Modeling an Educational Multi-Agent System in MaSE

Izabela Salotti Braga Gago, Vera M.B. Werneck, and Rosa M. Costa

Universidade do Estado do Rio de Janeiro – UERJ-IME
Rua São Francisco Xavier 524, 6o Andar Bloco B – Rio de Janeiro – RJ, Brazil
{vera,rcosta}@ime.uerj.br

Abstract. The growing complexity of modern systems is demanding more and more time from software designers and developers. The challenge of constructing complex systems is complicated by the geographic distribution of the systems. An agent oriented paradigm reinforces the software flexibility and the agent's social possibilities, taking space as a solution in software engineering. The idea of independent and autonomous entities capable of relating in the search of the system goals originated the multi-agent systems, where at least two agents are capable of interaction. Such systems possess great applicability in different knowledge areas, as for example, in Education. This work defines an Education Multi-Agent System, which is a learning education environment with multi-agents. The system aims at helping the teaching process on a specific topic. The system agents were modeled using the MaSE (Multi-Agent System Engineering) method.

1 Introduction

The educational software has been developed since the mid 60s to use Artificial Intelligence (AI) techniques. The software intends to create more flexible programs able to support individualized instructions. The Intelligent Tutoring Systems (ITS) is one of the programs that stand out because it permits the particularized teaching of specific domains by diagnosing the students' structures and knowledge levels and by respecting their learning style and rhythm. In addition, it promotes the shaping of well-structured educational interactions, which are adapted to different kinds of users [3].

The increasing use of multi-agent systems brings challenges that have not been studied yet, such as: how should we adapt elicitation of requirements to cope with agent properties like autonomy, sociability and proactiveness. The agent-oriented modeling is proposed as a suitable software engineering approach for complex organizational application domains that deal with the need for new applications. These requirements are not broadly considered by current paradigms. Autonomy and sociability aspects such as the dependency of an agent on another, and how critical this condition should be, have to be analyzed from the early stages of the software development process [13].

In this work the agent orientation paradigm is adopted in the modeling of an Intelligent Educational Environment, using the MaSE methodology for defining and designing the multi-agent system. We have chosen MaSE because this methodology uses some UML concepts and diagrams that make the process easier to learn and the

J. Liu et al. (Eds.): AMT 2009, LNCS 5820, pp. 335–346, 2009.

support tool free for use. It also has a good performance, leading to a high quality modeling.

This work is organized into 5 sections. Section 2 gives an overview of the MaSE methodology and section 3 defines the purpose of Intelligent Educational Multi-Agent Systems, discussing the system models and agents. Section 4 presents the model requirements and the project design of the system using MaSE. Finally section 5 concludes the work and presents future works.

2 MaSE Methodology

The main purpose of the Multi-agent System Engineering (MaSE) methodology is to support the designer in catching a set of initial requirements and analyzing, drawing, and implementing a multi-agent system (MAS). This methodology is independent of any agent's architecture, programming language, or communication framework. The MaSE treats the agents as a deeper object orientation paradigm, where the agents are object specializations. Instead of simple objects, with methods that can be invoked by other objects, the agents talk among themselves and proactively act in order to reach goals [5], [9].

MaSE is a specialization of traditional software engineering methodologies with two phases (Analysis and Design) and several activities [5]. The MaSE Analysis phase has three steps: Capturing Goals, Applying *Use Cases*, and Refining Roles. The Design phase has four activities: Creating Agent Classes, Constructing Conversations, Assembling Agent Classes, and System Design.

The Analysis phase aims at defining a set of roles that can be used to achieve the goals of the system level. These roles are detailed by a series of tasks, which are described by finite-state models. This process is performed in three steps represented by a Goal Hierarchy: *Use Cases*, *Sequence Diagrams*, *Concurrency Tasks* and *Role Model*.

Goals are what the system is trying to achieve and they generally remain constant throughout the rest of the Analysis and Design phases. Capturing Goals can be divided into two parts: identifying and structuring goals. The Goal representation is a form of hierarchy where the goals are decomposed.

Use Cases describe the behavior of agents for each situation in MAS. In the step *Applying Use Cases,* situations of the initial requirements are acquired and represented into *Use Cases* Diagrams and Descriptions, and UML *Sequence Diagrams*. These representations help the designer show the desired system behavior and its sequences of events.

