Practical Applications of the Web-Based Agent Platform 'Eve'

Ludo Stellingwerff, Jos de Jong, and Giovanni E. Pazienza

Almende BV,
Westerstraat 50, Rotterdam 3016 DJ,
The Netherlands
{ludo,jos,giovanni}@almende.org

Abstract. The existing approaches to build multi-agent systems fail at addressing the challenges posed by the current technology, where ubiquitous interconnected electronic devices are no more passive machines operated by humans but rather active computational components cooperating with humans. In order to tackle this problem, we have created a novel open-source web-based agent platform called 'Eve' that features some specific characteristics (e.g., platform and language independence, openness) that make it particularly suitable to be deployed in real-life applications. In this paper, we discuss the main features of Eve and present several use cases in which it has been successfully applied.

Keywords: agent platforms, multiagent systems, interoperability.

1 Introduction

Nowadays, both humans and software applications can be considered as entities with some degree of autonomy that interact with each other and with the environment without need of centralized coordination. In this framework, devices are modelled as agents that mimic some of the characteristics of human beings: they are autonomous (i.e., capable of taking decisions), intelligent (i.e., capable of adapt their behaviour on the basis of available data), and social (i.e., capable of communicating with humans as well as other agents). Therefore, the fundamental tool to handle (and profit from) such a complex yet promising scenario is to build an effective multi-agent system (MAS) [11] tailored for this novel technological context. A MAS is usually developed on an agent platform, which is usually chosen among the several dozens already available on the market (a good yet not recent overview can be found in [13]). Nevertheless, the great majority of them – usually Java based – consist of a closed and controlled environment (operating system or simulation environment) where agents can live and interact with each other. In general, these platforms suffer from connected to the scalability and to the robustness, mainly because of the presence of central directory services (which require memory to be stored) and to the amount of manual setup to add new agents, especially when the platform runs in a heterogeneous environment.

Y. Demazeau et al. (Eds.): PAAMS 2014, LNAI 8473, pp. 268–278, 2014.

In order to overcome these problems, we propose 'Eve' [8] that is an agent platform specifically thought to be deployed on a diverse distributed environment. Eve is inspired by the principles of human-agent collectives described CHAP [17] and in ORCHID [15], in which agents (both human and software) collaborate in a seamless and effective way. Eve has been the key component of several successful applications, especially in the fields of emergency management, smart grids, energy consumption optimisation, and coordination of complex tasks. In this paper, we also discuss several of these practical use cases, emphasising that they are just a few of those in which such innovative agent platform may find application.

The paper is structured as follows: in Sec. 2, we describe the architecture and the main features of Eve; in Sec. 3, we illustrate a few practical examples in which Eve has already found application; in Sec. 4, we draw the conclusions of our work.

2 Summary of the Main Features of Eve

2.1 Conceptual Architecture

From the architectural overview of Eve shown in Fig. 2, it is possible to catch a glimpse of its main features. It should be evident that Eve can be deployed on a number of different devices (smartphones, servers, local PCs, etc.) hosting different environments (JavaScript, Android, C++, etc.), all connected to the internet via the JSON-RPC protocol.

Thanks to this approach, Eve has some distinctive characteristics. First of all, Eve is platform independent since agents can live on any device: smartphones, robots, servers or, more generically, in the cloud. Second, Eve is language independent[1] because it dictates only the communication protocol (JSON-RPC) which works over existing transport layers (HTTP, XMPP) and a simple API, as described in and Table I. Third, Eve is an open agent platform: each agent has its own public[2] URLs and hence existing Eve-systems can be easily connected to the others. Furthermore, non-Eve systems can be connected to the Eve platform by making its API available via an Eve agent acting as a wrapper.

The architecture of Eve leads to further advantages among which is worth to emphasize the following ones:

Scalability: Eve is fully web-based and hence, from a practical point of view, there is no upper bound to the number of new agents that can be added to the system without degrading its performances.

Robustness: the state persistency in Eve is offloaded from the agents to the environment (i.e., the states of the agents are distributed), which makes Eve

[1] Currently, there are two mature implementations of Eve (one in Java and the other in JavaScript) whereas a third one (in Python) is in an embryonic stage.

