tick. MiddlewareAgents have a special queue for sending messages. Below is a comparative picture of how shipping is performed in JADE, and what change was made in the framework for adaptation over time.

In JADE, sending a message is instant, the programmer gives the order, and the message is sent ipso facto. MiddlewareAgent agent functions differently. It has a messaging queue, TickACLMessage, so that when a programmer gives the order to send a message, the message is automatically inserted in the scheduled queue with the associated(?)tick, which must also be sent. Thus, when a tick goes by, MiddlewareAgent immediately sends messages that are queued and 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, MiddlewareAgent executes a method designed to overwrite the final program. The aim of this method is to have a function similar to *step ()* of the Repast method in JADE, which is automatically executed when a tick goes by. This function is very convenient for developers, because by calling on every tick, it is possible to separate the actions of the agent in order to carry out them on a specific tick. Finally, after performing all tasks, MiddlewareAgent notifies its agent Repast counterpart, RepastAgent, to make the necessary changes in the simulation environment. Once all RepastAgent agents have been updated in the context of the simulation, the tick goes by, and Repast proceeds with the change to the next tick.

Table 1
Relation between AchieveREResponder (Jade) – MiddlewareAchieveREResponder

Jade	Midleware
AchieveREResponder (class)	MiddlewareAchieveREResponder (class)
protected ACLMessage handleRequest(ACLMessage request) (AchieveREResponder method)	protected MisiaTickACLMessage manejarPeticionRequest (ACLMessage requestMessage) (MiddlewareAchieveREResponder method)
protected ACLMessage prepareResultNotification(ACLMessage request, ACLMessage response) (AchieveREResponder method)	protected MisiaTickACLMessage manejarResultadoPeticionRequest (ACLMessage requestMessage, ACLMessage responseMessage) (MiddlewareAchieveREResponder method)
NotUnderstoodException (class)	MiddlewareNotUnderstoodException (class)
RefuseException (class)	MiddlewareRefuseException (class)
FailureException (class)	MiddlewareFailureException (class)

3.2. *Redefinition of FIPA protocols*

JADE has a number of implemented FIPA protocols, which help the programmer. With these protocols, it encapsulates the developer from having to prepare messages to be sent, sending the messages, or managing their reception, among other things. The FIPA protocols defined in JADE have been re-implemented in this framework to support the notion of time.

It is possible to observe the presence of two roles in the FIPA protocols implemented in Jade [23]: *Initiator* and *Responder* or *Participant.* Jade provides a predefined class for each role and each type of FIPA interaction protocol, or rather, for a certain group of FIPA protocols. The *jade.proto* package contains all the classes that, in the form of behaviors, facilitate the implementation of the FIPA communication protocols. Each pair of classes is told to implement a series of protocols. The middleware aims to adapt all these classes to their environment, so that an end user can use them as in Jade, without worrying about the presence of time. For example, with the first pair adapted (*AchieveREInitiator* and *AchieveREResponder*), it is possible to implement *FIPA-Request, FIPA-Query, FIPA-Recruiting, FIPA-Request-When* y *FIPA-Brokering* protocols.

To implement any of these protocols in the Middleware, it is necessary to use *AchieveREInitiator* (Jade class) and *MiddlewareAchieveREResponder*, the adapted class of the *Responder* role. *MiddlewareAchieveREResponder* is intended to replace *AchieveREResponder* (Jade class). It provides two handling methods, *manageRequest,* to send the first message in response, and *manageResultRequestRequest,* to send a second message to the agent with the Initiator role. In addition, it implements the exceptions, to try to provide the same Jade interface. These exceptions are important because Jade uses them to send messages of the rejection or not understanding of a task (i.e. if *Responder* role sends a message of acceptation for a task, the execution flow does not diverge in exception).

The messages of *refuse, failure* and *notUnderstood* will diverge in exceptions, which are also adapted to the notion of time to send these messages in the desired tick. The equivalence between Jade classes (and methods) and the middleware is shown in Table 1.