The third step of modeling is the transformation of MaSE goals into roles. A role is an expected abstract description behavior of each agent that aids in reaching the system goals. In general, the transformation of goals into roles is done in a proportion of one to one: each goal is mapped to a role. However, in some situations it is useful to combine goals into a single role.

The Role Diagram has a number of considerations regarding the representation role and communication between roles. First, each role associated with the same goals is listed below the role name. Often, these goals are represented by numbers used in the Goal Diagram. A set of tasks is associated with each role, representing the expected role behavior.

Once the roles are defined, you must describe the details of each task in the Role Diagram. The definitions of the tasks are shown in *Concurrent Tasks* Diagrams based on finite automata states. By definition, each task must be executed concurrently, while communicating with other internal or external tasks. A *concurrent task* is a set of states and transitions. The states represent the internal agent mechanism, while the transitions define tasks communications. Every transition has an origin and a destination state, a trigger, a guard condition and a transmission [8].

In general, the events that are sent as broadcasts or triggers are associated with events sent to work in the same role instance, requiring an internal coordination of each task. The representation of messages sent between agents uses two special events: send and receive. The send event represents the message sent to another agent and is denoted by send (message, agent), while the receive event is used to define the message received from another agent denoted by receive (message, agent).

The Design phase includes the following diagrams: Agent Classes, Conversations, Agent Architecture and Deployment Diagram. The first step in the design process involves the definition of each agent class, which is documented in an Agent Class Diagram. The system designer maps each role defined in the Role Diagram in at least one agent class. Since the roles are derived from the system goals and are responsible for meeting them, ensuring that each role is mapped in at least one agent class helps guarantee the implementation of the goals in the system. In general the agent classes can be thought of as templates defined in terms of roles they play and of the protocols they use to coordinate with other agents [5], [8].

An agent class has two components: one or more roles and their conversations with other agents. Conversations in an agent class are made between two players, the conversation initiator and the recipient. The conversation detail is expressed through the communication protocols based on *concurrent tasks* that were identified during the analysis in the Role Diagram.

Each task defined in the *Concurrent Task* Diagram describes a component in the agent class. The *concurrent tasks* can be translated into multiple conversations in the Conversation Diagram.

The definition of the agent architecture is performed in two steps: (i) definition of the agent architecture and (ii) components. The designer can choose the agent architecture, such as Belief-Desire-Intention (BDI), Reactive or Knowledge Base. For example, a reactive architecture includes components such as an interface to other agent classes, an internal controller class, a rule verifier, and sensors [4].

The final step in the MaSE design phase uses the Deployment Diagram to show the number, types and location of agents in the system. The diagram describes a system based on agent classes, and is very similar to the UML Deployment Diagram.

The system designer can use the Deployment Diagram to define different configurations of agents and platforms to maximize the processing power and bandwidth of a network.

3 Educational Multi-Agent System Overview

The Educational Multi-Agent System proposed in this work relies on the ITS' classical architecture, considering four models: Pedagogic, Expert, Student and Interface. Each model reflects the ability and the characteristics of the Educational System [12], [13].

The student knowledge should be measured through a questionnaire, and through concerns expressed by the student during the learning process. The student model can follow certain principles such as Differential Model, Overlay Model, Model of Disturbance, Simulation Model, Stereotype Model or Belief Model.

The pedagogical model contains teaching strategies adopted based on the input submitted by the student. Such strategies structure the way in which knowledge is represented and taught to the student, ensuring the success of the methods used by some teaching strategies such as Socrates, Coaching, Hypertext and Others. The tutor can use one or a combination of behaviors, and goals that may vary depending on the information received from the student (actions) and the goal's purpose (plan) [2].

The pedagogical agents may act as virtual tutors, virtual students or colleagues who assist in the virtual learning process. These action forms result on group formations that can distribute tasks between themselves, from greater (student modeling or selection of the strategy and tactics) or lesser (strategy responsibility by each agent) extension.

Dividing the system into smaller tasks reduces the complexity and the monitoring task executed at a higher abstraction level [10]. The learning process characterization developed in some multi-agents systems can be classified in three abstraction levels: (i) replication learning, (ii) learning by tautology or (iii) learning by interaction and shared dynamic [2].