[2] Authorization mechanisms have already been implemented, in case the application needs them.

insensitive to server/device failures. Although the network transport layer can contain single points of failure, Eve itself is a distributed agent platform, with no single point of failure by design.

Massive parallelization of the workload of an agent: agents can be multiplexed (i.e., multiple instances of the same agent sharing the same state can exist at once). Traditional agent platforms have 1 thread per agent; in contrast, Eve allocates up to n threads (where n has virtually no upper bound) when the agent is heavily loaded and no thread at all when the agent is idle. This approach has the additional advantage of reducing resource consumption for idle agents.

Seamless migration: In Eve, there is no difference in accessing local or remote agents, as agents are fully location agnostic. This feature allows seamless migration of agents between run-time environments.

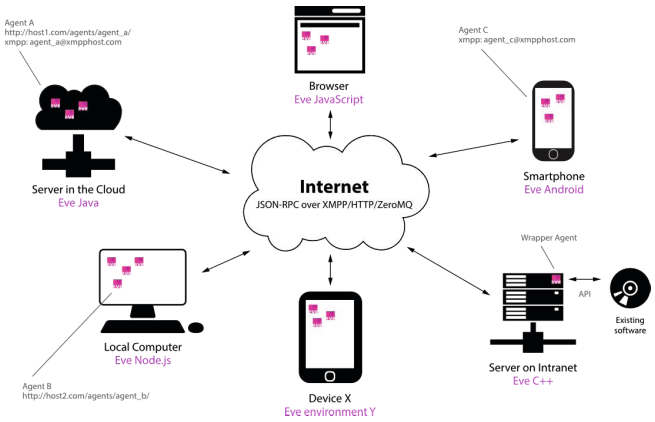

Fig. 1. Conceptual architecture of the Eve, showing that it can be deployed in a heterogeneous environment where different devices are connected to the internet via the JSON-RPC protocol

Eve is a fully decentralized system: there is neither central coordination nor centrally-stored list of all available agents. Interactions among agents are asynchronous and request-driven: agents get to know each other via the so-called shared services (e.g., acting in the same calendar or registering at the same locations service). Eve agents communicate via regular HTTP POST requests or via XMPP messages through the JSON-RPC protocol, which is a simple protocol using JSON (JavaScript Object Notation) to format requests and responses.

2.2 What Is an Eve Agent?

Defining what exactly an agent is has been a long-standing issue. However, there is a general agreement over the basic capabilities that an agent needs, as proposed in [19]:

- *Autonomy*: agents should be able to perform the majority of their problem-solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state.
- *Social ability*: agents should be able to interact, when they deem appropriate, with other software agents and humans in order to complete their own problem solving and to help others with their activities where appropriate.
- *Responsiveness*: agents should perceive their environment (which may be the physical world, a user, a collection of agents, the Internet, etc.) and respond in a timely fashion to changes which occur in it.
- *Proactiveness*: agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate.

This description fits very well to the way in which Eve agents have been devised; in particular, an Eve agent consists of: i) *code* containing the agents logic, which allows an agent to perform its tasks, take initiatives, learn, react, cooperate, negotiate, etc.; ii) *communication facilities* allowing the agent to interact with other agents (currently over HTTP and XMPP); iii) *clock*, enabling the agent to schedule tasks for itself; iv) *memory*, a place where the agent can store its state and history.

All Eve agents have a set of standard methods available, described in detail in Table 1. In particular, we have created methods to retrieve the agent id (*getId*), type (*getType*), version (*getVersion*), and description (*getDescription*). Also, there is a method to get all URLs of an agent (*getURLs*), and to subscribe (*onSubscribe*) or unsubscribe (*onUnsubscribe*) from the agent events.

2.3 Short Notes about FIPA Compliancy

Eve has been conceived to allow developers easy access to agent concepts – such as time autonomy and direct interagent communication – and its architecture relies in large part on existing technologies and infrastructures. These choices have led us to a pragmatic approach concerning the a priori compliancy to FIPA specifications. As a result, Eve is not fully FIPA compliant by design.