The communication protocols of JADE defines two roles, one which starts the conversation (Initiator role), and the other which evolves in the conversation (Responder role). The Initiator agent role will begin the conversation by sending a message to the recipient. In doing so, it follows the logic developed with the message queue. When a MiddlewareAgent agent wishes to follow a communication protocol in a given tick, it is simply necessary to add the protocol of communication to the agent in the tick established. Therefore, one of the functions of MiddlewareAgent agent after receiving a tick is to add communication protocols. The rest of the communication for sending and receiving messages involves re-implementing and recording different behaviors that make the different functions of the protocols. The novelty is that these new behaviors support middleware modules redefined for JADE, such as support TickACLMessage messages or the ability to respond to a message in a certain tick, without being immediately.

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

3.3. *Deploying infrastructure in a Cloud*

Both the middleware (including JADE and Repast) have been deployed in +Cloud system. The simulation environment is offered as a service in the Cloud, so

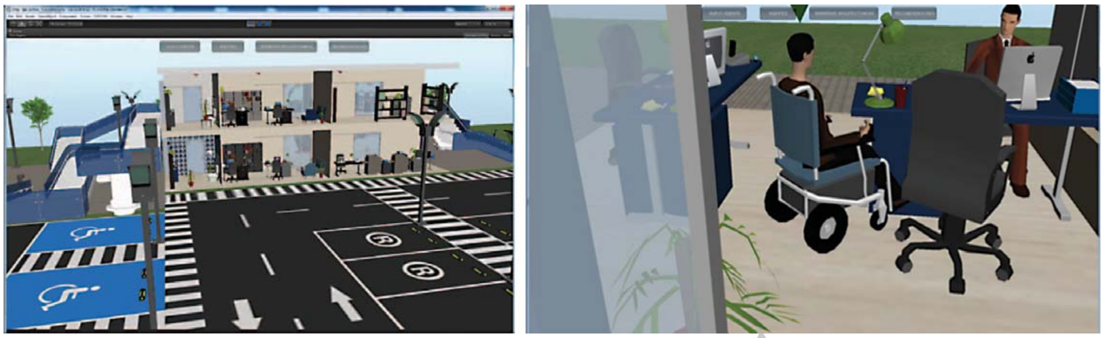

Fig. 4. Simulation in a 3D environment.

that end users can configure their own simulations, and access a 3D simulation environment that will get the results of the simulation.

The main components of the middleware have been deployed in the infrastructure layer of the cloud platform. These are:

The JADE platform is deployed as a service within the cloud. It is not possible to create a highly available service from the basic platform. However, thanks to the +Cloud platform, it is possible to deploy non-cloud services in the platform.

Repast, as with JADE, is deployed as a service. But the version used in the present study is Repast HPC (High Performance Computing), so the simulation service can use all the features offered in the Cloud infrastructure.

Finally, the middleware is deployed as an additional computing service, which interacts with JADE and Repast. This service provides high availability because its information is stored in the OSS and FSS services, provided by the +Cloud environment.

Moreover, within the software layer there is a set of applications deployed as services that provide a set of tools. For example, we can highlight the configuration and visualization environment, as well as a thin client that allows 3D modeling and simulation environments.

4. Experimental results and conclusions

A case study was developed using this middleware to create a MAS aimed at facilitating the employment of people with disabilities. Since it is possible to simulate the behavior of the agents in the work environment and observe the agents actions graphically in Repast, we propose a case study developing an intelligent environment for disabled people. This is a simple example that defines four jobs, which are occupied by four people with specific disabilities. Every job is composed of a series of tasks. Agents representing the workers have to carry out the jobs and, according to their capabilities, carry out the assignment with varying degrees of success. After 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.

The middleware is deployed into a +Cloud platform. This platform has several virtual nodes that provide computational resources to the middleware. +Cloud automatically allocates the resources to the components of the infrastructure taking instant demands into account during the simulation. As a result, the middleware does not need to take care of the computational needs.

Figure 4 shows two examples of the 3D simulation environment. It is possible to observe the entities representing worker agents through their 3D avatar. The scene has the option of hiding different components to facilitate visualization during a simulation.