The system is modeled using the paradigm of oriented agents, and each model's basic architecture is represented by an agent. We included other agents in the system to support the key actors. This model uses the definition of learning by replication, where each agent has a specific task, and one of them, the tutor task.

The next subsection describes the agent system models and their characteristics in the system. The whole system uses Artificial Intelligence (AI), Psychology and Pedagogy solutions, involving efficient ways to perform the most varied tasks.

3.1 Student Model

At first, the student model is not represented by an agent in the system architecture. While in the process of modeling it can be represented as an agent, in the proposed student model it is treated as an object that communicates with other elements of the system through messages because we consider this model the representation of the human user in the system. Thus, at certain times of the tutor learning process the student can behave in a reactive, as well as, an active form, indicating concern about a particular point of the subject [4], [12].

The user of the system is represented by the same type of student that looks more like an object, or may have more than one instance and communicates with the other elements of the system through messages.

The stereotype model is used in the modeling of the student, considering the student's initial knowledge in the following classification: beginner, intermediate, advanced and expert. This model type is a simplified model. The strategies of the teaching system are developed and modified according to the level of the student. At the end of each topic, an open question session is opened and a test is applied. After that the tutor starts the process of classifying the user in one of the levels.

3.2 Educational Model

The pedagogical model adopted in this system aims at guiding the student. The tutor sets the teaching strategies, knowledge representation and correction of the tests. At the end it shall be responsible for grading the student to a new level and selecting the next topic to be presented [2].

The tasks to be performed by the tutor in this system are divided in two agents: the Tutor agent and the Coordinator agent.

The tutor agent is responsible for receiving the education strategy and for selecting the module that starts the education process. At the end of the module presentation the tutor opens space for the student to ask questions based on a questionnaire, where the question to be selected by the student is the one that mostly represents his concern. After the question is selected the Tutor agent asks the Expert agent about the answer and the expert agent performs a search in the knowledge base. This cycle can be repeated until the student signals that he understood the question or until the answer possibilities in the knowledge base are exhausted. For the second case, a notification is sent via e-mail to a human teacher registered in the system, and in the future the teacher contacts the user to clarify his question.

The Coordinator Agent selects the strategy in education (or method) according to the initial student level, which will be obtained through a standard questionnaire to be filled out by the student at the beginning of the course. When the teaching strategy is selected the method will also be available to the Tutor agent. The Agent Coordinator is additionally responsible for selecting the next module to be submitted by the Tutor and must define goals for each course, such as teaching the subject until the end, taking questions from students and others. As these goals are met the Tutor agent goes through the teaching process. The definition of the course modules, the content of the next module to be presented, and the teaching strategy is the student's course plan, while the lecture plan is the course plan added to the goals to be achieved on each topic introduced to the student.

3.3 Expert Model

The expert model is responsible for the base knowledge maintenance, providing any necessary information to all actors of the system. The Tutor and Coordinator agents use the knowledge base in the selection of the teaching strategy and in the selection of a response to a query of the pupil, respectively.

The model represents a major expert system intelligent tutor and the rest of the system depends on its knowledge. Therefore, it should be modeled as an expert in order to address the peculiarities inherited to the field you want to express. For example, in a STI in the geometric area, the modules may have different features than other fields like history or nutrition [12].

In this work we do not propose to model the ontology for a specific domain, instead only the basic features of an expert system are modeled.

This expert model adopts the production rule model, which is the most appropriate for the representation of declarative knowledge. This agent is called Expert and has a memory region or blackboard. The region has a mechanism where the requests of

other agents and the system will be checked to match the rules in the knowledge base. In the case of correspondence, it starts the search process of production, which together with its rule is the knowledge. Thus knowledge is sent to the request agent.

3.4 Interface Model

The interface model is represented as streamlined as possible, since it depends on the displayed area, the teaching method style, presentation and used resources. In this work the interface is represented only as a mechanism for input and output of information between the system and user, where the system represents only the type of student.

3.5 Other System Agents

During the system requirement description the need for auxiliary specific tasks was noticed and two agents were detected: the Management agent and the Database Management agent.

The Management agent is active at the beginning of the system as are all the other agents, but it is the first to perform a task. It's first task is to detect the presence of the user in the system by monitoring the system. The agent will invite the user to register, which will have questions concerning the system domain area. This determines the student's initial level, which will be recorded in the database. Consequentially, the Management agent must create the student registration. The initial student level is recorded by the Coordinator agent that will select an appropriate strategy for the student education.

