Developing a Service Oriented Alternative for Distributed Multi-Agent Systems

Dante I. Tapia, Juan F. de Paz, Sara Rodríguez, Javier Bajo and Juan M. Corchado

Departamento Informática y Automática
Universidad de Salamanca
Plaza de la Merced s/n, 37008, Salamanca, Spain
{dantetapia; fcofds; srg; jbajope; corchado}@usal.es

Abstract. This paper presents a service oriented architecture approach that has enhanced the performance of a multi-agent system aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. The proposed architecture allows a more efficient distribution of the daily activities in the multi-agent system. The results obtained after testing the architecture in a real health care scenario demonstrate that the ALZ-MAS 2.0 based on the service oriented approach is far more robust and has better performance than the previous version of this system.

Keywords: Multi-agent Systems, Services Oriented Architectures, Case-Based Reasoning, Case-Based Planning, Health Care.

1 Introduction

The continuous growth of the Internet requires frameworks for web application integration [12]. Web applications are executed in distributed environments, and each part that composes the program can be located in a different machine. The absence of a strategy for integrating applications generates multiple points of failure that can affect the systems' performance. Some of the technologies that have acquired a relevant paper in the web during the last years are the multi-agent systems and the SOA architectures. This work describes a novel architecture for developing multi-agent systems and explains how it has been designed and applied to a real scenario. The architecture presents important improvements in the vision of the integration of web applications. One of the most important characteristics is the use of intelligent agents as the main components in employing a service oriented approach, focusing on distributing the majority of the systems' functionalities into remote and local services and applications. The architecture proposes a new and easier method of building distributed multi-agent systems, where the functionalities of the systems are not integrated into the structure of the agents, rather they are modelled as distributed services which are invoked by the agents acting as controllers and coordinators.

Agents have a set of characteristics, such as autonomy, reasoning, reactivity, social abilities, pro-activity, mobility, organization, etc. which allow them to cover several needs for highly dynamic environments. Agent and multi-agent systems have been

successfully applied to several scenarios, such as education, culture, entertainment, medicine, robotics, etc. [7]. The characteristics of the agents make them appropriate for developing dynamic and distributed systems, as they possess the capability of adapting themselves to the users and environmental characteristics [9]. Most of the agents are based on the deliberative Belief, Desire, Intention (BDI) model [15], where the agents' internal structure and capabilities are based on mental aptitudes, using beliefs, desires and intentions [3]. Nevertheless, complex systems need higher adaptation, learning and autonomy levels than pure BDI model [3]. This is achieved by modelling the agents' characteristics [15] to provide them with mechanisms that allow solving complex problems and autonomous learning. Some of these mechanisms are Case-Based Reasoning (CBR) [1] and Case-Based Planning (CBP), where problems are solved by using solutions to similar past problems [6] [7]. Solutions are stored into a case memory, which the mechanisms can consult in order to find better solutions for new problems. CBR and CBP mechanisms have been modelled as external services. Deliberative agents use these services to learn from past experiences and to adapt their behaviour according the context.

This paper briefly describes the FUSION@ architecture, a service oriented alternative for distributed multi-agent systems and ALZ-MAS 2.0, a multi-agent system aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. ALZ-MAS 2.0 is based on FUSION@ and implements a services oriented approach, where functionalities, including CBR and CBP mechanisms, are not integrated into the structure of the agents, rather they are modelled as distributed services and applications which are invoked by the agents.

In the next section, the problem description that motivated this work is presented. Section 3 briefly presents the FUSION@ architecture. Section 4 describes the basic components of ALZ-MAS 2.0 and shows how a CBP mechanism has been modelled for distributing resources. Finally Section 5 presents the results and conclusions obtained in this work.

2 Problem Description

Excessive centralization of services negatively affects the systems' functionalities, overcharging or limiting their capabilities. Classical functional architectures are characterized by trying to find modularity and a structure oriented to the system itself. Modern functional architectures like Service-Oriented Architecture (SOA) consider integration and performance aspects that must be taken into account when functionalities are created outside the system. These architectures are aimed at the interoperability between different systems, distribution of resources, and the lack of dependency of programming languages [5]. Services are linked by means of standard communication protocols that must be used by applications in order to share resources in the services network [2]. The compatibility and management of messages that the services generate to provide their functionalities is an important and complex element in any of these approaches.

