Improving the Tracing System in PANGEA Using the TRAMMAS Model

Luis Búrdalo[1], Andrés Terrasa[1], Vicente Julián[1], Carolina Zato[2], Sara Rodríguez[2], Javier Bajo[2], and Juan M. Corchado[2]

[1] Department of Computer Systems and Computation
Polytechnic University of Valencia
Valencia, Spain
{lburdalo,aterrasa,vinglada}@dsic.upv.es
[2] Department of Computer Science and Automation
University of Salamanca
Salamanca, Spain
{carol_zato,srg,jbajope,corchado}@usal.es

Abstract. This paper presents the integration of the tracing model TRAMMAS in an agent platform called PANGEA. This platform allows to developed multiagent systems modeled as Virtual Organizations. The concepts of roles, organizations and norms are fully supported by the platform assuring flexibility and scalability. Before TRAMMAS, this platform uses a Sniffer Agent to trace the information reducing its scalability as a centralized mechanism. TRAMMAS proposes the use of event tracing in multiagent systems, as an indirect interaction and coordination mechanism to improve the amount and quality of the information that agents can perceive in order to fulfill their goals more efficiently. Moreover, the event tracing system can help reducing the amount of unnecessary information.

Keywords: agent platform, multiagent systems, virtual organizations, IRC protocol, tracing systems.

1 Introduction

Distributed multi-agent systems (MAS) have become increasingly sophisticated in recent years, with the growing potential to handle large volumes of data and coordinate the operations of many organizations [21]. In these systems, each agent independently handles a small set of specialized tasks and cooperates to achieve the system-level goals and a high degree of flexibility [10]. Multiagent systems have become the most effective and widely used form of developing this type of application in which communication among various devices must be both reliable and efficient. One of the problems related to distributed computing is message passing, which is in turn related to the interaction and coordination among intelligent agents. Consequently, a multiagent architecture must necessarily provide a robust communication platform and control mechanisms.

J. Pavón et al. (Eds.): IBERAMIA 2012, LNAI 7637, pp. 422–431, 2012.

This article presents a multiagent platform based on a Virtual Organization (VO) paradigm. In this paradigm, the social behavior (based on abstractions such as norms, teams, organizations, roles, commitments, etc.) plays and important role and it has to be incorporated as a decentralized mechanism. This platform called PANGEA (Platform for Automatic coNstruction of orGanizations of intElligent Agents) includes a robust communication model that allows intelligent agents to connect from a variety of devices. On the other hand, TRAMMAS is a tracing model that is incorporated to the platform to improve the amount and quality of the information that agents can perceive from both their physical and social environment, in order to fulfill their goals more efficiently.

The remainder of the paper is structured as follows: the next section introduces some previous works made in tracing systems. Section 3 presents an overview of the TRAMMAS model. Section 4 explains the inclusion of TRAMMAS inside PANGEA. Next, Sect. 5 presents a case study and some results. Finally, Sect. 6 shows some conclusions.

2 Related Work

The tracing systems within the multiagent architectures have been traditionally used for tasks of "debugging" and the control of certain agents' behavior.

The most outstanding example of this case is the Sniffer Agent and the Introspecter Agent of JADE [23]. The Sniffer Agent allows registering all the messages sent and received by the MAS and later, by means of a log file, to examine its content. The Introspecter Agent allows knowing all the events related to the life cycle of an agent, the messages sent and received as well as its behavior. Nevertheless, in this model all the communication flow is centralized and must pass through this agent to be analyzed and later, registered. Once the information is in the log files, humans must study it since is not prepared for the treatment by agents. The own agents cannot extract log information and the procedure cannot be automated. JADEX [8] provides a Conversation Center, which allows a user to send messages directly to any agent while it is executing and to receive answers to those messages from a user-friendly interface. The JACK [7, 12] multiagent platform supports monitoring communication between agents by means of Agent Interaction Diagrams. It also provides a Design Tracing Tool, to view internal details of JACK applications during execution, and a Plan Tracing Tool, to trace the execution of plans and the events that handle them. Other examples of tracing facilities provided by platforms are ZEUS's [13] Society Viewer and Agent Viewer, which display organizational inter-relationships among agents and their messages and agent's internal state. Also, JASON [11] provides a Mind Inspector tool to examine agents' internal state.