The Database Management agent is responsible for both meeting the other system agent requirements of databases, for maintaining the information validation and formatting the correct information required by other agents.

4 Educational Multi-Agent System MaSE Modeling

The Educational Multi-Agent System was named Educ-MAS and was developed based on MaSE methodology and on the issues discussed and defined in the previous section. The tool used to draw the diagrams in MaSE was AgentTool [1] that guides the methodology application [5], [7], [8].

In the Analysis first step, the main goal and its sub-goals were identified from the initial set of the system. They were then structured in a goal tree as shown in Figure 1.

The goal *Define Tuition Plan* was structured based on the four models described in section 2. Our examination of the importance and inter-relationships of the goals was also taken into consideration. The *Promoting individual learning* is partitioned into four goals: *Explore Student, Plan Tuition, Manage Knowledge* and *Manage Tuition*. The goals are also decomposed into other goals. For example the goal *Plan Tuition* was partitioned into two sub-goals (*Consult Defined Goals* and *Define Tuition Plan*) and the sub-goal *Define Tuition Plan* has two sub-goals named *Defining Content of the Module* and *Defining Plan Presentation of the Modules*.

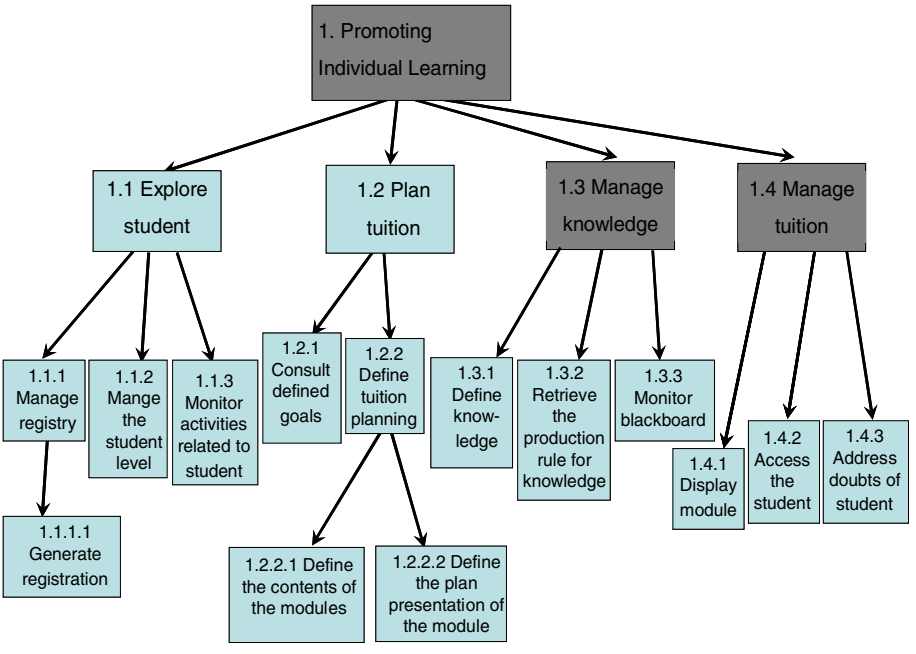

Fig. 1. Educ-MAS Goals

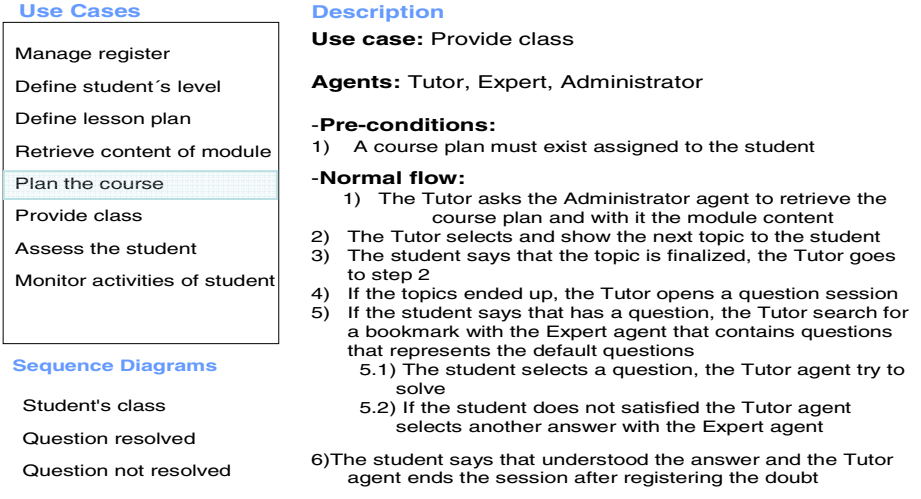

Fig. 2. Use Case Provide Class

After defining the goals, eight primary *use cases* were generated based on the goals *Explore Student* and *Plan Tuition*. The Figure 2 shows the *use case Provide Class*, its description and also the name of *Sequence Diagrams* that retracts the scenarios of this

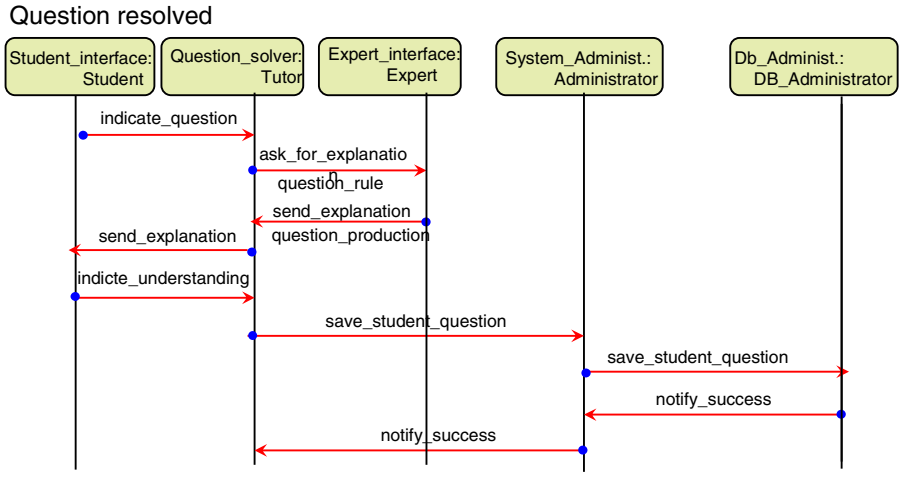

Fig. 3. Question Resolved *Sequence Diagram*

