Improving the Interoperability in Multi-agent Systems

Corchado J. M. and Cuesta P.

Artificial Intelligent Research Group
Department of Languages and Computing Systems
University of Vigo, Campus As Lagoas 32004, Ourense, Spain
Email {corchado, pcuesta}@uvigo.es
Tlf.: +34 988387010 Fax.: +34 988387001

Abstract. During the design and development of multi-agent systems, all aspects related to interoperability among each of the individual agents, that form part of the distributed system, require special attention. This paper shows how communication agents facilitate the interoperability between the components of multi-agent systems. These agents are in charge of the communication between other agents. They increment the flexibility of distributed systems and facilitate the incorporation of new agents into such systems. A multi-agent advisory system is an example of an architecture of this type and it is presented in this paper.

1 Introduction

This paper presents a multi-agent system architecture that facilitate the communication and co-operation of heterogeneous components. Multi-agent systems represent a new way of analysing, designing and implementing complex software systems. These systems are composed of several interacting agents. Models of interactions are co-operation, co-ordination and negotiation. The flexibility and high-level nature of these interactions provide the power of this paradigm.

An agent is a computer system, situated in some environment, which is capable of flexible autonomous action in order to meet its design objectives [1]. An autonomous agent is a system situated within, and a part of, an environment that senses that environment and acts on it over time in pursuit of its own agenda and so as to effect what it senses in the future [2].

The term agent is used to denote a computer systems that enjoys the following properties [1, 3]:

- Autonomy; agents operate without the direct intervention of humans (or other agents) and have control over their own actions and internal state. In a stronger sense, they are systems capable of learning from experience.
- Social ability; agents interact with other agents in order to complete their own problem solving and to help others with their activities.

- Reactivity; agents perceive their environment and respond to changes that occur in it. Agents receive sensory inputs from the environment and they can perform actions in order to change the environment in some way.
- Pro-activeness; agents are able to exhibit opportunistic, goal-directed behaviour, and take the initiative where appropriate.

The presence of these four properties in a software entity provides the power of the agent paradigm and distinguishes agent systems from others software paradigms.

In addition to having these properties, agents could be modelled using concepts that are more usually applied to humans. Shoham describes agents as entities with mental states such as believes, capabilities, choice and commitments. These states determine the actions that the agents perform and which can be affected by the information that they receive [4].

Since multi-agent systems could be composed of agents implemented in different programming languages and which could be running in different execution environment, it is necessary to define mechanisms that guaranty the adequate communication among the agents of the system in order to achieved the required functionality.

This paper shows how facilitators (communication agents) can be used in multi-agent systems to guaranty the interoperability between their components. Facilitators contain information about the rest of the agents of the system, including their location, low level communication protocol, and the KQML performative [5] subset used by the system.

Facilitators have been used for the implementation of a multi-agent Advisory System (MAAS). The aim of this system is to advise the students of the computing science degree of the University of Vigo about the optional subjects that they should enrolled.

Following, the paper describes the multi-agent system created in the framework of this investigation and discusses the role of the agent in charged of the communication among other agents.

2 Multi-agent Advisory System

The multi-agent system includes an advisory agent which uses a case-based reasoning (CBR) system [6] to help students to select the most suitable subjects by taking into consideration their background and previous experiences. Identifying and selecting the subjects that may be more interesting or more useful for a student is a complex task. The Spanish educational system allows students enrolled in a university degree to select a number of "optional" subjects. The "optional" subjects must complement the compulsory subjects and give the students the opportunity to specialise in a particular area. Although there are students that know what they want to study in order to achieve their academic goals, many of them have problems to select the most suitable subjects. All students must enrol within all the compulsory subjects during the first year; the following years they can enrol in subjects which they want and in the order that they desire.

The architecture of the Multi-agent advisory system has been influenced by the architecture of the Project Monitoring Intelligent Agent (PMIA) system [7]. The aim of the PMIA was to develop an architecture that facilitates the design, implementation and communication of heterogeneous autonomous dedicated agents. The intention of the authors of the PMIA system was to study the learning abilities of autonomous agents that incorporated either symbolic or connectionist problem solving mechanisms.