Figure 5 shows a messaging visualization in Repast. Above each agent is its name, for identification. The red spheres correspond to workers. The cylinders correspond to the jobs of the workers, who are also actors in the simulation. The different colors indicate the type of job: dark blue represents an administrative assistant; yellow represents a customer service representative; green represents a workplace ordinance, the purple cones represent the agents belonging to the quality control department, and the orange cubes represent the agents belonging to the human resources department.

Figure 6 shows an example of the execution of this case study. There are two ways of visualizing MAS simulation: the agent interaction protocol, and the agent entity.

The middleware provides the capability to visualize the sequence of messages between agents and the entity

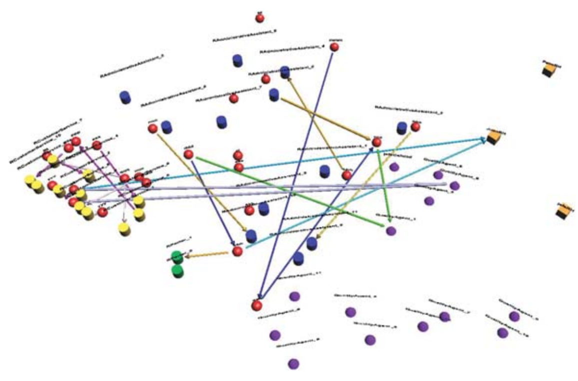

Fig. 5. Visualizaiton of messaging in Repast.

agent, and its iteration with the environment as shown in Fig. 6. The union of these two platforms requires, firstly, a highly efficient environment for the creation of MAS, incorporating the benefits of JADE to create the systems, as is the use of FIPA standards; and secondly, the visual representation and extraction of simulation data to different applications provided by Repast.

In this study we have presented a new infrastructure to facilitate the development of simulation and visualization environments. The platform is innovative, as it integrates multiagent systems and cloud computing technologies to take advantage of both the distributed and adaptive capacities of multiagent systems and the storage and elasticity capabilities of cloud computing environments. The use of multiagent systems demonstrates a high capacity and efficiency to develop virtual environments and to provide new alternatives for social simulation. The proposed approach provides the JADE platform with new time simulation algorithms, which make it very appropriated for agent-based simulation and prediction. Additionally, the integration with the +Cloud platform facilitates a flexible and adaptive repository that can be accessed in a ubiquitous manner and provides dynamic and ubiquitous computation.

The proposed approach provides an innovative framework to design, develop and evaluate intelligent environments. In this sense, the work contributes to improve the state of the art of Ambient Intelligence and to provide a new and effective approach for labor integration of people with disabilities.

The middleware was tested within the framework of two national Spanish projects to develop simulation environments aimed at facilitating the integration of handicapped people into the workforce. The results have shown that the proposed approach is appropriate for simulating intelligent environments using a social perspective. The new infrastructure allows the design and development of adaptive social environments as well as the simulation and prediction of potential situations in such environments. Domain experts and end users participated in the design of the infrastructures and in the experiments, and have remarked on the usability of the infrastructure, paying special attention to the flexibility to define virtual environments and to the prediction and analysis capabilities of the infrastructure.

The goal is to improve the interactivity by allowing specialists to interact with the live execution of basic functionalities such as play, pause, stop and