A through discussion of the motivations behind this choice is beyond the scopes of this paper. However, we would like to emphasise that the last version of FIPA specifications is from 2000 (revised in 2002) [2], and that in some cases they do not align with the modern netcentric approach of agent systems. For instance, FIPA specifications mandates hat an agent system defines a directory facilitator; Eve goes beyond this concept, letting agents know each other via shared services, thus avoiding the need for a global registry of agents. Also, it is the underlying

Table 1. Set of standard methods available to all Eve agents

Method	Description
getId	Retrieve the agent id. In Eve, an agent may have multiple URLs but only one id. The agent id is not globally unique, since agents running on different platforms may have the same id.
getType	Retrieve the agent type, which is typically the class name of the agent.
getVersion	Retrieve the agent version number.
getDescription	Retrieve a description of the agent functionality.
getURLs	Retrieve an array with the agent URLs. An agent can have multiple URLs for different transport services, such as HTTP and XMPP.
getMethods	Retrieve a list with all available methods.
onSubscribe	Subscribe to an event of this agent; the provided callback URL and method will be invoked when the event is triggered.
onUnsubscribe	Unsubscribe from one of this agent events.

assumption of FIPA that agent- systems should form a common global ontology and a common API using a globally accepted language. This strict coupling (at specification level) may even limit the interoperability between agent-systems and complicate the development. In our experience, it is more effective favour loose coupling between agents and bridge systems by injecting ontology-mapping translation agents, using ad hoc (but well documented) languages.

There is some evidence [1] that FIPA has a roadmap for introducing several compliance levels: from minimal-FIPA compliance level, which represents the lowest requirements, up to a full-FIPA compliance level, which comprises all current mandatory parts of normative specifications. This process is still ongoing, but our belief is that Eve will have an intermediate-FIPA compliance level.

2.4 On the Comparison with Related Approach

Nowadays, there are numerous widely-employed agent platforms, such as JADE [7], AgentScape [16], and A-globe [18], only to mention a few. Performing a throughout comparison between Eve and all of them is out of the scopes of this paper, which is rather focused on the practical applications of Eve. However, it is worth to mention that a fair comparison of Eve with existing agent platforms is difficult to make, mainly for two reasons. First, the core concepts of Eve are different from those of traditional agent platforms: Eve is fully web-based and lacks of any 'centralised' feature, and its main strengths are in the fact of beings platform-independent and easily deployable, which are difficult to translate into numbers. Second, there is very little literature about the comparison of multi-agent systems: some recent works (e.g., [9]) makes some original proposal – based on some classical works [12] – which is though not applicable to Eve because Eve has been built to operate in different context; [10] proposes

yet another approach, which though focuses on a very particular aspect possibly missing the overall evaluation. Curiously, some concrete effort to define common criteria to evaluate different systems was made in the past (see [14]) but the current technological framework condemns to early obsoleteness any effort of this sort.

Still, in the near future we plan to publish an ad-hoc paper dedicated to this issue which may help to define at least some preliminary metrics to evaluate the performances of Eve.

3 Examples of Practical Applications of Eve

Eve has already been used in several commercial and research projects covering different scientific fields. In the following, we describe a four existing practical applications of Eve, even though more are expected to be implemented in the near future. What we report here is a summary of the work carried out; more details and demos can be found on the Eve website [6].

3.1 Autonomous (Re)scheduling of Appointments

In the last two decades, there has been a major shift towards calendaring software to keep track of events and (re)schedule appointments. Especially when several people are involved, these tasks can be a tedious time-consuming burden as, in general, different calendaring softwares do not interact seamlessly with each other. As a consequence, people waste a considerable amount of time *scheduling* rather than *doing*.

In order to tackle this issue, Eve has been successfully used to build an agent-based scheduling system, whose conceptual architecture is shown in Fig. 2. It is composed of four kinds of agents:

- *personal agents*, which are the virtual counterpart of each user, and hence they represent the users' preference and learn autonomously the user preferences (e.g., working shifts) based on the past meetings;
- *meeting agents*, which negotiate and schedule the appointments based on the information coming from all other agents;
- *context agents*, which have access to the users' calendar by communicating with the servers (Gmail, Exchange, etc.) and to other context data – such as traffic conditions and weather – that may influence the behaviour of the users.
- *location agents*, which are a special kind of context agents whose sole task is connecting to location services and calculate travel times between user locations.

