Cloud Computing Integrated into Service-Oriented Multi-Agent Architecture

Sara Rodríguez[1], Dante I. Tapia[1], Eladio Sanz[1], Carolina Zato[1], Fernando de la Prieta[1], Oscar Gil[1]

[1] Departamento de Informática y Automática, Universidad de Salamanca
Plaza de la Merced, s/n, 37008, Salamanca, Spain
{srg, dantetapia, esanz, carol_zato, fer, oscar.gil}@usal.es

Abstract. The main objective of Cloud Computing is to provide software, services and computing infrastructures carried out independently by the network. This concept is based on the development of dynamic, distributed and scalable software. In this way there are Service-Oriented Architectures (SOA) and agent frameworks which provide tools for developing distributed systems and multi-agent systems that can be used for the establishment of cloud computing environments. This paper presents CISM@ (Cloud computing Integrated into Service-Oriented Multi-Agent) architecture set on top of the platforms and frameworks by adding new layers for integrating a SOA and Cloud Computing approach and facilitating the distribution and management of functionalities.

Keywords: Cloud Computing, Multi-Agent Architecture, SOA.

1 Introduction

The main objective of Cloud Computing is for the network to independently provide software, services and computing infrastructure. This concept is based on the development of dynamic, distributed and scalable software. These types of software usually require the creation of increasingly complex and flexible applications, so there is a trend toward reusing resources and sharing compatible platforms or architectures. In some cases, applications require similar functionalities already implemented into other systems, which are not always compatible.

For this reason, it is necessary to develop new functional architectures capable of providing adaptable and compatible frameworks and allowing access to services and applications regardless of time and location restrictions. There are Service-Oriented Architectures (SOA) and agent frameworks [16], [17], [18], which provide tools for developing distributed systems and multi-agent systems [8], [11], [2] that can be used for the establishment of cloud computing environments. However, these tools do not solve the development requirements of these systems by themselves. Therefore, it is necessary to develop innovative solutions that integrate different approaches in order to create flexible and adaptable systems, especially for achieving higher levels of interaction with people in a ubiquitous and intelligent way.

One of the most prevalent alternatives in distributed architectures is agent and multi-agent systems which can help to distribute resources and reduce the central unit tasks [1]. CISM@ (Cloud Computing Integrated on Service-oriented Multi-Agent) architecture is set on top of the platforms and frameworks by adding new layers for integrating a SOA and Cloud Computing approach and facilitating the distribution and management of functionalities. A distributed agent-based architecture provides more flexible ways to move functions to where actions are needed. Additionally, the programming effort is reduced because it is only necessary to specify global objectives so that agents cooperate in solving problems and reaching specific goals, thus giving the systems the ability to generate knowledge and experience. Unfortunately, the difficulty in developing a multi-agent architecture lies in the need of having more complex analysis and design stages, which implies more time to reach the implementation. Moreover, the system control is reduced because the agents need more autonomy to solve complex problems.

The main purpose of this research is to design and implement an architecture that is applicable to the development of intelligent highly dynamic environments. The architecture proposes several features capable of being executed in dynamic and distributed environments to provide interoperability in a standard framework. These features can be implemented in devices with limited storage and processing capabilities. CISM@ integrates intelligent agents with a service-oriented philosophy.

In the next section, the problem that has motivated the development of the CISM@ architecture, combining concepts of Cloud Computing, SOA and Multi-Agent, systems will be explained. The specific characteristics and the agent-based architecture will be described in section 3. Finally, section 4 will present the results and the conclusions obtained.

2 Problem Description

Distributed architectures are aimed at achieving interoperability between different systems, distribution of resources, and the lack of dependency of programming languages [4]. Functionalities are linked by means of standard communication protocols that must be used by applications in order to share resources in the network [1]. The compatibility and management of messages between functionalities is an important and complex element in any of these approaches.

Agent and multi-agent systems (MAS) combine classical and modern functional architecture aspects. We have that an agent is anything with the ability to perceive its environment through sensors and respond in the same environment through actuators, assuming that each agent may perceive its own actions and learn from the experience [10]. MAS is defined as any system composed of multiple autonomous agents with incomplete capabilities to solve a global problem, where there is no global control system, the data is decentralized and the computing is asynchronous) [15].