Apart from those tools provided by multiagent platforms themselves, there are many tracing facilities provided by third party developers. This is the case of Java Sniffer [14], developed by Rockwell Automation based on JADE's Sniffer Agent. Another third party tool based on JADE's Sniffer Agent is ACLAnalyser [16], which intercepts messages interchanged by agents during the execution of the application

and stores them in a relational database, which can be lately inspected to detect social pathologies in the MAS. These results can be combined with data mining techniques to help in the multiagent system debugging process [17]. MAMSY, the management tool presented in [18] lets the system administrator monitorize and manage a MAS running over the Magentix multiagent platform [19]. MAMSY provides graphical tools to interact with the MAS and visualize its internal state at run time. In [20], the authors describe an advanced visualization tools suite for MAS developed with ZEUS, although the authors also claim these tools could be used with CommonKADS.

As previously mentioned, the multiagent system that is proposed is based on Virtual Agent Organizations [6]. Consequently, the PANGEA platform makes it possible to create open systems that resolve the inflexibility of a multiagent system. The new open and collaborative architectures require a control focused on the interaction and global knowledge rather than autonomous behaviors. For this reason, traceability has become a key point for the distributed knowledge. As it can be appreciated, tracing facilities in MAS are usually conceived as debugging tools to help in the validation and verification processes. It is also usual to use these tracing tools as help for those users which have to understand how the MAS works. Thus, generated events are designed to be understood by a human observer who would probably use them to debug or to validate the MAS and tracing facilities are mostly human-oriented in order to let MAS users work in a more efficient and also comfortable way. Some multiagent platforms provide their own tracing facilities, although there is also important work carried out by third party developers. However, even those tracing facilities which were not designed by platform developer teams are usually designed for a specific multiagent platform. This reason leads us to integrate TRAMMAS with our platform to probe its independency and to achieve a distributed way to share knowledge between our PANGEA agents in a distributed way.

3 TRAMMAS Overview

Multiagent systems can be considered to be formed by a set of tracing entities or components which are susceptible of generating and/or receiving certain information related to their activity as trace events. A trace event is a piece of data representing an action which has taken place during the execution of an agent or any other component of the multiagent system. Each trace event has these attributes [14]:

- Event type: Trace events can be classified according to the nature of the information which they represent. This event type is necessary for tracing entities in order to be able to interpret the rest of the data attached to the trace event.
- Time stamp: Global time at which the event took place, necessary to be able to chronologically sort events produced anywhere in the multiagent system.
- Origin entity: The tracing entity which originated the event.
- Attached data: Additional data which could be necessary to correctly interpret the trace event. The amount and type of these data will depend on the event type. Some trace events may not need any additional information.

Tracing entities can be considered to be playing two different tracing roles. When they are generating trace events, tracing entities are considered Event Source entities (ES). When they are receiving trace events, tracing entities are considered Event Receiver entities (ER). Any tracing entity can start and stop playing any of these two roles, or both, at any time.

This architecture considers three different kinds of tracing entities: Agents, artifacts and aggregations.

On the one hand, agents are all those autonomous and proactive entities which define the multiagent system behavior. On the other hand, artifacts are all those passive elements in the multiagent system (databases, physical sensors and actuators, etc.) susceptible of generating events at run time or receiving them as an input [25]. Artifacts can combine in order to perform more complex tasks, generating or receiving trace events as a tracing individual. From the point of view of the tracing system, these combinations of artifacts are also modeled as single artifacts.

If the multiagent system supports aggregations of agents (or agents and artifacts), such as teams or organizations, then such aggregations are considered by the tracing system as single tracing entities, in the sense that trace events can be generated from or delivered to these entities as tracing individuals. Agents and artifacts within an aggregation are still tracing entities and thus, they can also generate and receive trace events individually, not only as members of the aggregation.

From the point of view of the architecture, the multiagent platform can be seen as a set of agents and artifacts. Therefore, the components of the platform are also susceptible of generating and receiving trace events.

When a tracing entity is playing the ER tracing role, the tracing system provides it with a stream, which can be seen as a special mailbox where the Trace Manager delivers the trace events for this ER entity. These streams can either be pieces of memory or log files. In both cases, the ER entity which owns the stream has to limit its size in order not to overload its resources.