Whenever needed, a meeting agent negotiates with the personal agents of all participants as well as with other existing meeting agents in order to find the most convenient time slot and location for the meeting, and reschedule it when the circumstances change. This process is totally transparent to the user,

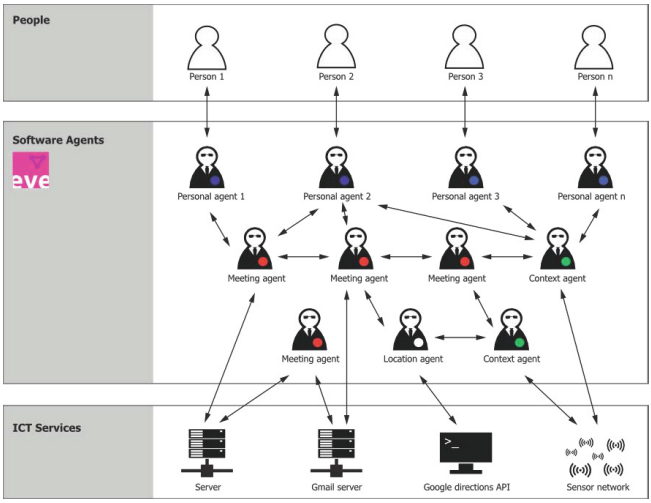

Fig. 2. Conceptual architecture of the Eve-based autonomous scheduling multiagent system

who may – but is not required to – interact with his own personal agent via smartphone apps, web interfaces, instant-messaging (e.g., chat), and standard telephone through Interactive Voice Response. Therefore, as a result of the application of Eve, the slow and cumbersome communication between humans, or between a human and his calendaring software, can be substituted (or at least facilitated) by the fast and seamless communication between software agents.

Eve is specifically suited to this application for a number of reasons: first, it is language- and environment- independent and hence Eve agents can be easily developed for the different OS hosted on the personal devices; second, the fact that Eve is fully decentralised allows this system to rely on a very limited amount of messages, as devices (better to say, *personal agents*) can autonomously take peer-to-peer actions, without communication to a central unit; third, Eve agents are lightweight, and thus particularly fit to devices which may have limited memory and processing power.

3.2 Emergency Management

Natural disasters and accidents do not take into account municipal or national borders; in fact, in case of a large-scale incident, such as a terrorist attack or a major fire, different agencies from different regions or even countries have to work together and it is often very difficult, because of incompatible systems, organizational structures and protocols. This is the main motivation behind the BRIDGE [4] project, whose main goal is to develop a platform to provide technical support for multi-agency collaboration in large-scale emergency relief efforts, taking care of their (IT) systems, people, and protocols. In this framework, agent

technology is very useful to solve communication and collaboration problems, especially because a multi-agent system can make sure that the right information is passed to the right agencies and that the relevant experts are involved. Therefore, a passive communication infrastructure can be transformed into an active system that can initiate connections.

In the BRIDGE project, Eve is used to model and support the various emergency response resources involved: different software agents are used to represent interests of different parties, and thus quickly negotiate and collaborate. Also, the software agents are mainly meant to perform rapid, repetitive tasks, such as quick communication or aggregation of data whereas people are involved when difficult tasks must be executed or decisions made. Through negotiation between Eve agents, ad hoc groups of resources are formed to handle tasks. The agents also obtain, aggregate and interpret sensor data, which is then communicated to field commanders and to other agents, thus providing situational awareness. The Eve agents are run on a cloud platform and on mobile smartphones, whereas the communication between the Eve agents and other components of the BRIDGE platform is done through the EDXL-RM protocol.

There are at least two key aspects of Eve emphasised in this application: first, the fact that being language- and platform- independent Eve can act as a middleware among devices of different nature; second, the fact that the combination of local and cloud counterparts of Eve agents makes the system particularly robust to infrastructural failures (which are critical in emergency managing situations) as well as to the fact that portables devices can be switched off or out of battery (which would allow the 'cloud' agents to be still active anyway).