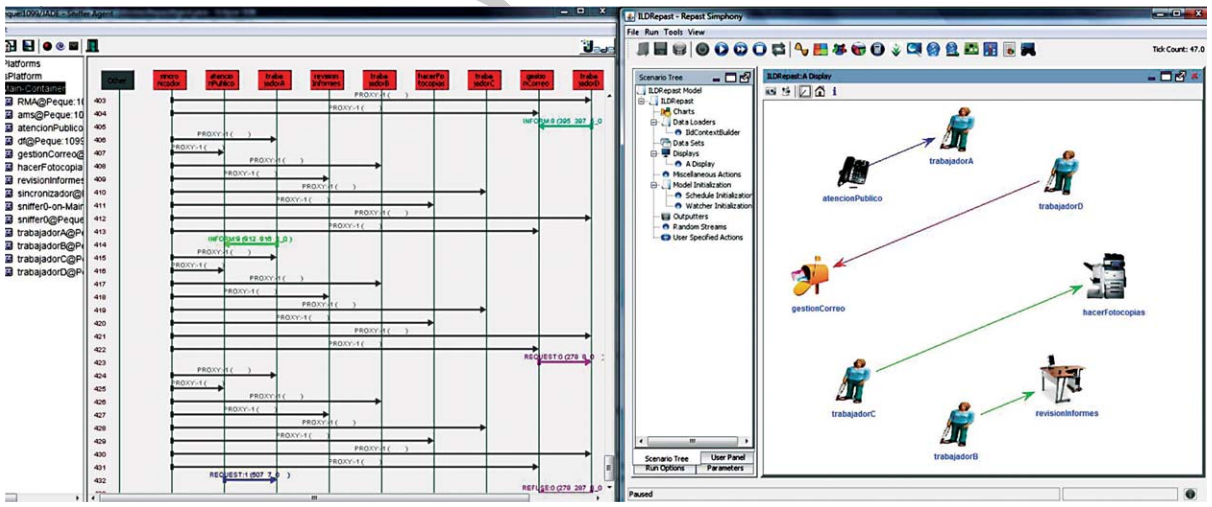

Fig. 6. Visualization of messages sequence in Jade and RepastDiscussion and future work.

increase/decrease the speed, by means of putting some substances in the position and observing the emergent behavior. It would optimize self-organization and the proposal of new hypotheses. Moreover, it would allow the generation of reports about the information visualized during the simulation process in several levels of detail, which could increase the comprehension of the process.

This simulation middleware was joined with a proximity detection system that is used by people with disabilities to facilitate their integration in the workplace [48]. When the proximity system detects an individual person, different actions can be performed on the computer in order to facilitate workplace integration; for example, automatically switching on/off the computer, identifying user profile, launching applications, and adapting the workplace to the specific needs of the user. Many of these tasks are time consuming because of the computation time required for the simulation system to predict the position where the worker is going to be. The system can advance these tasks in order to have the workplace ready on time for the worker. Finally, the main goal of the system is to detect the proximity of a person to a computer using Zigbee technology [48] and then activate the workplace instantly. More information about this information system can be seen here [42].

It is also important to note that, thanks to the underlying Cloud Computing environment, it is possible to be aware about the consumption of resources. For example, +Cloud resources can be optimized by displaying the graph of used, maximum available and cached memory. It is necessary to remark that this approach facilitates the use of cloud computing technologies to develop Ambient Intelligence environments in and easy and useful manner, which is very important given the growing relevance that cloud computing is acquiring in computational technologies.

Our further work focuses on testing the proposed infrastructure on a wide variety of environments and evaluating the obtained results with more domain experts and end users. This will help us to have a more realistic evaluation of the adaptability and usability of the proposed infrastructure. Furthermore, we are working on a more general multiagent architecture that incorporates adaptive mechanisms to develop open virtual organization-based environments. This way, we are exploring new communication protocols and new adaptation and re-organization algorithms. Finally, we are exploring new possibilities to improve the +cloud platform, such as advanced negotiation technologies. These are our next challenges.

Acknowledgments

This work has been partially supported by the MICINN project TIN2012-36586-C03-03.

References

[1] A High-Performance Agent-Based Modeling Platform. Repast HPC. http://repast.sourceforge.net/docs/api/hpc/. (Last access in 2013).

[2] A Globe simulation platform. http://exile.felk.cvut.cz/aglobe/ (Last access in 2013).

[3] Amazon S3. Amazon Simple Storage Service. http://aws.amazon.com/es/s3/ (Last Access in 2013).

[4] G. Anastasi, M. Conti, M. Di Francesco and A. Passarella, Energy conservation in wireless sensor networks: A survey, *Ad Hoc Networks* **7** (2009), 537–568.

[5] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, R. Ariel, I. Stoica and M.A. Zaharia, A view of cloud computing, *Communications of the ACM* **53**(4) (2010), 50–58.

[6] F. Bellifemine, G. Caire, A. Poggi and G. Rimassa, Jade a white paper, *EXP in search of innovation* **3**(3) (2003), 6–19.