One of the most prevalent alternatives to these architectures is the multi-agent systems technology which can help to distribute resources and reduce the central unit

tasks [2]. A distributed agents-based architecture provides more flexible ways to move functions to where actions are needed, thus obtaining better responses at execution time, autonomy, services continuity, and superior levels of flexibility and scalability than centralized architectures [4]. Additionally, the programming effort is reduced because the agents cooperate in solving problems and reaching specific goals, thus giving the systems the ability to generate knowledge and experience.

Agent and multi-agent systems combine classical and modern functional architecture aspects. Multi-agent systems are structured by taking into account the modularity in the system, and by reuse, integration and performance. Nevertheless, integration is not always achieved because of the incompatibility among the agents' platforms. The integration and interoperability of agents and multi-agent systems with SOA and Web Services approaches has been recently explored [2]. Some developments are centred on communication between these models, while others are centred on the integration of distributed services, especially Web Services, into the structure of the agents [13] [14] [10] [11]. Although these developments provide an adequate background for developing distributed multi-agent systems integrating a service oriented approach, most of them are in early stages of development, so it is not possible to actually know their potential in real scenarios.

3 The FUSION@ Architecture

The development of AmI-based software requires creating increasingly complex and flexible applications, so there is a trend toward reusing resources and share compatible platforms or architectures. In some cases, applications require similar functionalities already implemented into other systems which are not always compatible. At this point, developers can face this problem through two options: reuse functionalities already implemented into other systems; or re-deploy the capabilities required, which means more time for development, although this is the easiest and safest option in most cases. While the first option is more adequate in the long run, the second one is most chosen by developers, which leads to have replicated functionalities as well as greater difficulty in migrating systems and applications. This is a poorly scalable and flexible model with reduced response to change, in which applications are designed from the outset as independent software islands.

FUSION@ has been designed to facilitate the development of distributed multi-agent systems with high levels of human-system-environment interaction, since agents have the ability to dynamically adapt their behaviour at execution time. It also provides an advanced flexibility and customization to easily add, modify or remove applications or services on demand, independently of the programming language. FUSION@ formalizes four basic blocks: Applications, which represent all the programs that can be used to exploit the system functionalities. 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; An Agents Platform as the core of FUSION@, integrating a set of agents, each one with special characteristics and behaviour. These agents act as controllers and administrators for all applications and services, managing the adequate functioning of the system, from

services, applications, communication and performance to reasoning and decision-making; Services, which are the bulk of the functionalities of the system at the processing, delivery and information acquisition levels. Services are designed to be invoked locally or remotely; and finally a Communication Protocol which allows applications and services to communicate directly with the Agents Platform. The protocol is based on SOAP specification and it is completely open and independent of any programming language [5].

These blocks are managed by means of pre-defined agents which provide the basic functionalities of FUSION@: CommApp Agent is responsible for all communications between applications and the platform; CommServ Agent is responsible for all communications between services and the platform; Directory Agent manages the list of services that can be used by the system; Supervisor Agent supervises the correct functioning of the other agents in the system; Security Agent analyzes the structure and syntax of all incoming and outgoing messages; Manager Agent decides which agent must be called by taking into account the services performance and users preferences; Interface Agents are designed to be embedded in users' applications. Interface agents communicate directly with the agents in FUSION@ so there is no need to employ the communication protocol, rather the FIPA ACL specification.

FUSION@ also facilitates the inclusion of context-aware technologies that allow systems to automatically obtain information from users and the environment in an evenly distributed way, focusing on the characteristics of ubiquity, awareness, intelligence, mobility, etc. The goal in FUSION@ is not only to distribute services and applications, but to also promote a new way of developing systems focusing on ubiquity and simplicity. An example of a service in FUSION@ can be observed in Figure 1.

SERVICE	DESCRIPTION	
readCHIP	This service identificates a chip once it has been detected for a RFID reader.	
P R O F I L E ClientRole User Agent	ProviderRole Devices Agent	
Inputs	Outputs	
IdDevice: string typeDevice: string deviceLocation: location	[typeCHIP OK] IdCHIP: string IdUser: string chipLocation: location	[typeCHIP NOT OK]

Fig. 1. ReadCHIP: An example of service in FUSION@.