The PMIA system [8] was composed of several advisory agents, which are capable of allocating the most appropriate academic member of the staff to supervise an undergraduate or postgraduate project. The advisory agents employ case based reasoning and artificial neural networks (ANN) mechanisms. These agents identify an appropriate supervisor for a student and also support project management by scheduling students/lecturers meeting. This structure allowed the study and comparison of artificial neural networks and case based reasoning systems and showed how they could be used to increase the autonomy of Agents. This system design was influenced by the agent architecture proposed by the Knowledge Sharing Effort, used KQML and was implemented using HTML, Perl and Visual C++. The simplicity, effectiveness and flexibility of this model have encouraged us to use this architecture to build our application.

The Multi-agent advisory system (MAAS) includes personal agents (or assistants) associated to students and lecturers, an advising agent which uses a case-based reasoning system [6] to provide advice to students, an information agent and a network of facilitators or contact agents that facilitate the interaction between agents and with the users. This system design is also influenced by the agent architecture proposed by the Knowledge Sharing Effort [5], using the Knowledge, Query and Manipulation Language (KQML) for agent communication.

2.1 Components of the MAAS system

MAAS is composed of the following elements: Teacher agent generators, Teacher agents (or assistants), Student agent generators, Student agents (or assistants), Facilitators, an Advisory agent, and an Information agent. Figure 1 presents a diagram of the Multi-agent Advisory System.

Facilitators:
The agents that form part of this system use facilitators to enable the communication among them. This system follows the standards of the Agent Communication Language (ACL) in order to exchange information, about their needs and capabilities,

with their local facilitator. These facilitators are in charge of searching for the most adequate path to send the information to other facilitators, these facilitators pass this information to the agents within their domain, and that are capable of satisfying the request. Each agent includes a router capable of sending and receiving messages using KQML, identifying a set of performatives (words) and satisfying a protocol. These routers are independent processes, so the communication of agents is asynchronous.

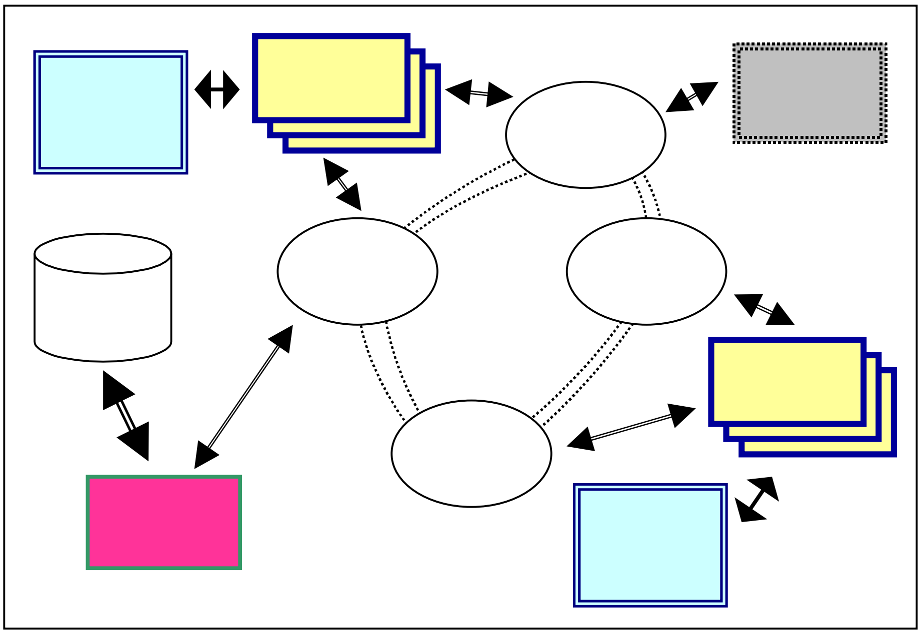

Fig. 1: Architecture of the MAAS.

Teacher Agents:

Each lecturer is associated to a Teacher Agent (or assistant). The Teacher Agent knows which subjects are taken by the lecturer associated to it. The agent obtains this information by interrogating the lecturer. This information is stored in the virtual knowledge base of the agent (VKB). The lecturer can modify this information at any time sending the agent new information about the subjects that he/she is responsible for. They have been implemented using Java and are platform independent. Lecturers interact with their agents via Java applets and agents contact the lecturer by the email system.