[7] L.O. Bonino da Silva, F. Ramparany, P. Dockhorn, P. Vink, R. Etter and T. Broens, A service architecture for context awareness and reaction provisioning, *IEEE Congress on Services (Services 2007)* (2007), pp. 25–32.

[8] M.L. Bote-Lorenzo, Y.A. Dimitriadis and E. Gómez-Sánchez, Grid Characteristics and Uses: A Grid Definition, *Postproceedings of the First European Across Grids Conference (ACG'03), Springer Verlang. LNCS Volume* **2003** (2970), 291–298.

[9] R. Buyya, Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities, In: *10th IEEE International Conference on High Performance Computing and Communications*, HPCC '08, 2008, pp. 5–13.

[10] Y. Cheng, M.Y.H. Low, S. Zhou, W. Cai and C. Seng Choo, Evolving agent-based simulations in the clouds, In: *Third International Workshop on Advanced Computational Intelligence (IWACI'10)* (2010), pp. 244–249.

[11] Citrix Netscaler. http://www.cns-service.com/citrix/citrix-netscaler.aspx. (Last access in 2013).

[12] J.C. Collis, D.T. Ndumu, H.S. Nwana and L.C. Lee, The zeus agent building toolkit, *BT Technol Journal* **16**(3) (2008), 60–68.

[13] E. Corchado, M.A. Pellicer and M.L. Borrajo, A MLHL based method to an agent-based architecture, *International Journal of Computer Mathematics* **86**(10, 11) (2008), 1760–1768.

[14] J.M. Corchado, J. Bajo, Y. De Paz and D.I. Tapia, Intelligent environment for monitoring alzheimer patients, agent technology for health care, *Decision Support Systems* **44**(2) (2008), 382–396.

[15] R.C. Cordeiro, J.M. Fonseca and A. Donellan, Euronet, LabA cloud based laboratory environment, In: *Global Engineering Education Conference (EDUCON'12)* (2012), pp 1–9.

[16] P. Davidsson, Multi agent based simulation: Beyond social simulation, *Multi Agent Based Simulation*, Springer Verlag LNCS series, Volume 1979, (2001), pp. 97–107.

[17] R. Dębski, A. Byrski and Kisiel-Dorohinicki, Marek, Towards and agent-based augmented cloud, *Journal of Telecommunications and Information Technology* **1** (2012), 16–22.

[18] DMASON. Distributed Multi-Agents simulation toolkit. http://isis.dia.unisa.it/projects/dmason/ (Last access in 2013).

[19] H. Erdogmus, Cloud computing: Does nirvana hide behind the Nebula? *IEEE Software* **26**(2) (2009), 4–6.

[20] R. Fuentes-Fernández, S. Hassan, J. Pavón, J.M. Galán and A. López-Paredes, Metamodels for role-driven agent-based modeling, *Computational and Mathematical Organization Theory* **18**(1) (2010), 91–112.

[21] R. Grewa and P. Pateriya, A rule-based approach for effective resource provisioning in hybrid cloud environment, *International Journal of Computer Science and Informatics* **4** (2012), 101–106.

[22] S. Heras, F. De la Prieta, V. Julian, S. Rodríguez, V. Botti, J. Bajo and J.M. Corchado, Agreement technologies and their use in cloud computing environments, *Progress in Artificial Intelligence* **1**(4) (2012), 277–290.

[23] JADE, Java Agent Development Platform http://JADE.tilab.com

[24] G.T. Jayaputera, A.B. Zaslavsky and S.W. Loke, Enabling run-time composition and support for heterogeneous pervasive multi-agent systems, *Journal of Systems and Software* **80**(12) (2007), 2039–2062.

[25] Y. Li, W. Shen and H. Ghenniwa, *Agent-Based Web Services Framework and Development Environment*, Computational Intelligence, In: Blackwell Publishing, **20**(4) (2004), 678–692.