Multi-agent architectures and frameworks such as Open Agent Architecture (OAA) [8], RETSINA [11] and JADE [2] define agent-based structures to resolve distributed computational problems and facilitate user interactions. Nevertheless, integration is not always achieved because of two reasons. The first is the incompatibility between the agent platforms. To resolve this, two alternatives are possible. One of them is centered on the communication between the different models of the platform [3] and the other is focused on the integration of distributed services in the agent infrastructure [9]. The second reason is that Cloud-based architectures usually do not provide intelligent computational and interactive mechanisms. CISM@ combines both paradigms, trying to take advantage of their strengths and avoid their weaknesses.

Although these developments provide an adequate background for developing distributed MAS with a cloud computing approach based on Web Services, most of them are in early stages of development, so it is not actually possible to know their potential in real scenarios. In addition, CISM@ not only provides communication and integration between distributed agents, services and applications; it also proposes a new method for facilitating the development of distributed MAS by means of modeling the functionalities of the agents and the systems as services and applications based on cloud computing.

Cloud Computing is an innovative concept, which is based on the compendium of a group of technologies. These proportionate a computational paradigm to offer services in three different levels: IaaS, PaaS y SaaS [5], [13], [7].

The IaaS (Infrastructure as a Service) level is oriented to provide hardware and/or software equipment (operating system, process capacity, storage, etc.) as a technology environment in different developments. The capacity to vary under demand is also included in the characteristics of the services [12], [14]. The PaaS (Platform as a Service) level offers integration but with higher abstraction, allowing the construction of customized services by different services [6]. The SaaS (Software as a Service) level offers software with a specific purpose that is totally functional and available through the Internet.

Although the concept of providing services at three different levels is clearly accepted, the needed standards and interfaces are currently still not defined and therefore, the interoperability between different Cloud Computing platforms is an open problem today [10]. CISM@ aims to solve these problems, by specifying a well-defined architecture that integrates the main frameworks of intelligent agents in order to provide Cloud Computing services, both at software (SaaS) and platform (PaaS) level.

CISMA@ allows the development of MAS with increased scalability and reutilization of resources. In addition, CISMA@ allows the extraction and modeling of the agents functionalities as independent services and applications, giving as a result, lighter agents in terms of computational processing. A distributed approach provides the architecture with a greater capacity for recovery from errors, and thus, a greater flexibility to adjust its behavior in execution time.

3 CISM@ Architecture

CISM@ is a novel architecture which integrates a cloud computing approach with SOA and intelligent agents for building systems that need be dynamic, flexible, robust, adaptable to changes in context, scalable and easy to use and maintain. The architecture proposes a new and easier method to develop distributed intelligent systems, where cloud services can communicate in a distributed way with intelligent agents, even from mobile devices, independent of time and location restrictions. The architecture focuses on distributing the majority of the systems' functionalities into remote and local services and applications. The functionalities of the systems are not integrated into the structure of the agents; rather they are modeled as distributed services and applications that are invoked by the agents acting as controllers and coordinators. Because the architecture acts as an interpreter, the users can run applications and services programmed in virtually any language, but have to follow a communication protocol that every cloud service must incorporate. Another important functionality is that, thanks to the agents' capabilities, the systems developed can make use of reasoning mechanisms or learning techniques to handle cloud services according to context characteristics, which can change dynamically over time. Agents and cloud services can communicate in a distributed way, even from mobile devices. This makes it possible to use resources no matter its location. It also allows the starting or stopping of agents, services or devices separately, without affecting the rest of resources, so the system has an elevated adaptability and capacity for error recovery.

CISM@ is based on agents because of their characteristics, such as autonomy, reasoning, reactivity, social abilities, pro-activity, mobility, organization, etc., which allow them to cover several needs for highly dynamic environments, especially ubiquitous communication and computing and adaptable interfaces. CISM@ combines a cloud computing approach built on top of Web Services and intelligent agents to obtain an innovative architecture, facilitating ubiquitous computation and communication and high levels of human-system-environment interaction. It also provides an advanced flexibility and customization to easily add, modify or remove services on demand, independently of the programming language. The goal in CISM@ is not only to distribute services, but also to promote a new way of developing highly dynamic systems focusing on ubiquity and simplicity. CISM@ provides a flexible distribution of resources and facilitates the inclusion of new functionalities in highly dynamic environments based on Cloud Computing concept. It also provides the systems with a higher ability to recover from errors and a better flexibility to change their behavior at execution time.

