Towards an Ontology-based Framework for Building Multiagent Intelligent Tutoring Systems

Ig Bittencourt[2], Evandro Costa[1], Hyggo Almeida[2], Baldoino Fonseca[1], Guilherme Maia[1], Ivo Calado[1] and Alan Silva[2]

[1]Computation Institute – Federal University of Alagoas
Campus A. C. Simões, BR 104 - Norte, Km 97, C. Universitária, Maceió, AL – Brasil
{evandro,bfsn,jgmm,iaarc,alan}@tci.ufal.br

[2]Federal University of Campina Grande
Rua Aprigio Veloso, 882 - Bodocongo, 58.109-900 – Campina Grande – PB – Brasil
ibert@dsc.ufcg.edu.br,hyggo@dee.ufcg.edu.br

Abstract. *Intelligent Tutoring Systems (ITS) are inherently complex, domain-oriented software systems which are frequently pointed out by researchers as suitable applications for the multi-agent approach. Developing and maintaining Multi-agent ITS are a hard task since it involves different stakeholders, with different expert and roles, such as developers, for developing new software features; domain experts, for managing ITS knowledge domain; authors, for customizing ITS execution for a given context; and users, which are not aware about ITS complexity and require a friendly user interface to interact with the system. Some works have been proposed to support the development of ITS, but they do not consider the stakeholders involved in the whole development and maintenance processes. In this paper we present a framework for designing, developing, and maintenance of Multi-agent ITS. This framework aims to be useful to ITS developers, domain experts, authors and users, providing a different view for each stakeholder, with different tools to support their activities. Indeed, it is introduced the first steps towards the framework architecture, design, and extension points, detailing how to customize them for specific domains focusing mainly on developers. Finally, to illustrate our proposal approach a case study is presented.*

1. Introduction

Software engineering continually searches for effective approaches to manage the complexity that is inherent in most software systems. Intelligent Tutoring Systems (ITS) are a kind of complex, domain-oriented software systems which are frequently pointed out by researchers as suitable applications for the multi-agent approach.

Developing and maintenance of ITS applications are hard tasks, proving to be complex and often requires a high cost of production and maintenance [Aleven et al. 2006]. It includes different stakeholders, with different expert and roles, such as developers, for developing new software features; domain experts, for managing ITS knowledge domain; authors, for customizing ITS execution for a given context; and users, which are not aware about ITS complexity and require a friendly user interface to interact with the system.

Some works have been proposed to support the development of ITS, but they do not consider the stakeholders involved in the whole development and maintenance process. To address these concerns, this paper introduces the first steps towards the framework for designing, developing, and maintenance of Multi-agent ITS.

The proposed framework aims to be useful to ITS developers, domain experts, authors and users, providing a different view for each stakeholder, with different tools to support their activities. Particularly, it provides to developers an approach to guide the development of ITS according to the multi-agent architecture derived from Mathema model [Costa et al. 1998]. This model offers an agent-based ITS designed for providing cooperative interactions between human and artificial agents, primarily motivated by problem solving situations. Its main goal is to increase the opportunities for students to construct their own knowledge through a problem-based learning approach.

Additionally, in order to building ITSs applications, the developers have to use ontologies to configure all the extension points (such as dimensional view of the mathema and agents), detailing how to customize them for specific domains. Furthermore, a case study is presented to describe the extension points of the framework.

The remainder of this paper is organized as follows. The research context is discussed in Section 2. The proposed framework is described in Section 3. A case study by using this framework is discussed in Section 4. Finally, conclusions are presented in Section 5.

2. Research Context

This section aims to primarily describe some important characteristics of Mathema model which has been considered useful to clarify this work. The Mathema model was used as a conceptual basis for the proposed paper, because it approaches a model for multi-agent-based intelligent tutoring systems. Moreover, the implementation architecture for the proposed framework is presented.

2.1. Multi-layer Architecture

The architecture showed is more concerned with implementation aspects and roles presented in ITSs. These roles can be divided into two types. 1) the roles concerning the ITS's building process and 2) the roles concerning the usage of a generated ITS.

The roles regarding the conception and development are: i) developers/programmers: they are responsible for developing and adding new functionalities to the framework layer; ii) Authors/Non-programmers: they are responsible for configuring the system by defining the learning objects, specify the models (domain, student, and pedagogical), and others. In addition, author as knowledge engineers is presented in this layer, being responsible for configuring the knowledge based mechanisms, such as case-based reasoning and rule-based reasoning; iii) Users: they are responsible for providing/defining the requirements of the intelligent tutoring system.

In addition, the roles concerning ITS's usage are: i) Students: they learn through the interaction with pedagogical researches and with others (human and/or artificial) agents; ii) Teachers: they collaborate/give support to students in the learning process; ii) Artificial Agents: they are computational agents that interact with students by providing cooperative support during the problem-solving process. Figure 1 shows a multi-layer architecture for building agile intelligent tutoring systems.