Student Agents:

Given that there are Teacher Agents, equally, there are Student Agents (or assistants) associated with students. Student Agents work in a similar way as Teacher Agents, such that each Student Agent is dedicated to a particular student. The Student Agent learns about the background, areas of interest, and academic activities of the student. This information is stored in the virtual knowledge base of the agent (VKB), together with the advises provided by the Advisory Agent or the Teacher Agent. They have been implemented using Java and are platform independent. Students interact with their agents via Web pages and agents contact them back by the email system. When a lecturer or student requires a personal agent; it is created and associated to him/her.

Teacher and Student agent generators:

Lecturers and students may contact the Teacher or the Student agent generators using a web browser and request it to create a new personal agent; after which the Teacher or the Student agent generators generate a new teacher agent or a new student agent and send it to the teacher's or student's computer system. The Teacher and Student Agent Generators are only actuators that automate the process of setting up the Teacher and Student assistants.

Information Agent:

The task assigned to the information agent is to obtain information about the lecturers, students and to subjects and store it in a central database. This agent receives information related to their background, personal interest, courses that they have attended, etc. This information is stored and retrieved from the database when the Advisory agents need it. The information agent can only access the database; this constraint makes it easier to maintain a consistent database.

Advisory Agent:

This agent uses a case-based reasoning system [9] to help the students to select the most suitable optional subjects. This agent refreshes its case-base after the exam period, when new students arrive or when a student needs to update his/her information. When a student makes a request for advice, the contact agents selects the appropriate advisory agent and it sends back to the student the advice. The advisory agents can access the information of the central database via the information agent.

2.2 Case-based Reasoning System

A Case-based reasoning system is composed of 4 different stages: retrieval, reuse, review and retain. The case based reasoning system obtains data from a case-base that contains information about the student's background and academic records. A CBR system is a cyclical problem solving reasoning mechanism in which the four steps mentioned above are run sequentially [9, 10].

In this case, during the retrieval stages the background and academic record of the student asking for advice are compared with the cases stored in the case base. K-nearest neighbour algorithms are used to select the most similar cases. The cases, which are most similar to the new case, are selected. The Euclidean distance between the retrieved cases and the problem case is used to refine the initial selection of cases and a number of subjects are initially recommended, during the reuse step. This initial advice is reviewed before it is sent back to the student. During the review stage the background of the student and his/her personal aptitudes are compared with features of the recommended subjects such as: degree of complexity, orientation (practical, theoretical or both), requirement of knowledge of foreign languages, need for programming skills, knowledge of different hardware (PC, Workstation, Mackintosh), etc. The outcome of the review process is the set of subjects that are recommended to the student.

Learning is achieved when the student is evaluated after attending the recommended subjects. The marks obtained in the subjects show the students performance and ultimately the effectiveness of the advising mechanism. The outcome is used to eliminate or modify cases and to prune the case base. If the CBR does not have cases that match the characteristics of a particular student; the advisory agent communicates it to a predefined lecturer who studies the case and advises the student.

2.3 Sequence of Operation

Lecturers and students can contact their correspondent teacher or student agent generator and apply for a personal agent. The communication between lecturers and students with the teacher or student agent generator is carried out via Java applets. Lecturers who are teaching one or more subjects introduce information about the subjects that they are teaching via their associated assistant. This information is stored in a centralised database and recovered by the advisory agents when required. The students introduce also information related to their background, areas of interest and academic achievements. They introduce this information to the database via their student assistants. Both Lecturers and Students communicate with their assistant agents via Java Applets.

When a student is interested in advice, it is communicated to his/her personal student assistant. The student assistant sends a request for advice to the corresponding advisory agent and this agent sends back the advice. The facilitators (contact agents) associated to each student agent establish the communication between the agents. If the Advisory agent is not capable of providing a reasonable answer, because there are either contradictory or not sufficient cases in its case-base, the advising agents communicates with a predefined lecturer (each group of students has assigned a tutor lecturer), and the lecturer will study the individual case and send the advice to the student. Student and lecturers can also contact each other to arrange meeting between them using the mechanisms provided by the multi-agent system.

2.4 Agents Communication

Several communication paths can be stabilised between agents:

- *Student* ⇔ *Student assistant*
  The students contact, via Java applets, their Student Agents when they require advice. Then the agents reply with the advice.