Figure 1 shows the readCHIP service. This service has been implemented to facilitate indoor location based on RFID technology. When a RFID reader detects the presence of a chip, the readCHIP service is automatically invoked. The inputs considered for this service consists of the device identification, the type of device, and the location of the device. At this moment the service checks the type of CHIP and calculates the location information, that is, the identification for the chip, the user

identification and the coordinates which determine the physical position. This information is then sent to the Devices Agent in order to be automatically processed.

In the next section, ALZ-MAS 2.0 is presented, where FUSION@ has helped to distribute most of its functionalities and re-design a completely functional multi-agent system aimed at improving several aspects of dependent people.

4   ALZ-MAS 2.0

ALZ-MAS 2.0 is an improved version of ALZ-MAS (ALZheimer Multi-Agent System) [6] [7], a multi-agent system aimed at enhancing the assistance and health care for Alzheimer patients living in geriatric residences. The main functionalities in the system are managed by deliberative BDI agents, including Case-Based Reasoning (CBR) and Case-Based Planning (CBP) mechanisms.

ALZ-MAS structure has five different deliberative agents based on the BDI model (BDI Agents), each one with specific roles and capabilities:

- *User Agent*. This agent manages the users' personal data and behaviour (monitoring, location, daily tasks, and anomalies). The *User Agent* beliefs and goals applied to every user depend on the plan or plans defined by the super-users.
- *SuperUser Agent*. This agent inserts new tasks into the Manager Agent to be processed by a CBR and CBP mechanisms.
- *ScheduleUser Agent*. It is a BDI agent with a CBP mechanism embedded in its structure. It schedules the users' daily activities and obtains dynamic plans depending on the tasks needed for each user. There is one *ScheduleUser Agent*s for each nurse connected to the system.
- *Admin Agent*. It runs on a Workstation and plays two roles: the security role that monitors the users' location and physical building status (temperature, lights, alarms, etc.) through continuous communication with the *Devices Agent*; and the manager role that handles the databases and the task assignment.
- *Devices Agent*. This agent controls all the hardware devices. It monitors the users' location (continuously obtaining/updating data from sensors), interacts with sensors and actuators to receive information and control physical services (wireless devices status, communication, temperature, lights, door locks, alarms, etc.).

In the initial version of ALZ-MAS, each agent integrated its own functionalities into their structure. If an agent needs to perform a task which involves another agent, it must communicate with that agent to request it. So, if the agent is disengaged, all its functionalities will be unavailable to the rest of agents. This has been an important issue in ALZ-MAS, since agents running on PDAs are constantly disconnecting from the platform and consequently crashing, making it necessary to restart (killing and launching new instances) those agents. Another important issue is that the CBR and CBP mechanisms are integrated into the agents. These mechanisms are busy almost all the time, overloading the respective agents. Because CBR and CBP mechanisms are the core of the system, they must be available at all times. The system depends on these mechanisms to generate all decisions, so it is essential that they have all processing power available in order to increase overall performance. In addition, the use of CBR and CBP mechanisms into deliberative BDI agents makes these agents

complex and unable to be executed on mobile devices. In ALZ-MAS 2.0, these mechanisms have been modelled as services to distribute resources.

The entire ALZ-MAS structure has been modified, separating most of the agents' functionalities from those to be modelled as services. However, all functionalities are the same in both approaches, since we have considered it appropriated to compare the performance of both systems in identical conditions. As an example showing the differences between both approaches, the next sub-section describes the CBP mechanism that has been extracted from the *ScheduleUser Agent* structure and modelled as a service.