The architecture was developed as a multi-layer architecture and it has the following layers:

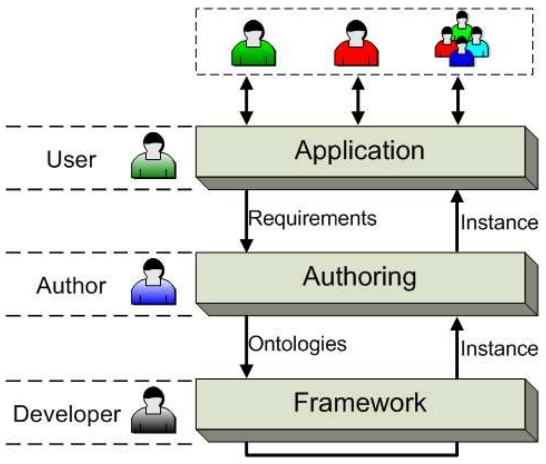

Figura 1. The Multi-layer Architecture.

- Framework: it is maintained by developers who can add new functionalities. The inputs of this layer are three ontologies: 1) Mathema Ontology: it represents the educational specification, defining the pedagogical, student, and domain models; 2) Inference Ontology: it represents the ontology used by knowledge engineers to configure inference mechanisms and 3) Interaction Ontology: this ontology is responsible for the interaction between the agents. The output of this layer is an instance of the framework;
- Authoring: this layer is responsible for providing authors with a user-friendly interface which is used in the ontologies specification. The input of this layer are the requirements of the desired ITS application and the output represents ontologies populated with individuals according to these requirements;
- Application: this layer represents the user application and is used to: i) define the requirements of the desired ITS, where these requirements regard fundamental information for personalized tutoring systems and ii) final users as students, teachers, and others.

The focus of the paper is the framework layers with its input and output aspects. The next sections show details about this layer.

3. The proposed Framework

We have developed an ontology-based framework, called *ForBILE*, to facilitate the development of multiagent intelligent tutoring systems. The goals of this framework are three. First, assure the low time cost for building intelligent tutoring systems, with a minimal amount of code modification. Second, provide an adaptive application according to the necessities of the user. Third, evolve the autonomous tutoring agent's knowledge and inference capabilities. The technologies used in the development of the framework were *Tomcat*, *Jade*, *Protégé* and *OWL-DL*. Figure 2 shows the ontology-based framework for multiagent building intelligent tutoring systems.

These agents were developed using the Jade Framework which provides mechanisms for agent interation and message exchange. In addition, Jade implements the interoperability standards for agent communication (FIPA). However, in order to agents interact and provide students with personalized tutoring systems, some specifications have to be described. Three ontologies were developed in order to assure the agile ITS's development. The next subsection discusses the developed ontologies.

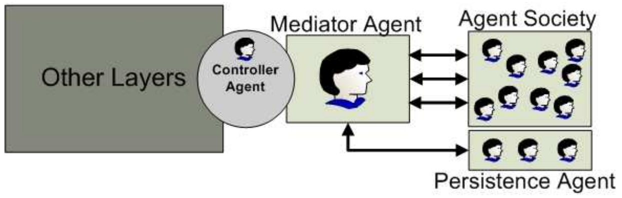

Figura 2. Agent-based Learning System.

3.1. Ontologies

The ontologies were used to: i) assure the interaction among the agents, ii) specify the domain, student, and pedagogical models and iii) configure inference mechanisms.

3.1.1. Interaction Ontology

A communication protocol was defined to the agents in the framework. This protocol was specified through the construction of an ontology using Protégé. This ontology is defined by a triple, which are: Agent(basic information about the agents), Service(services provided by each agent in the framework) and Ability(abilities presented in the pair $< Agent, Service >$). Each agent in the framework is an individual in the ontology. The specified ontology is described in Figure 3.