CISM@ sets on top of existing agent frameworks by adding new layers to integrate a cloud computing approach and facilitate the provision and management of services at two different levels, Software as a Service (SaaS) and Platform as a Service (PaaS) [5]. Therefore, the CISM@ framework has been modeled following the Cloud Computing model based on SOA, but adding the applications block which represents the interaction with users. These blocks provide all the functionalities of the architecture. CISM@ adds new features to common agent frameworks, such as OAA, RETSINA and JADE and improves the services provided by these previous architectures. These previous architectures have limited communication abilities.

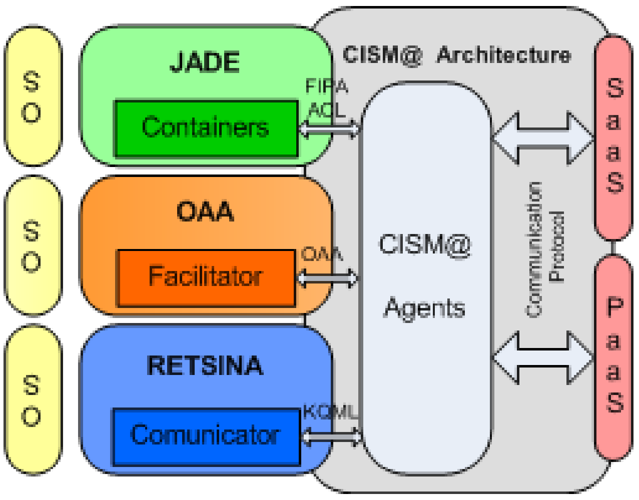

Fig. 1. CISM@ Architecture over Different Agent Platforms

As can be seen in Figure 1, CISM@ defines four basic blocks:

1. *PaaS (Platform as a Service).* This involves all the custom applications that can be used to exploit the system functionalities. Applications are dynamic and adaptable to context, reacting differently according to the particular situation. They can be executed locally or remotely, even on mobile devices with limited processing capabilities, because computing tasks are largely delegated to the agents and services.
2. *Agent Platform.* This is the core of CISM@, integrating a set of agents, each one with special characteristics and behavior. The agents act as controllers and administrators for all cloud services, managing the adequate functioning of the system. In CISM@, services are managed and coordinated by deliberative BDI agents with distributed computation and coordination abilities.
3. *SaaS (Software as a Service).* These represent the activities that the architecture offers. Services are designed to be invoked locally or remotely and they can be organized as local services, Web Services, Cloud services, or even as individual stand-alone services. Services can make use of other services to provide the functionalities that users require. CISM@ has a flexible and scalable directory of services, so they can be invoked, modified, added, or eliminated dynamically and on demand.
4. *Communication Protocol.* This allows applications and services to communicate directly with the agent platform. The protocol is completely open and independent of any programming language, facilitating ubiquitous communication capabilities. This protocol is based on SOAP specification to capture all messages between the platform and the services and applications [4]. All external communications follow the same protocol, while the communication amongst agents in the platform follows the FIPA Agent Communication Language (ACL) specification.

One of the advantages of CISMA@ is that the users can access the system through distributed applications, which run on different types of devices and interfaces (e.g. computers, cell phones, PDAs). All requests and responses are handled by the agents in the platform. The agents analyze all requests and invoke the specified services either locally or remotely. Services process the requests and execute the specified tasks. Then, the services send back a response with the result of the specific task.

A simple case can better demonstrate the basic functioning of CISM@ when it is requesting a service. A user needs to calculate the sum of two numbers and wants to do it through a mobile device (e.g. PDA) remotely connected to the system. The user executes a mathematical toolkit, which provides him with a large set of formulas from which he selects the sum function, introduces a set of values, and clicks a button to get the result. When the user clicks the button, the application sends a request to the platform to find a service that can process that request. The agents invoke the appropriate service and send the request. The service processes the request and sends the result back to the agents, which in turn, send it to the application. It is obvious that invoking a remote service to execute a sum is not the best choice. But imagine a large-scale process that uses complex AI (Artificial Intelligence) techniques, such as genetic algorithms, data mining, neural networks, etc. where the limited processing capacity of the mobile device makes it impossible to calculate. In this case, the service may be in a powerful computer and could be remotely invoked by the mobile device.