use case. The scenarios are Student's Class, Questions Resolved and Questions Not Resolved. For each one we built a Sequence Diagram that shows how the system behaves. Figure 3 demonstrates the agent behavior with the second scenario of the case *Provide Class*. We defined for each *use cases* the description and their scenario *Sequence Diagram*. The whole specification of Educ-MAS can be found in [11].

We developed a set of roles and tasks to show how the goals are reached based on the goals, the *use cases* (diagrams and descriptions) and the *Sequence Diagrams*. The Figure 4 represents the Preliminary Role Diagram where the goals were mapped to system roles. For example, the *Course Administrator* role achieves the goals *Plan Tuition, Consult Defined Goals, Define Tuition Plan*, *Defining Content of the Module* and *Defining Plan Presentation of the Modules*. In the complete Role Diagram we introduced the tasks responsible for the roles and the associations among themselves to reach the responsible goal's roles.

Once the Role Diagram was developed, we defined a *Concurrent Task* Diagram for each task as presented in Figure 5 for the task Monitor Blackboard. This task is associated to the Expert interface role, which is responsible for the 1.3.3 goal that monitors the blackboard in order to interface the knowledge base introducing the contents and answering questions.

The Analysis Models were built showing the behavior of the system by deriving and establishing the goals that were used to create the use cases and the sequence Diagrams. Those goals were mapped into roles and tasks defined in the Role Diagram and detailed in the Concurrence Task diagram. In the Design phase we use those diagrams and we constructed the individual components of the agent classes as presented in Figure 6. We chose a simple agent architecture and an example of a Tutor agent class partial structure component as shown in Figure 7, where the attributes and methods were derived from the *Concurrence Task* diagram.