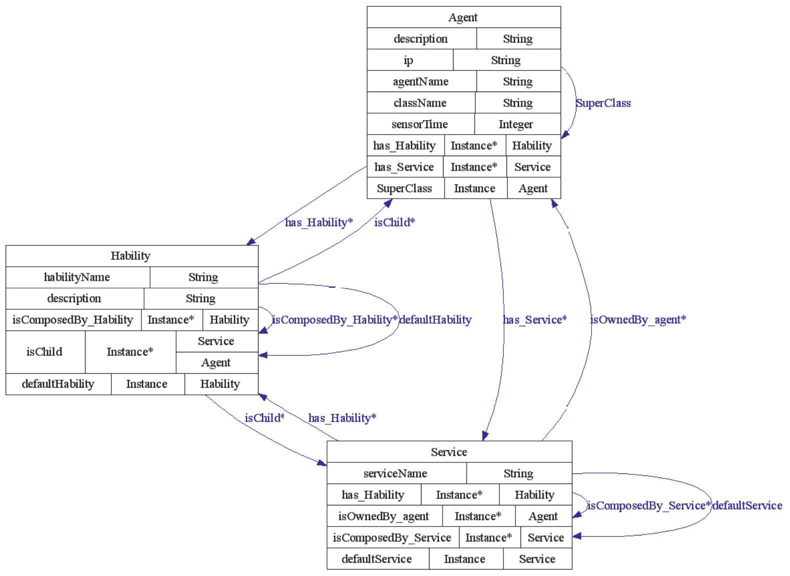
Agent
description | String
ip | String
agentName | String
className | String
sensorTime | Integer
has_Hability | Instance* | Hability
has_Service | Instance* | Service
SuperClass | Instance | Agent
SuperClass
has_Hability*
isChild*
Hability
habilityName | String
description | String
isComposedBy_Hability | Instance* | Hability
isChild | Instance* | Service | Agent
defaultHability | Instance | Hability
isComposedBy_Hability*defaultHability
has_Service*
isOwnedBy_agent*
isChild*
has_Hability*
Service
serviceName | String
has_Hability | Instance* | Hability
isOwnedBy_agent | Instance* | Agent
isComposedBy_Service | Instance* | Service
defaultService | Instance | Service
isComposedBy_Service*defaultService

Figura 3. Interaction Ontology.

Indeed, this ontology has information about the implementation, like the name of the packages and description of each service/ability. Due to this aspect, if any service/ability has more than one implementation, a default implementation is defined in the ontology. In other words, this ontology allows inversion of control[1] in the framework.

[1]Inversion of Control is one of the properties presents in a framework [Fontoura et al. 2001].

3.1.2. Mathema Ontology

The ontology was developed through the integration with other researches, contributing as with ontologies as with theoretical approaches [Bittencourt et al. 2006a]. These contributions are cited along the following subsections.

Domain Model The domain model is responsible for the knowledge about *what will be taught*. The researches evaluated to build this model were [P. Dillenbourg 1992, Chen and Mizoguchi 2004, Costa et al. 1998]. The Figure 4 shows the structure of the ontology based on the three-dimensional view of the domain according to Mathema Model.

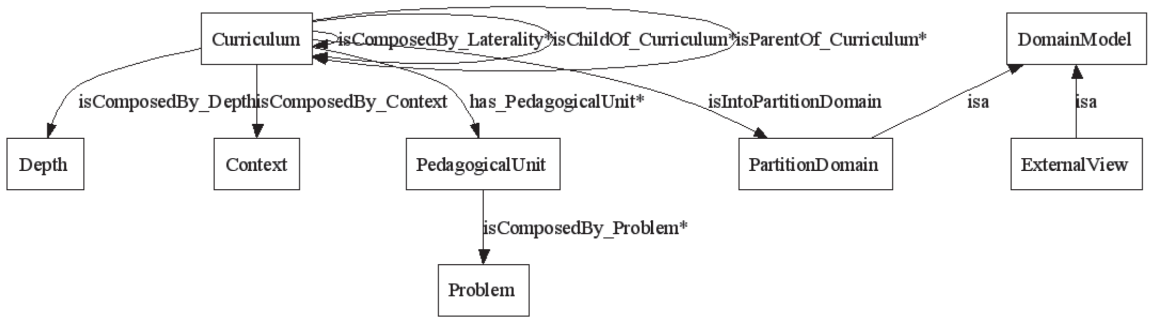

Figura 4. Three-dimensional view of the Mathema.

Student Model The construction of the model was developed through the evaluation of [Chen and Mizoguchi 2004, Chepegin].

The Student Model has the knowledge about *who will be taught*, that is, this model contains information about the student being taught. The types of information necessary to this model are: i) *Static Information*: the student information that do not change during the student-system interaction (see Figure 5); ii) *Dynamic Information*: the student information that change during the student-system interaction. Usually, this information is associated with the domain information, like student cognitive diagnosis. Figure 6 presents interaction features between the student and the system.

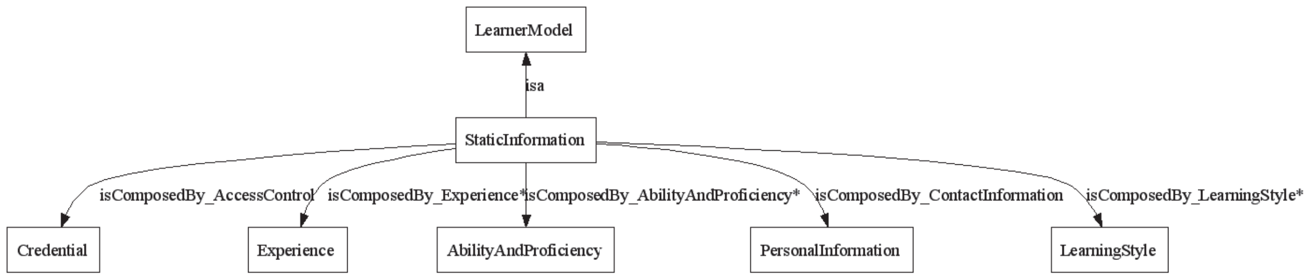

Figura 5. Student Static Information.