Event types are modeled in this architecture as tracing services. A tracing service is a special service which is offered by an ES entity to share its trace events, in a similar way to a traditional service. Each ES entity can offer different tracing services, and the same tracing service can be offered by many different ES entities.

As with traditional services, when an ER entity is interested in receiving trace events of a specific event type, which are generated by a given ES, it has to request the corresponding service. From that moment on, the Trace Manager starts recording the corresponding trace events and delivering them directly to the ER stream until the ER cancels the request. The Trace Manager only records those trace events, which have been requested by an ER entity, so that no resources are spent in recording and delivering trace events, which have not been requested by any ER entity.

The Trace Manager provides a list of all the available tracing services and the ES entities, which offer them. When an ES entity wants to offer any tracing information, it must inform the Trace Manager in order to publish the corresponding tracing service so that other tracing entities can request it if they are interested in its trace events. When a tracing entity does not want to receive certain trace events anymore it has to cancel the request to the corresponding tracing service.

In order to let ES entities decide which ER entities can receive their trace events, when an ES entity publishes a tracing service, it has also to specify which agent roles are authorized to request that service to that ES entity (direct authorization). In this way, when an ER entity wants to request a tracing service to an ES, it has to be able to assume one of the authorized agent roles. ER entities which are authorized to request a tracing service to certain ES entity can also authorize other roles to request the same tracing service to that ES entity. This is defined as authorization by delegation. In this way, the tracing system maintains an authorization graph for each tracing service which is being offered by each ES. This authorization graph is dynamic, since tracing entities can add and remove authorizations at run time. When an authorization, direct or by delegation, is removed, all those delegated authorizations which depended on the removed one are also removed.

The tracing system does not control which entities can assume each role in order to request or to add authorizations for a tracing service. It is the multiagent platform which has to provide the necessary security mechanisms no prevent agents from assuming inappropriate roles.

4 Description of PANGEA Including TRAMMAS

Developing PANGEA, we are looking for a platform that can integrally create, manage and control VOs. When launching the main container of execution, the communication system is initiated; the agent platform then automatically provides the following agents to facilitate the control of the organization:

- OrganizationManager: the agent responsible for the actual management of organizations and suborganizations. It is responsible for verifying the entry and exit of agents, and for assigning roles. To carry out these tasks, it works with the OrganizationAgent, which is a specialized version of this agent.
- InformationAgent: the agent responsible for accessing the database containing all pertinent system information.
- ServiceAgent: the agent responsible for recording and controlling the operation of services offered by the agents.
- NormAgent: the agent that ensures compliance with all the refined norms in the organization. For example, preventing an agent to take an unauthorized role.
- Sniffer: manages the message history and filters information by controlling communication initiated by queries.

One of the most important features that characterize the platform is the use of the IRC protocol for communication among agents. Internet Relay Chat (IRC) is a Real Time Internet Protocol for simultaneous text messaging or conferencing. This protocol is regulated by 5 standards: RFC1459 [1], RFC2810 [5], RFC2811 [4], RFC2812 [2] y RFC2813 [3]. This allows for the use of a protocol that is easy to implement, flexible and robust. The open standard protocol enables its continuous evolution. There are also IRC clients for all operating systems, including mobile devices.

All messages include the following format: *prefix command command-parameters\r\n*. The prefix may be optional in some messages, and required only for entering messages; the command is one of the originals from the IRC standard. For the diffusion of the defined trace event taking into account the format of the IRC messages, the event attributes have been included as parameters of the messages. The communication platform is able to treat the messages according to its format and to distribute them suitably.

In line with this design, the inclusion of TRAMMAS in PANGEA is relatively easy. As previously commented, a tracing service is a special service which is offered by an ES entity to share its trace events. Therefore, the unique existing condition is that, as far as possible, an ES entity should implement its tracing service as a Web Service. This allows the ServiceAgent of PANGEA to offer the services to all the agents in the rest of suborganizations.