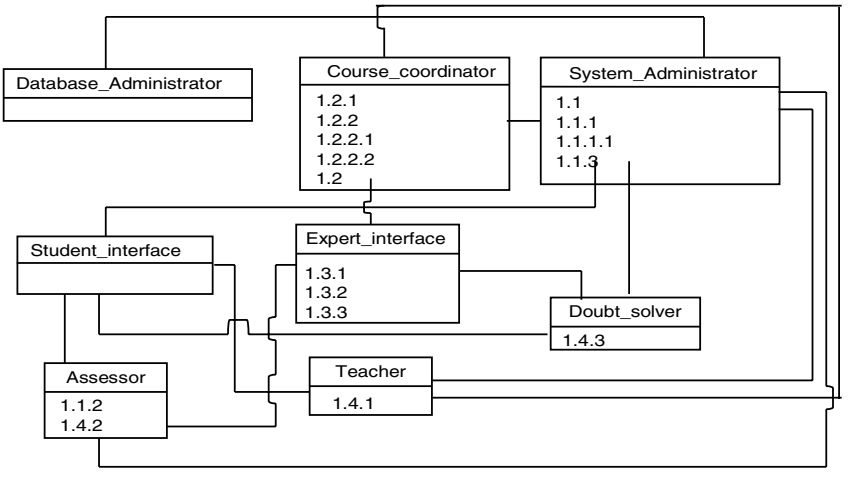

Fig. 4. The Educ-MAS Partial Role Diagram

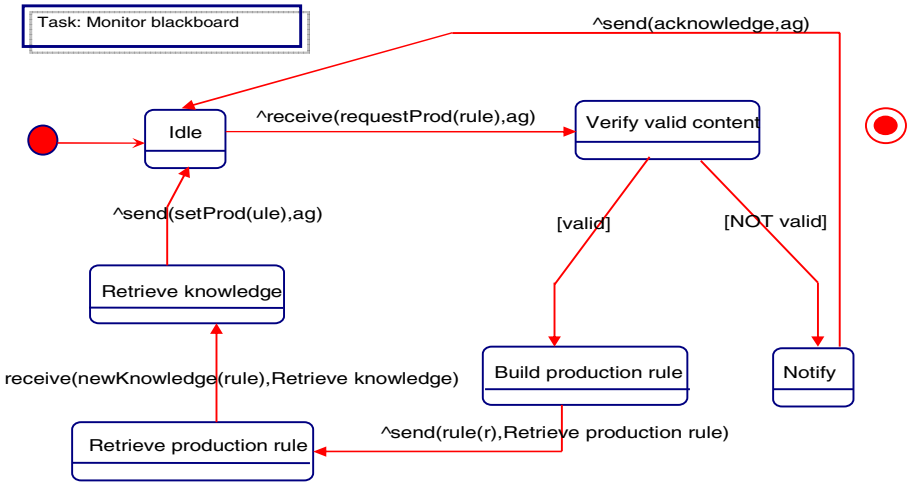

Fig. 5. *Concurrent Task* Diagram for the task Monitor blackboard

The last step in the Design phase was to develop an overall operational design. The Educ-MAS is presented in Figure 8. The Tutor, Administrator, Coordinator and Expert Agents are defined in an environment as a system. The Interface (Student Model) starts at the student's computer where the Database Management and the other part of the system are in network computers.