[26] X. Liu, A multi-agent-based service-oriented architecture for inter-enterprise cooperation system, In: *Proceedings of the Second international Conference on Digital Telecommunications (ICDT'07)*, IEEE Computer Society, 2007. pp. 22

[27] S. Luke, C. Cioffi-Revilla, L. Panait and K.M. Sullivan, A new multiagent simulation toolkit, In: *Proceedings of the 2004 SwarmFest Workshop*, 2004.

[28] S. Malik, F. Huet and D. Caromel, Cooperative cloud computing in research and academic environment using virtual Cloud, *International Conference on Emerging Technologies* (2012), pp. 1–7.

[29] P. Mell and T. Grance, *The NIST definition of Cloud Computing*, In: NIST Special Publication, 2011, pp. 800–145.

[30] M.J. North, T. Collier Nicholson and R. Vos Jerry, Experiences creating three implementations of the repast agent modeling toolkit, *ACM Transactions on Modeling and Computer Simulation* **16**(1) (2006), 1–25.

[31] M.J. North, T.R. Howe, N.T. Collier and J.R. Vos, The repast symphony runtime system, In: *Proceedings of the Agent 2005 Conference on Generative Social Processes*, Models, and Mechanisms.

[32] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, The eucalyptus open-source cloud-computing system, In: *9th IEEE/ACM International Symposium Cluster Computing and the Grid, 2009*, CCGRID '09, 2009, pp. 124–131.

[33] C. Pautasso, O. Zimmermann and F. Leymann, Restful web services vs. "big"' web services: Making the right architectural decision, In: *Proceedings of the 17th international conference on World Wide Web, WWW '08*, 2008, pp. 805–814.

[34] Repast, http://repast.sourceforge.net/repast_3/index.html

[35] A. Ricci, C. Buda and N. Zaghini, An agent-oriented programming model for SOA & web services, In: *5th IEEE International Conference on Industrial Informatics (INDIN'07)*, 2007, pp. 1059–1064.

[36] J.W. Ross and G. Wsterman, Preparing for utility computing: The role of IT architecture and relationship management, *IBM Systems Journal* **43**(1) (2004), 5–19.

[37] M.O. Shafiq, Y. Ding and D. Fensel, Bridging multi agent systems and web services: Towards interoperability between software agents and semantic web services, In: *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)* (2006), pp. 85–96.

[38] A. Shendarkar, K. Vasudevan, S. Lee and Y.-J. Son, Crowd simulation for emergency response using BDI agent based on virtual reality, In: *Proceedings of the 2006 Winter Simulation Conference*, 2006, pp. 545–553.

[39] M. Sim, Agent-based Cloud commerce, In: *IEEE International Conference on Industrial Engineering and Engineering Management* (2009), pp. 717–721.

[40] Swarm, http://www.swarm.org

[41] D. Talia, Clouds meet agents: Towards intelligent cloud services, *IEEE Internet Computing* **16**(2) (2012), 78–81.

[42] G. Villarrubia, A. Sánchez, I. Barri, E. Rubión, A. Fernández, C. Rebate, J.A. Cabo, T. Álamos, J. Sanz, J. Seco, C. Zato, J. Bajo, S. Rodríguez and J.M. Corchado, *Proximity Detection Prototype Adapted to a Work Environment*, In: Ambien Intelligent – Software and Applications, Springer AISC Volume 153. 2002. pp. 51–59.

[43] G. Vizzari, G. Pizzi and F.S.C. da Silva, A framework for execution and visualization of situated agents based virtual environments, In: *Workshop dagli Oggetti agli Agenti*, 2007, pp. 22–25.

[44] VMWare. http://www.vmware.com/es/ (Last access in 2013).

[45] C. Walton, *Agency and the Semantic Web*, In: Oxford University Press, Inc. 2002.

[46] M. Wooldridge and N.R. Jennings, *Agent Theories, Architectures, and Languages: A Survey*, In: Intelligent Agents, Springer, 1995, pp. 1–22.

[47] F. Zambonelli, N.R. Jennings and M. Wooldridge, Developing multiagent systems: The gaia methodology, *ACM Transactions on Software Engineering and Methodology* **12** (2003), 317–370.

[48] ZigBee Standards Organization, ZigBee Specification Document 053474r13, ZigBee Alliance (2006).