The Web Services Architecture model uses an external directory, known as UDDI (Universal Description, Discovery and Integration), to list all available services. Each service must send a WSDL (Web Services Description Language) file to the UDDI to be added to the directory. Applications consult the UDDI to find a specific service. Once the service is located, the application can start communication directly with the selected service. However, CISM@ does not include a service discovery mechanism, so applications must use only the services listed in the platform. In addition, all communication is handled by the platform, so there is no way to interact directly between applications and services. Moreover, the platform makes use of deliberative agents to select the optimal option to perform a task, so users do not need to find and specify the service to be invoked by the application. These features have been introduced in CISM@ to create a secure communication between applications and services. They also facilitate the inclusion of new services regarding their location and application that users can make use of.

CISM@ is a modular multi-agent architecture, where services and applications are managed and controlled by deliberative BDI agents. There are different kinds of agents in the architecture, each one with specific roles, capabilities and characteristics. This fact facilitates the flexibility of the architecture when incorporating new agents. However, there are pre-defined agents, which provide the basic functionalities of the architecture.

The CISM@ pre-defined agents are described next:

1. *PaaS Agent.* This agent is responsible for all communications between applications and the agent platform. It manages the incoming requests from the applications to be processed by services. It also manages responses from services (via the platform) to applications. All messages are sent to Security Agent for their structure and syntax to be analyzed.

2. *SaaS Agent.* This agent is responsible for all communications between services and the agent platform. The functionalities are similar to PaaS Agent but backwards. All messages are sent to Security Agent for their structure and syntax to be analyzed. This agent also periodically checks the status of all services to know if they are idle, busy, or crashed.
3. *ServiceDir Agent.* This agent manages the list of services that can be used by the system. For security reasons, the list of services is static and can only be modified manually; however, services can be added, erased or modified dynamically. There is dynamic information that is constantly being modified: the service performance (average time to respond to requests), the number of executions, and the quality of the service (QoS). This last data assigns a value between 0 and 1 to all services. All new services have a quality of service value set to 1. This value decreases when the service fails (e.g. service crashes, no service found, etc.) or has a subpar performance compared to similar past executions. Security must be a major concern when developing this kind of systems. For this reason CISM@ does not implement a service discovery mechanism. However, agents can select the most appropriate service (or group of services) to accomplish a specific a task.
4. *Control Agent.* This agent supervises the correct functioning of the other agents in the system.
5. *Security Agent.* This agent analyzes the structure and syntax of all incoming and outgoing XML messages.
6. *Manager Agent.* The Manager Agent decides which service must be called by taking into account the QoS and users' preferences. Users can explicitly invoke a service, or can let the Manager Agent decide which service is the best to accomplish the requested task
7. *Interface Agent.* This kind of agent was designed to be embedded in users' applications. Interface agents communicate directly with the agents in CISM@. These agents must be simple enough to allow them to be executed on mobile devices, such as cell phones or PDAs. All high demand processes must be delegated to services.

CISM@ is an open architecture that allows developers to modify the structure of the agents described before. Developers can add new agent types or extend the existing ones to conform to their projects needs. However, most of the agents' functionalities should be modeled as services, releasing them from tasks that could be performed by services.

4 Conclusions and Future Work

CISM@ facilitates the development of dynamic and intelligent multi-agent systems. Its model is based on a cloud computing approach where the functionalities are implemented using Web Services. The architecture proposes an alternative where agents act as controllers and coordinators. CISM@ exploits the agents' characteristics to provide a robust, flexible, modular and adaptable solution that can cover most requirements of a wide diversity of distributed systems. All functionalities, including those of the agents, are modeled as distributed services and applications.

One of the objectives of the research activity is testing the application of Cloud Computing and Cloud services to systems and platforms oriented to Ambient Intelligent environments. Therefore, the next step in this research will be the testing of the proposed architecture in a real case study; in particular, its implementation in an e-health system to enhance assistance and health care for patients with dependencies. This system is suitable for validating the architecture because includes a large number of distributed services with different natures which can be integrated to test the architecture.