As seen on Figure 2, the entire ALZ-MAS structure has been modified according to FUSION@ model, separating most of the agents' functionalities from those to be modelled as services. However, all functionalities are the same in both approaches, since we have considered it appropriated to compare the performance of both systems to prove the efficiency of FUSION@ model.
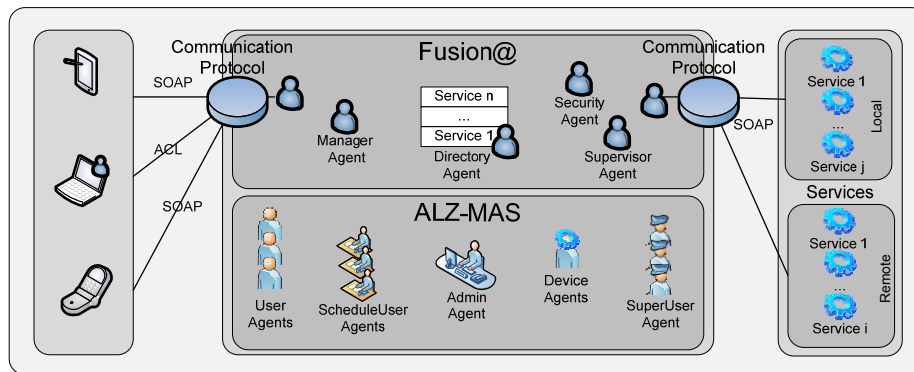

Fig. 2. ALZ-MAS 2.0 basic structure

4.1 A Case-Based Planning Mechanism for Scheduling Daily Activities

As previously mentioned, some agents in ALZ-MAS integrate CBR and CBP mechanisms (then modelled as services in ALZ-MAS 2.0), which allow them to make use of past experiences to create better plans and achieve their goals. These mechanisms provide the agents greater learning and adaptation capabilities. The main characteristics of the CBP mechanism are described in the remainder of this section.

Case-Based Reasoning (CBR) is a type of reasoning based on past experiences [1]. CBR solve new problems by adapting solutions that have been used to solve similar problems in the past, and learn from each new experience. The primary concept when working with CBR is the concept of case, which is described as a past experience composed of three elements: an initial state or problem description that is represented as a belief; a solution, which provides the sequence of actions carried out in order to solve the problem; and a final state, which is represented as a set of goals. CBR manages cases (past experiences) to solve new problems. The way cases are managed

is known as the CBR cycle, and consists of four sequential phases: retrieve, reuse, revise and retain. The retrieve phase starts when a new problem description is received. Similarity algorithms are applied so that the cases with the problem description most similar to the current one can be retrieved from the cases memory. Once the most similar cases have been retrieved, the reuse phase begins by adapting the solutions for the retrieved cases in order to obtain the best solution for the current case. The revise phase consists of an expert revision of the proposed solution. Finally, the retain phase allows the system to learn from the experiences obtained in the three previous phases, and consequently updates the cases memory.

CBP comes from CBR, but is specially designed to generate plans (sequence of actions) [6] [7]. In CBP, the proposed solution for solving a given problem is a plan. This solution is generated by taking into account the plans applied for solving similar problems in the past. The problems and their corresponding plans are stored in a plans memory. The reasoning mechanism generates plans using past experiences and planning strategies, which is how the concept of Case-Based Planning is obtained [7]. CBP consists of four sequential stages: the retrieve stage, which recovers the past experiences most similar to the current one; the reuse stage, which combines the retrieved solutions in order to obtain a new optimal solution; the revise stage, which evaluates the obtained solution; and retain stage, which learns from the new experience. Problem description (initial state) and solution (situation when final state is achieved) are represented as beliefs, the final state as a goal (or set of goals), and the sequences of actions as plans. The CBP cycle is implemented through goals and plans. When the goal corresponding to one of the stages is triggered, different plans (algorithms) can be executed concurrently to achieve the goal or objective. Each plan can trigger new sub-goals and, consequently, cause the execution of new plans. In practice, what is stored is not only a specific problem with a specific solution, but also additional information about how the plans have been derived. As with CBR, the case representation, the plans memory organization, and the algorithms used in every stage of the CBP cycle are essential in defining an efficient planner.

In the initial version of ALZ-MAS, the CBR and CBP mechanisms are deeply integrated into the agents' structure. In ALZ-MAS 2.0, these mechanisms have been modelled as services linked to agents, thus increasing the system's overall performance. To generate a new plan, a *ScheduleUser Agent* (running on a PDA) sends a request to the platform. The message is processed and the platform invokes the mechanism (or service). The mechanism receives the message and starts to generate a new plan. Then, the solution is sent to the platform which delivers the new plan to all *ScheduleUser Agents* running. The CBP service creates optimal paths and scheduling in order to facilitate the completion of all tasks defined for the nurses connected to the system [7].