An *EventTracing Suborganization* has been included to create the tracing system. Figure 1 shows the agents and its relationships. This suborganization carry out the tasks that the model TRAMMAS assign to the Trace Manager. Four agents form the suborganization:

- TraceEntityAgent in charge of registering and managing all the tracing entities.
- TracingServicesAgent in charge of registering and managing tracing services offered by ES entities.
- SubscriptionAgent, which stores and manages subscriptions to each tracing service and ES entity.
- AuthorizationAgent which stores and manages the authorization needed for each tracing service and ES entity.

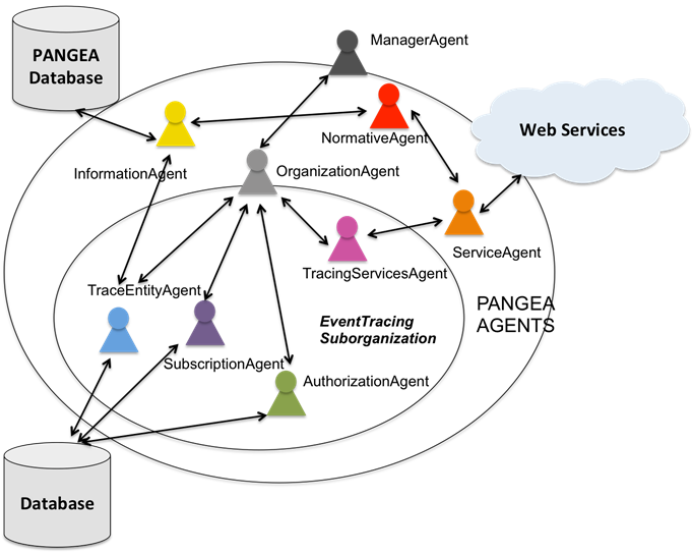

Fig. 1. Platform overview

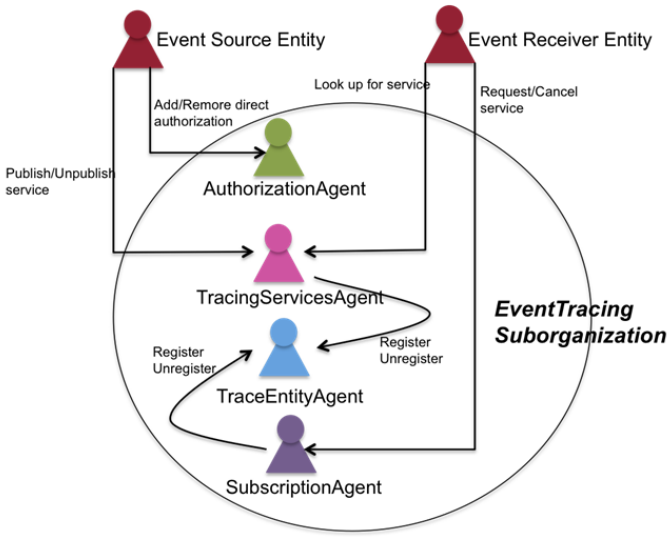

Fig. 2. Interactions between agents in the EventTracing Suborganization

Figure 2 shows how tracing entities interact with the *EventTracing Suborganization*.

5 Case Study and Results

The case study presents an example of VO, where different techniques are used to share information among agents. The agents created by PANGEA are implemented using different technologies and have different features, among which are the use of sensors. Virtual Organizations of agents are an interesting possibility to handle the large amounts of data provided by sensors because they can provide the necessary capacity to handle open and heterogeneous systems such as those normally found in the information fusion process. Several agents in the VO will be deployed on computers within a LAN and various agents will be on mobile devices.

Theoretically, the cost of transmitting the necessary information between them can be used to measure the efficiency and scalability of PANGEA platform. It also enables to compare the techniques used in the construction of each of the agents.

Let us consider a VO focuses on people detection, specifically developed for a work environment, which can facilitate tasks such as activating and personalizing the work environment; these apparently simple tasks are in reality extremely complicated for some people with disabilities [24].

ZigBee sensors are used to deploy the detection prototype. ZigBee is a low cost, low power consumption, two-way wireless communication standard that was developed by the ZigBee Alliance [22]. It is based on the IEEE 802.15.4 protocol, and operates on the ISM (Industrial, Scientific and Medical) band at 868/915MHz and a 2.4GHz spectrum.