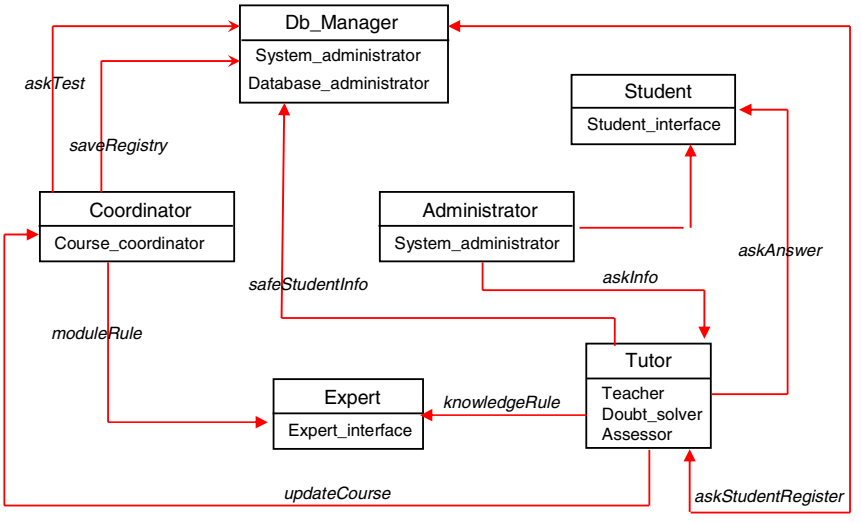

Fig. 6. The Educ-MAS Agent Class Diagram

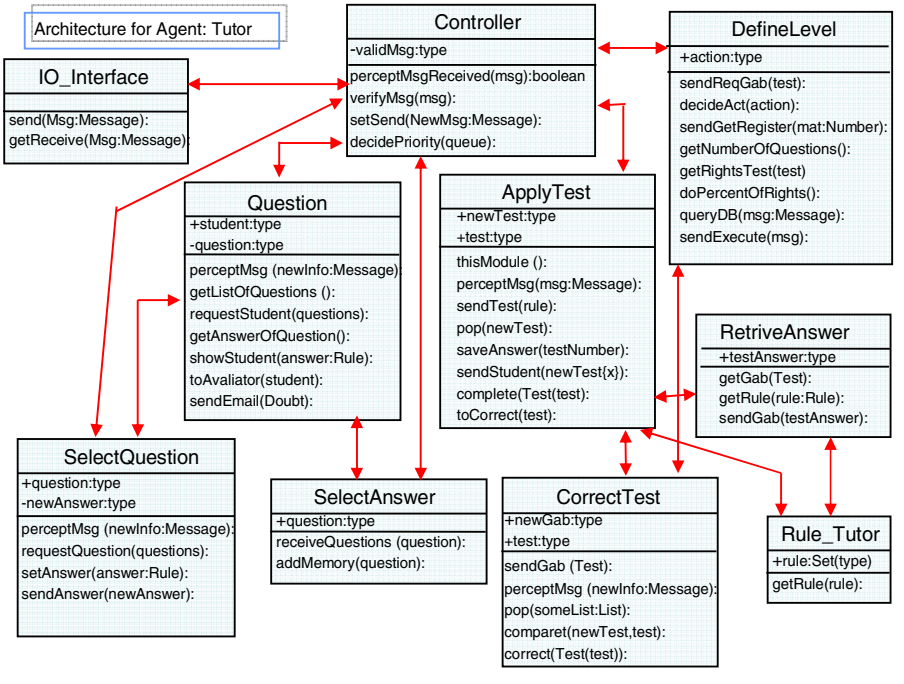

Fig. 7. Tutor Agent Class Partial Structure Component

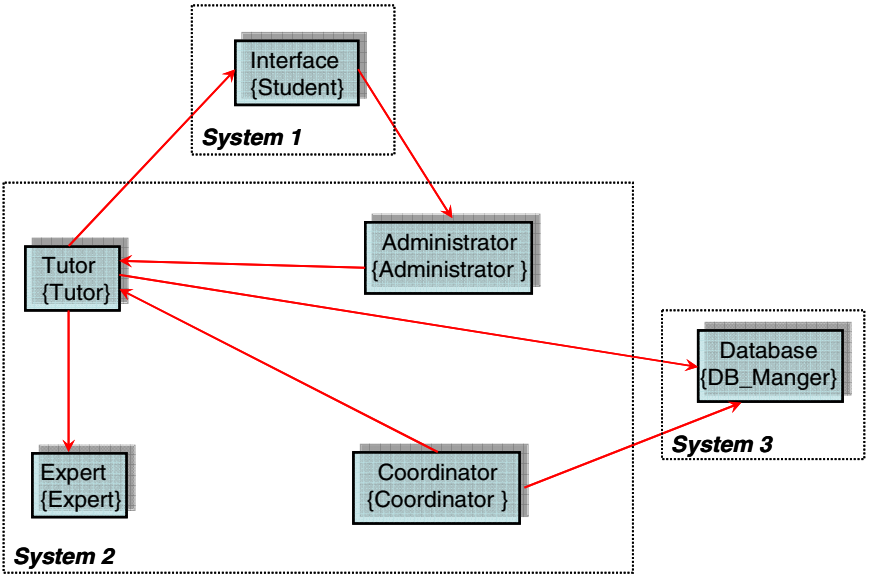

Fig. 8. The Educ-MAS Deployment Diagram

5 Conclusion

During the development of this work several concepts related to the paradigm of the oriented agents were explored. The MAS has a wide application both in industry and in science. Since the MAS is an area of artificial intelligence, agents can be considered capable to make decisions, to learn and thus to be able to update their knowledge base. Therefore, the ability of MAS as a system to optimally solve problems to achieve the system goals becomes an attraction to the pedagogy area.