5   Results and Conclusions

The integration of web applications plays an important role in the advance of the internet. This paper has presented the FUSION@ architecture, which proposes a novel approach for integrating: applications, agents and services. FUSION@

facilitates the inclusion of context-aware technologies that allow systems to automatically obtain information from users and the environment in an evenly distributed way. The proposed architecture has been used to develop the ALZ-MAS 2.0 system, a variation of the previous ALZ-MAS system, which models the CBP-BDI and CBR-BDI mechanisms as services. The performance of ALZ-MAS 2.0 has been highly improved.

Several tests have been done to demonstrate if a SOA approach is appropriate to distribute resources and optimize the performance of multi-agent systems, in this case ALZ-MAS 2.0. The tests consisted of a set of requests delivered to the CBP mechanism which in turn had to generate paths for each set of tasks (i.e. scheduling). For every new test, the cases memory of the CBP mechanism was deleted in order to avoid a learning capability, thus requiring the mechanism to accomplish the entire planning process. A task is a java object that contains a set of parameters (TaskId, MinTime, MaxTime, ScheduleTime, UserId, location, etc.). ScheduleTime is the time in which a specific task must be accomplished, although the priority level of other tasks needing to be accomplished at the same time is factored in. The CBP mechanism increases or decreases ScheduleTime and MaxTime according to the priority of the task: *ScheduleTime = ScheduleTime-5min*TaskPriority and MaxTime = MaxTime+5min*TaskPriority*. Once these times have been calculated, the path is generated taking the RoomCoordinates into account. There were 30 defined agendas each with 50 tasks. Tasks had different priorities and orders on each agenda. Tests were carried out on 7 different test groups, with 1, 5, 10, 15, 20, 25 and 30 simultaneous agendas to be processed by the CBP mechanism. 50 runs for each test group were performed, all of them on machines with equal characteristics. Several data have been obtained from these tests, notably the average time to accomplish the plans, the number of crashed agents, and the number of crashed services. For ALZ-MAS 2.0 five CBP services with exactly the same characteristics were replicated.

Fig. 3(Top) shows the average time needed by both systems to generate the paths for a fixed number of simultaneous agendas. The previous version of ALZ-MAS was unable to handle 15 simultaneous agendas and time increases to infinite because it was impossible to perform those requests. However, ALZ-MAS 2.0 had 5 replicated services available, so the workflow was distributed and allowed the system to complete the plans for 30 simultaneous agendas. Another important data is that although the previous version of ALZ-MAS performed slightly faster when processing a single agenda, performance was constantly reduced when new simultaneous agendas were added. This fact demonstrates that the overall performance of ALZ-MAS 2.0 is better when handling distributed and simultaneous tasks (e.g. agendas), instead of single tasks. Fig. 3(Down) shows the number of crashed agents for both versions of ALZ-MAS during tests. None of the tests where agents or services crashed were taken into account to calculate the data presented in Fig. 3, so these tests were repeated. As can be seen, the previous version of ALZ-MAS is far more unstable than ALZ-MAS 2.0. These data demonstrate that this approach provides a higher ability to recover from errors.
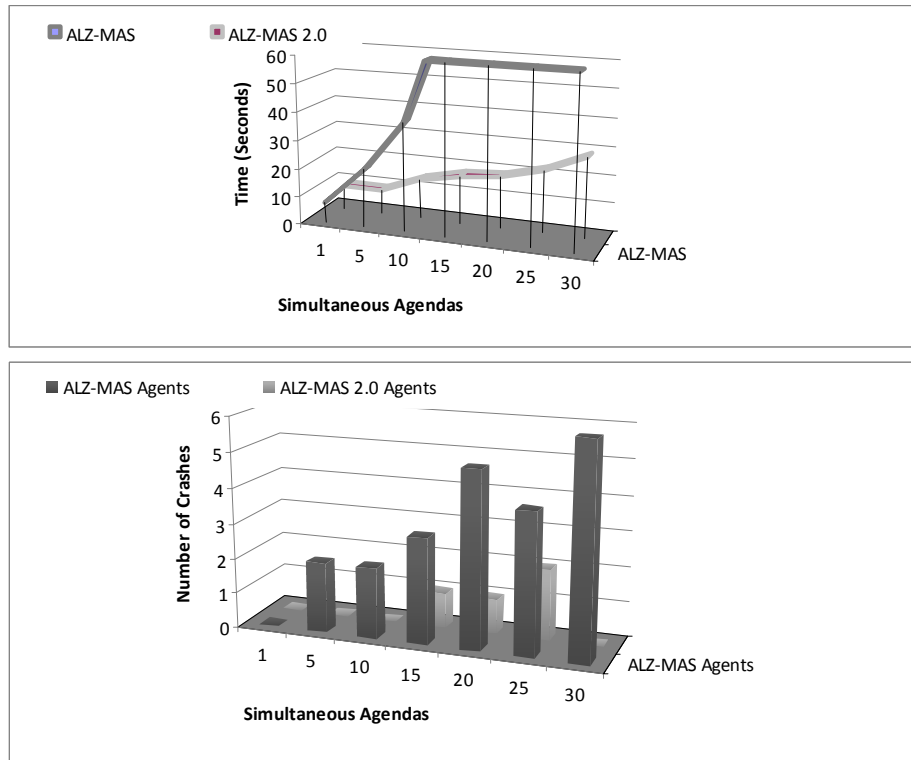
Fig. 3. Top: Time needed for both systems to generate paths for a group of simultaneous agendas; Down: Number of agents crashed at both systems