The proposed proximity detection system is based on the detection of presence by a localized sensor called the control point which has a permanent and known location. Once the Zigbee tag carried by the person has been detected and identified, its location is delimited within the proximity of the sensor that identified it. Consequently, the location is based on criteria of presence and proximity, according to the precision of the system and the number of control points displayed. The parameter used to carry out the detection of proximity is the RSSI (Received Signal Strength Indication), a parameter that indicates the strength of the received signal. This force is normally indicated in mW or using logarithmic units (dBm). 0 dBm is equivalent to 1mW. Positive values indicate a signal strength greater than 1mW, while negative values indicate a signal strength less than 1mW [24].

In our Case Study we have a distribution of computers and laptops in a real office environment, separated by a distance of 2 meters. The activation zone is approximately 90cm, a distance considered close enough to be able to initiate the activation process. It should be noted that there is a "Sensitive Area" in which it is unknown exactly which computer should be switched on; this is because two computers in close proximity may impede the system's efficiency from switching on the desired computer. Tests demonstrate that the optimal distance separating two computers should be at least 40cm.

The agents share certain information about the state of the sensors so that other agents can carry out the detection in an optimal way. For instance, important increases in the RSSI of sensors may be indicative of a proximity to a computer and so on.

The example considers the transmission of relevant information of sensors between agents which may be interested. The internal reasoning process by which agents receive information from sensors is out of scope of this work. The case study will be considered to be in a general situation where there are n_{sens} agents in charge of controlling n sensors in the system and there is a total amount of n_{rem} remarkable situations to be reported to agents. Table I shows the number of transmissions as a function of the number of remarkable situations occurred in the system. The number of transmissions in the worst case is in the same order for both techniques (broadcast and the *EventTracing Suborganization*). However, the best case is constant for event tracing while it is higher using broadcasting.

Table 1. Summary of best and worst case costs as a function oft he number of N_{sens} agents for a constant number of remarkable situation ($K_{remarkable}$)

Number of transmissions for n_{rem} situations		
	Best Case	***Worst Case***
Broadcast	$k_{remarkable}*(2+n_{sens})$	$k_{remarkable}*(2+n_{sens})$
EventTracing Suborganization	0	$k_{remarkable}*(2+n_{sens})$

Results show that the event tracing technique provides a way to coordinate different agents in charge of sensors without having to contact directly with none of them. The amount of information interchanged among agents in the system is reduced to the minimum necessary, which makes the system more efficient and scalable.

6 Conclusions

This paper has presented a platform called PANGEA, which has been improved thanks to TRAMMAS. PANGEA has great potential to create open systems, and more specifically, virtual agent organizations. This architecture includes various tools that make it easy for the end user to create, manage and control these systems. One of the greatest advantages of this system is the communication platform that, by using the IRC standard, offers a robust and widely tested system that can handle a large number of connections, and that additionally facilitates the implementation for other potential extensions. Before TRAMMAS, the Sniffer agent offers services that can be invoked to study and extract message information but this was centralized and limited if we want to create a platform for building Large-Scale Agent-Based Systems.

TRAMMAS offers an additional indirect communication mechanism which lets agents and other entities in the system generate trace events, as well as receiving events generated by other entities. The incorporation of this model to PANGEA has improved the way in which entities and agents perceive each other and their environment, which in turn improves the way in which high-level social abstractions can be developed and incorporated to the multiagent system.

Finally, the event tracing suborganization can help reducing the amount of unnecessary information which has to be transmitted and processed, while keeping agents' internal logic as simple as possible and thus, contributing to the scalability and feasibility of VOs.