As a conclusion we can say that although CISM@ is still under development, preliminary results demonstrate that it is adequate for building complex systems and exploiting composite services. However, services can be any functionality (mechanisms, algorithms, routines, etc.) designed and deployed by developers. CISM@ has laid the groundwork to boost and optimize the development of future projects and systems that combine the flexibility of a cloud computing approach with the intelligence provided by agents. CISM@ makes it easier for developers to integrate independent services and applications because they are not restricted to programming languages supported by the agent frameworks used (e.g. JADE, OAA, RETSINA). The distributed approach of CISM@ optimizes usability and performance because it can obtain lighter agents by modelling the systems' functionalities as independent services and applications outside of the agents' structure, thus these may be used in other developments.

Acknowledgments
The heading should be treated as a 3rd level heading and should not be assigned a number.

References

1. Ardissono, L., Petrone, G., Segnan, M.: A Conversational Approach to the Interaction with Web Services. Computational intelligence, vol. 20, 2004, pp. 693-709.
2. Bellifemine, F., Poggi, A., Rimassa, G.: JADE–A FIPA-compliant Agent Framework. In: Proceedings of PAAM, 1999, pp. 97-108.
3. Bonino da Silva, L.O., Ramparany, F., Dockhorn, P., Vink, P., Etter, R., Broens, T.: A Service Architecture for Context Awareness and Reaction Provisioning. In: IEEE Congress on Services, 2007. pp. 25-32.
4. Cerami, E. Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. O'Reilly Media, Inc., 2002.
5. Foste, I., Zhao, Y., Raicu, I,. Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. In: Grid Computing Environments Workshop (GCE'08). DOI 10.1109/GCE.2008.4738445. pp. 1-10.
6. Google App Engine. http://code.google.com/intl/es-ES/appengine/
7. Gruman, G., Knorr, E.: What Cloud Computing really means. Info World, April 2008. Electronic Magazine, available at http://infoworld.com/article/08/04/07/15FE-cloud-computing-reality_1.html.

8. Martin, D.L., Chever, A.J., Moran, D.B.: The Open Agent Architecture: A framework for Building Distributed Software Systems. Applied Artificial Intelligence, vol. 13, 1999, pp. 91-128.
9. Ricci, A., Buda, C., Zaghini, N.: An agent-oriented programming model for SOA & web services. 5th IEEE Conference on Industrial Informatics (INDIN'07), Vienna, Austria. pp. 1059-1064.
10. Russell, S.J., Norvig, P., Canny, J.F., Malik, J., Edwards, D.D.: Artificial Intelligence: a Modern Approach. Prentice Hall Englewood Cliffs, NJ, 1995.
11. Sycara, K., Paolucci, M., Van Velsen, M., Giampapa. J. The RETSINA MAS Infrastructure. Autonomous Agents and Multi-Agent Systems, vol. 7, Jul. 2003, pp. 29-48.
12. S3 Amazon. http://aws.amazon.com/s3/
13. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the Clouds: Towards a Cloud Definition. SIGCOMM Comput. Commun. Rev., Vol. 39 , Nr. 1New York, NY, USA: ACM (2009) , p. 50—55
14. Varia, J.: Amazon white paper on cloud architectures. Sept 2008. Available at http://aws.typepad.com/aws/2008/07/white-paper-on.html.
15. Wooldridge, M.: An Introduction to MultiAgent Systems, Wiley, 2002.
16. Maamar, Z., Kouadri, S., Yahyaoui, H.: Toward an Agent-Based and Context-Oriented Approach for Web Services Composition. IEEE Transactions on Knowledge and Data Engineering 17(5), pp. 686-697, 2005.
17. Buhler, P., Vidal, J.M.: Integrating Agent Services into BPEL4WS Defined Workflows. In Proceedings of the 4th International Workshop on Web-Oriented Software Technologies, pp. 244-251, 2004.
18. Fuentes-Fernández, R., García-Magariño, I., Gómez-Sanz, J. J., Pavón, J.: Integration of Web Services in an Agent-Oriented Methodology. International Transactions on Systems Science and Applications 3, pp. 145-161, 2007.