- *Student assistant* ⇔ *Facilitator* ⇔ *Advisory Agent*
  The Student Agent sends a message to the Advisory Agent asking for advice via the facilitator, and the Advisory Agent replies with the advice, also via the facilitator.

- *Teacher assistant* ⇔ *Lecturer*
  The Teacher Agent informs the lecturer that the Advisory Agent can not provide advise, then the lecturer studies the particular case and provides the advice to his/her corresponding personal agent. The lecturers communicate with their agents using Java applets.

- *Student assistant* ⇔ *Facilitator* ⇔ *Teacher assistant*
  The Teacher Agent sends the advice (provided by the lecturer) to the Student Agent.

The language used, in the MAAS system, for the communication between agents is the KQML [5, 11]. This language is based in a transport model of point-to-point message passing.

3 Communication between agents using facilitators

There are many different types of multi-agent system architecture [7]. Each of the Architectures described in the literature models the MAS from a different point of view depending on the background of the researchers working in the project, the problem to solve and the type of tools used to develop the system. In our case the MAS has been built following three steps:

- Defining the tasks to perform by the system.
- Grouping together and conceptualising as Agents the elements and tasks with strong relationships.
- Defining a communication mechanism between agents.

To guaranty a successful communication model it was intended to create a simple mechanism that facilitates the elimination and incorporation of agents to the system. Facilitators were used to control the information flow in the multi-agent advisory system because they can facilitate the communication between agents even if they are heterogeneous. Facilitators maintain information about the other agents that form part of the system. This information is basically the address in which to locate them, what type of agents are, what information can receive and send (and how) and the state of the agent (active, inactive or waiting for response).

In the present system agents have been implemented in Java, C and C++ and all of them are installed on a PC, except the information agent, which functions within a UNIX workstation and interacts with an ORACLE database. The use of facilitators has allowed the interoperability of components implemented in different computing languages and installed in different computer architectures. Java applets allows the communication between the components of the systems and the users. Since facilitators and agents use a predefined subset of KQML performatives it is also easy to incorporate new components in the system. If such new component are implemented using a different software or hardware architecture than the architecture of the existing agents, the design of a new facilitator, which is capable of communicating with the already existing facilitators and with the new components, shall be required. Then if new components need to be incorporated into the system it can easily be done.

Student and Lecturer assistants are associated to two facilitators to guaranty a reliable service because these components are spread in several networks with heave information traffic.

4 Conclusion

This architecture facilitates the construction of distributed systems and makes it possible to create successful multi-agent systems in a relatively simple manner. The systems built following this model are also flexible in the sense that they can automatically increase or decrease the number of agents that compose the system. This model of interaction also facilitates the communication and co-operation between heterogeneous components of multi-agent systems. As shown in this paper, facilitators offer the possibility of integrating components with different characteristics (implementation languages and execution environments).

References

1. Jennings N. R., Sycara K., Wooldridge M. (1998). A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent, Systems Volume 1, Issue 1, 1998. pp. 7-38
2. Franklin, S., Graesser A. (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag, 1996.
3. Wooldrige M., Jennings N.R. (1995). Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, vol. 10(2) pp. 115-152, 1995.
4. Shoham Y. (1993). Agent-oriented Programming. Artificial Intelligence, 60 (1): 51-52, 1993.
5. Finin T., Weber J., Wiederhold G., Genesereth M., Fritzson R., McGuire J., Shapiro S., Beck C., (1993) “Specification of the KQML Agent-Communication Language.”, 1993.
6. Aamodt A. and Plaza E. (1994). Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches. AICOM. Vol. 7. No 1. March 1994.
7. Corchado J.M., Lees B., Fyfe C. (1997). “Proyect Monitoring Intelligent Agent System: Communication and Coordination”, Technical Report, University of Paisley, September 1997.
8. Corchado J. M., Lees B., Rees N. (1997) “A Multi-agent System “Test Bed” For Evaluating Autonomous Agents”, Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, California, February 5-8. 1997

9. Kolodner J., (1993). Case-Based Reasoning. Morgan Kaufmann, 1993.
10. C. K. Riesbeck, R. C. Schank, "Inside Case-Based Reasoning", Lawrence Erlbaum Ass. Hillsdale, 1989.
11. KQML, Knowledge, Query and Manipulation Language, http://www.cs.umbc.edu/kqml/