Although these tests have provided us with very useful data, it is necessary to continue experimenting with FUSION@. A SOA approach is an efficient way to distribute resources and develop more robust multi-agent systems, especially when handling complex mechanisms as the CBP presented.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7, 39-59, IOS Press (1994)
2. Ardissono, L., Petrone, G., Segnan, M.: A conversational approach to the interaction with Web Services. Computational Intelligence, 20, 693-709, Blackwell Publishing (2004)
3. Bratman, M.E.: Intentions, plans and practical reason. Harvard University Press, Cambridge, MA (1987)

4. Camarinha-Matos, L.M., Afsarmanesh, H.: A Comprehensive Modeling Framework for Collaborative Networked Organizations. Journal of Intelligent Manufacturing, 18(5), 529-542, Springer Netherlands (2007)
5. Cerami, E.: Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. O'Reilly & Associates, Inc. 1st Edition (2002)
6. Corchado, J.M., Bajo, J., Abraham, A.: GERAmI: Improving the delivery of health care. IEEE Intelligent Systems, Special Issue on Ambient Intelligence, 23(2), 19-25 (2008)
7. Corchado, J.M., Bajo, J., De Paz, Y., Tapia, D.I.: Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. Decision Support Systems, 44(2), 382-396, Eslevier, Netherlands (2008)
8. de Paz, J.F., Rodríguez, S., Bajo, J., Corchado, J.M.: Dynamic Case Based Planning. In: 8th Int. Conference on Computational and Mathematical Methods in Science and Engineering, vol. 1, pp. 213-224, La Manga del Mar Menor, Spain (2008)
9. Jayaputera, G.T., Zaslavsky, A.B., Loke, S.W.: Enabling run-time composition and support for heterogeneous pervasive multi-agent systems. Journal of Systems and Software, 80(12), 2039-2062 (2007)
10. Li, Y., Shen, W., Ghenniwa, H.: Agent-Based Web Services Framework and Development Environment. Computational Intelligence, 20(4), 678-692, Blackwell Publishing (2004)
11. Liu, X.: A Multi-Agent-Based Service-Oriented Architecture for Inter-Enterprise Cooperation System. In: 2nd International Conference on Digital Telecommunications. IEEE Computer Society, Washington, DC (2007)
12. Oren, E., Haller, A., Mesnage, C., Hauswirth, M., Heitmann, B., Decker, S.: A Flexible Integration Framework for Semantic Web 2.0 Applications. IEEE Software, vol. 24, no. 5, pp. 64-71, (2007)
13. Ricci, A., Buda, C., Zaghini, N.: An agent-oriented programming model for SOA & web services. In: 5th IEEE International Conference on Industrial Informatics, pp. 1059-1064, Vienna, Austria (2007)
14. Shafiq, M.O., Ding, Y., Fensel, D.: Bridging Multi-Agent Systems and Web Services: towards interoperability between Software Agents and Semantic Web Services. In: 10th IEEE International Enterprise Distributed Object Computing Conference, pp. 85-96, IEEE Computer Society, Washington, DC (2006)
15. Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, 10(2), 115-152, Cambridge University Press (1995)