3.3 Energy-Efficient Data Centres

Renewable energy sources – such as wind, water and solar energy – are not predictable and hence power suppliers can therefore not guarantee a completely 'green' energy supply during peak hours. In case of high demand, they are forced to rely on dirty energy produced by diesel-fuelled generators, or they have to transport energy from faraway sources. In the case of data centers, it would be ideal to estimate their power needs in order to allow power suppliers can better anticipate future energy demands. This is the rationale behind the All4Green project [3], which brings together relevant stakeholders to create a 'sustainable ICT ecosystem' for the datacenter sector. By enabling datacenters, power suppliers and end-users to communicate their expected supply and demand, ICT resources can be better allocated to provide requested services, while saving energy and reducing greenhouse gas emissions.

In the All4Green project, Eve is used to negotiate the most efficient energy balance between energy providers and large- scale energy users, specifically data centers. The context of this negotiation process is given in the form of GreenSLA contracts, which offer a template for energy profiles of data centers. By communicating the expected energy profiles, the energy providers provide a goal for the Eve agents to work towards. Also, Eve has a communication middleware role: the toolset has offered an in-place replacement of a Springframe work SOAP stack,

still adhering and actively enforcing the (Java Interface) contract between the communicating partners; this is achieved through the proxy-agent generation of Eve. The way in which Eve has been used in this project also shows that Eve can work properly in a strict Java EE infrastructure.

Similarly as before, the role of Eve as a communication middleware is emphasised in this application; also, the fact that Eve agents are proactive and are able to carry out autonomously even complex negotiation tasks is particularly critical too.

3.4 Dynamic Management of Smart Grids

The energy market suffers from structural inertia: in theory, energy prices should follow a standard supply-and-demand-mechanism; in practice, the market is not able to adapt to the rapid changes in supply and demand of energy. One of the reasons behind this mechanism is that the current energy grid is still based on a centralized and inflexible management of supply and demand of energy, but the modern energy market is becoming much more dynamic. The current grid is not prepared for a distributed energy market, in which individual households cause energy supply peaks by generating their own solar power on a sunny day. Managing this scenario by applying an Internet-of-Things-approach is the motivation of the INERTIA project [5], which that aims to model and use the energy flexibility of terniary buildings to lower peak electricity demand. At each building, INERTIA will support the building manager with making optimal decisions and taking limited control of Distributed Energy Resources (DERs). Many of such buildings will be connected with a hub, where energy flexibility can be pooled and traded with energy providers (or other consumers).

Within INERTIA, Eve is used as an integration platform for the software that will run at each building. This is achieved by wrapping numerous pieces of existing and new software within Eve agents, elegantly separating the means of communication from the functionality within the agents. As the different software components are written in different languages, the fact that Eve is language independent can ease development. As some of the software components implements cross-cutting concerns, the Eve based architecture also performs a function similar to aspect-oriented programming: it facilitates loose coupling between the agents implementing the cross-cutting concern and the rest of INERTIA. Moreover, agents that consist of small bits of logic such as a learning algorithm can be instantiated as often as required effortlessly.

Besides Eve as an architectural solution, Eve agents are also used to represent and interact with the physical reality: users, DERs, and spaces (which can contain multiple users and DERs). Such an agent representation will allow the end users to acquire a complete breakdown of the aggregated energy use and flexibility to the level of individual consumers and DERs, helping analysis and improvement of energy performance. One of the key benefits of using Eve agents is their ability to migrate: agents representing humans can associate with different space agent to represent moving from one room to another and migrate to a whole new server to represent moving from one office building to the next (e.g., when changing jobs).

Finally, ongoing development of Eve implementations are expected to allow the user interfaces of the agents to be conveniently dynamic and modular.

This application relies on all aspects of Eve emphasised in the previous three applications: its role as a middleware among different platforms; the fact that it is language-independent and lightweight, which then makes it particularly suited to be deployed in an heterogeneous Internet-of-Things approach; its robustness, which is particularly critical when managing electric grids.

4 Conclusion

In this paper, we have introduced the main characteristics of Eve, a novel agent platform that has been explicitly devised to be applied in the current technological framework composed of a multitude of devices based on different platforms and programming languages, and described a few successful practical applications of it.