Although the development of artificial intelligence applied to MAS has evolved over the years, the intelligent tutoring systems still have certain limitations when compared to human professors and teachers. The reason for why the student does not learn certain topics, for example, is still not fully resolved in intelligent tutoring systems. This is an important feature to develop further knowledge especially on the student system.

Currently, the development of intelligent tutors for distance education has gained importance, and has been implemented to be accessible over the Internet, enabling teaching at any time or place.

The ITS modeled in this work was developed considering only the common features of ITS, disregarding the knowledge tutor field. However the MaSE modeling in its original definition does not mention any particular method of knowledge base modeling. The developers of the methodology understood this need and changed the life cycle of modeling MASE in order to include these types of systems. For the STI knowledge base construction we will use an ontology that contributed with the organization of the expert knowledge that seemed to be a tool to clarify the expert reasoning. The ontology represents the world in terms of an explicit conceptualization

that can be understood by different people and can be used for different approaches for computer systems development.

The MASE methodology introduced a further step in the process of analysis, specifying that the definition of ontology should happen within the modeling of the system [6]. This step will be applied during this phase when we build two specific EducMAS: the Geometry learning for high school and the Nutrition Bio-availability in a Nutrition Graduating program.

References

1. Agent Tool, http://macr.cis.ksu.edu/projects/agentTool
2. Bercht, M., Vicari, R.M.: Pedagogical agents with affective and cognitive dimensions. In: Sanchez, J. (ed.) Proceedings of the V Congreso Iberoamericano de Informática Educativa, 2000, Vinãdel Mar. RIBIE 2000. V Congreso Iberoamericano de Informática Educativa, RIBIE 2000, CD-ROM. Universidade do Chile (2000)
3. Bordini, R.H., Lesser, V., Viccari, R.M., Bazzan, A.L., Janone, R.O., Basso, D.M.: AgentSpeak(XL): efficient intention selection in BDI agents via decision-theoretic task scheduling. In: Castelfranchi, C., Johnson, W.L. (eds.) Proceedings of the AAMAS 2002—autonomous agents and multi-agents systems 2002, pp. 1294–1302. ACM Press, Bologna (1999)
4. Bryson, J., Stein, L.: Modularity and design in reactive intelligence. In: Int. Joint Conf. on Artificial Intelligence IJCAI 2001, Seattle (USA), pp. 1115–1120 (2001)
5. Deloach, S.A.: Analysis and Design using MaSE and agentTool. In: 2th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001), Miami University, Oxford (2001)
6. Dileo, J., Jacobs, T., Deloach, S.: Integrating Ontologies into Multiagent Systems Engineering. In: Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems AOIS 2002 (2002)
7. Goulart, R.R.V., Giraffa, L.M.M.: Arquiteturas de Sistemas Tutores Inteligentes. In: TECHNICAL REPORT SERIES – Faculdade de Informática – PUCRS – Brasil, http://www.pucrs.br/inf/pos/mestdout/rel_tec/tr011.pdf
8. wHenderson-Sellers, B., Giorgini, P. (eds.): Chapter XI – Multi-Agent Systems Engineering: An Overview and Case Study. In: Agent Oriented Methodologies. 1ed: Igi Global, pp. 317–340 (2005)
9. O'Malley, S.A., DeLoach, S.A.: Determining when to use an agent-oriented software engineering paradigm. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, p. 188. Springer, Heidelberg (2002)
10. Russell, S., Norving, P.: Intelligent Agents. In: Artificial Intelligence: A Modern Approach, 3rd edn. (2002), http://www.cs.berkeley.edu/~russell/aima1e/chapter02.pdf
11. suppressed
12. Vicaria, R.M., Floresa, C.D., Silvestrea, A.M., Seixasb, L.J., Ladeirac, M., Coelho, H.: A multi-agent intelligent environment for medical knowledge. Artificial Intelligence in Medicine 27, 335–366 (2003)
13. Wooldridge, M., Jennings, N.: Intelligent Agents: Theory and Practice (1997), http://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html.html