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

References

1. Oikarinen, J., Reed, D.: Internet Relay Chat Protocol. RFC 1459, IETF (1993)
2. Kalt, C.: Internet Relay Chat(Client Protocol. RFC 2812, IETF (2000)
3. Kalt, C.: Internet Relay Chat(Server Protocol. RFC 2813, IETF (2000)
4. Kalt, C.: Internet Relay Chat(Channel Management. RFC 2811, IETF (2000)
5. Kalt, C.: Internet Relay Chat(Architecture. RFC 2811, IETF (2000)
6. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Application 15(3), 200–222 (2001)
7. Agent Oriented Software Pty Ltd.: JACK^{TM} Intelligent Agents Teams Manual. s.l.. Agent Oriented Software Pty. Ltd. (2005)
8. Pokahr, A., Braubach, L.: JadexTool Guide. p. 66. University of Hamburg, Germany (2008)
9. Helsinger, A., Wright, T.:Cougaar A Robust Configurable Multi Agent Platform. In: 2005 Aerospace Conference AC 2005. pp.1–10. IEEE (2005)
10. Gruver, W.: Technologies and Applications of Distributed Intelligent Systems. IEEE MTT– Chapter Presentation, Waterloo, Canada (2004)

11. Bordini, R.H., Hübner, J.F., Vieira, R.: Jason and the Golden Fleece of Agent-Oriented Programming. In: Bordini, R., Dastani, M., Dix, J., El FallahSeghrouchni, A. (eds.) Multi-Agent Programming – Languages. Platforms and Applications, vol. 15, pp. 3–37. Springer, Heidelberg (2005)
12. Agent Oriented Software Pty. Ltd.: Jack Intelligent AgentTracing and Logging Manual, p. 85 (2008)
13. Collis, J., Ndumu, D., Nwana, H., Lee, L.: The Zeus Agent Building Tool-kit. BT Technology Journal 16(3), 60–68 (1998)
14. Búrdalo, L., Terrasa, A., García-Fornes, A., Espinosa, A.: Towards Providing Social Knowledge by Event Tracing in Multiagent Systems. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 484–491. Springer, Heidelberg (2009)
15. Tichy, P., Slechta, P.: Java Sniffer 2.7 User Manual (2006)
16. Botia, J., Hernansaez, J., Skarmeta, F.: Towards an Approach for Debugging MAS Through the Analysis of ACL Messages. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 301–312. Springer, Heidelberg (2004)
17. Botia, J., Hernansaez, J., Gomez-Skarmeta, A.: On the Application of Clustering Techniques to Support Debugging Large-Scale Multi-Agent Systems. In: Bordini, R., Dastani, M., Dix, J., El FallahSeghrouchni, A. (eds.) PROMAS 2006. LNCS (LNAI), vol. 4411, pp. 217–227. Springer, Heidelberg (2007)
18. Sanchez-Anguix, V., Espinosa, A., Hernandez, L., Garcia-Fornes, A.: MAMSY: A Management Tool for Multi-Agent Systems. In: Demazeau, Y., Pavón, J., Corchado, J.M., Bajo, J. (eds.) PAAMS 2009. AISC, vol. 55, pp. 130–139. Springer, Heidelberg (2009)
19. Alberola, J., Mulet, L., Such, J., Gacía-Fornes, A., Espinosa, A., Botti, V.: Operating System Aware MultiagentPlatform Design. In: 5th European Workshop on Multi-Agent Systems (EUMAS 2007), pp. 658–667 (2007)
20. Ndumu, D.T., Nwana, H.S., Lee, L.C., Haynes, H.R.: Visualization and Debugging of Distributed MultiagentSystems. Applied Artificial Intelligence 13(1-2), 187–208 (1999)
21. Helsinger, A., Thome, M., Wright, T.: Cougaar – AScalable, Distributed Multi-Agent Architecture. In: IEEE International Conference on Systems, Man and Cybernetics (SMC 2004), vol. 2, pp. 1910–1917. IEEE (2004)
22. ZigBee Standards Organization: ZigBee Specification Document 053474r13. ZigBee Alliance (2006)
23. Bellifemine, F., Caire, G., Trucco, T., Rimassa, G., Mungenast, R.: JADE Administrator's Guide. TILAB (2007)
24. Villarrubia, G., Sánchez, A., Zato, C., Bajo, J., Rodríguez, S., Corchado, J.M.: Proximity Detection Prototype Adapted to a Work Environment. In: Novais, P., Hallenborg, K., Tapia, D.I., Rodríguez, J.M.C. (eds.) Ambient Intelligence - Software and Applications. AISC, vol. 153, pp. 51–58. Springer, Heidelberg (2012)
25. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&AMeta-Model for Multi-Agent Systems. Autonomous Agents and Multi-Agent Systems 17(3), 432–456 (2008)