Eve is intrinsically platform- and language- independent and it can be considered as one of the precursors of a new generation of agent platforms, which will be drastically different from (and hence difficult to compare to) the current ones. Among the other features, we want to emphasize that Eve is a fully decentralized open agent platform: agents have their own public URLs but there is no centrally- stored list of all available agents. These characteristics make Eve very scalable (one could claim that the system is as scalable as the web itself) and robust. As future developments of Eve, we plan to extend the number of working implementations beyond the current ones, which are in Java and JavaScript.

Currently, Eve has been applied to several areas, including emergency management, smart grids, energy-efficient data centers, and context interpretation. In each of them, Eve has proved to be an effective solution to problems that otherwise would have been hard to solve, especially those connected to the interoperability of heterogeneous devices. Among the new applications on which we are currently working, we can mention a new system to capture and interpret non-conformity events in a large manufacturing company where Eve will be used as the infrastructure for an Internet of Things approach.

Acknowledgement. This research has been partially funded by the European Union Seventh Framework Programme under grant agreements no. 261817 (BRIDGE project), no. 288674 (All4Green project), and no. 105543 (INERTIA project). The authors are grateful to Andries Stam, Rick van Krevelen, Remco Tukker, and Janny Remakers for their support writing this paper.

References

1. Minimal FIPA and FIPA compliance levels work plan (2001), `http://www.fipa.org/docs/wps/f-wp-00018/f-wp-00018.html`
2. FIPA specifications (2002), `http://www.fipa.org/specifications/`
3. All4Green project (2012), `http://www.all4green-project.eu/`

4. Bridging resources and agencies in large-scale emergency management, BRIDGE project (2012), http://www.bridgeproject.eu/en
5. INERTIA project (2013), http://www.inertia-project.eu/
6. Eve - a web-based agent platform (2014), http://eve.almende.com/
7. Bellifemine, F., Poggi, A., Rimassa, G.: Jade: a fipa2000 compliant agent development environment. In: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 216–217. ACM (2001)
8. de Jong, J., Stellingwerff, L., Pazienza, G.E.: Eve: a novel open-source web-based agent platform. In: 2013 IEEE International Conference on Systems, Man and Cybernetics. IEEE (2013)
9. Di Bitonto, P., Laterza, M., Roselli, T., Rossano, V.: Evaluation of multi-agent systems: Proposal and validation of a metric plan. In: Nguyen, N.T. (ed.) Transactions on CCI VII. LNCS, vol. 7270, pp. 198–221. Springer, Heidelberg (2012)
10. Ben Hmida, F., Lejouad Chaari, W., Tagina, M.: Performance evaluation of multiagent systems: Communication criterion. In: Nguyen, N.T., Jo, G.-S., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2008. LNCS (LNAI), vol. 4953, pp. 773–782. Springer, Heidelberg (2008)
11. Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous Agents and Multi-agent Systems 1(1), 7–38 (1998)
12. Kusek, K.J.G.J.M.: A performance analysis of multi-agent systems. International Transactions on Systems Science and Applications 1(4) (2006)
13. Leszczyna, R.: Evaluation of agent platforms. European Commission, Joint Research Centre, Institute for the Protection and Security of the Citizen, Ispra, Italy, Tech. Rep. (2004)
14. Occello, M., Guessoum, Z., Boissier, O., et al.: Un essai de définition de critères pour l étude comparative de plates-formes multi-agents. Technique et Science Informatiques (TSI) 21(4) (2002)
15. U. of Southampton. The ORCHID project. (2014), http://www.orchid.ac.uk/
16. Overeinder, B.J., Brazier, F.M.T.: Scalable middleware environment for agent-based internet applications. In: Dongarra, J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 675–679. Springer, Heidelberg (2006)
17. Serban, R., Guo, H., Salden, A.: Common hybrid agent platform–sustaining the collective. In: 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), pp. 420–427. IEEE (2012)
18. Šišlák, D., Rollo, M., Pěchouček, M.: *A*-globe: Agent platform with inaccessibility and mobility support. In: Klusch, M., Ossowski, S., Kashyap, V., Unland, R. (eds.) CIA 2004. LNCS (LNAI), vol. 3191, pp. 199–214. Springer, Heidelberg (2004)
19. Wooldridge, M., Jennings, N.R.: et al. Intelligent agents: Theory and practice. Knowledge Engineering Review 10(2), 115–152 (1995)