Pedagogical Model It has knowledge about *how to teach*, that is, how the interaction will be conducted. Usually, this interaction occurs through an instructional plan that takes into account cognitive aspects of the students. The pedagogical model construction was

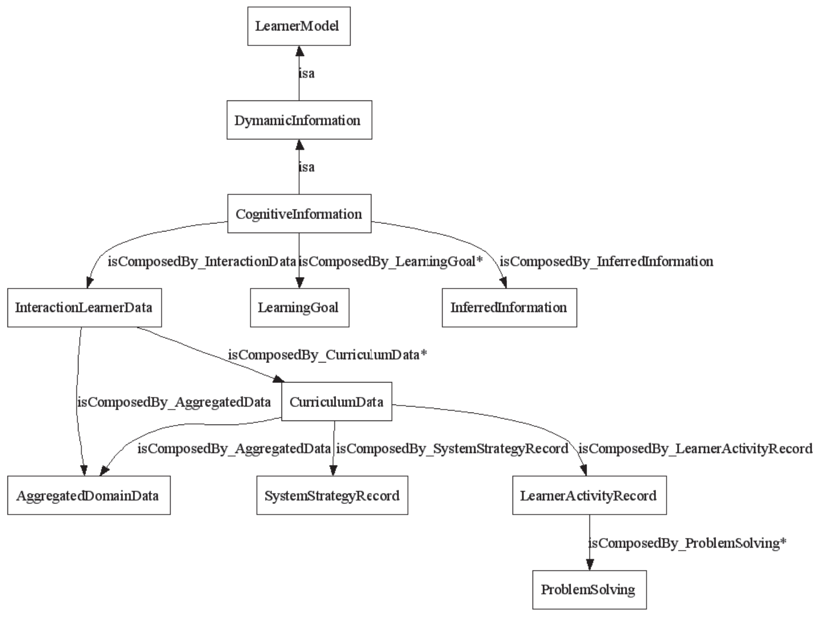
LearnerModel
isa
DymamicInformation
isa
CognitiveInformation
isComposedBy_InteractionData
isComposedBy_LearningGoal*
isComposedBy_InferredInformation
InteractionLearnerData
LearningGoal
InferredInformation
isComposedBy_CurriculumData*
isComposedBy_AggregatedData
CurriculumData
isComposedBy_AggregatedData
isComposedBy_SystemStrategyRecord
isComposedBy_LearnerActivityRecord
AggregatedDomainData
SystemStrategyRecord
LearnerActivityRecord
isComposedBy_ProblemSolving*
ProblemSolving

Figura 6. Dynamic information regard students.

based on the works [du Boulay and Luckin 2001, Kumar et al. 2004, Major et al. 1997]. Moreover, the instructional plan (as shown in Figure 7) makes use of pedagogical strategies and tactics that correspond to the way a student or a group of students are taught.

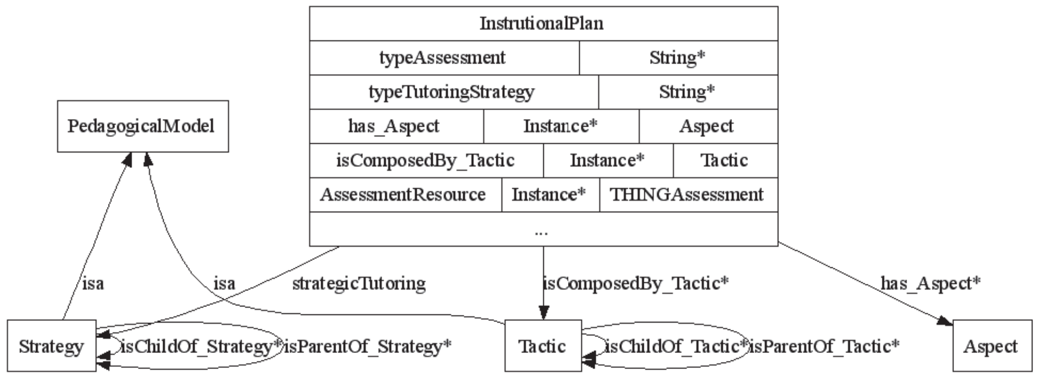

Figura 7. Pedagogical Model.

3.1.3. Inference Ontology

The use of an inference mechanism occurs, first, through the specification of the inference ontology. This ontology allows integration of inference mechanisms, dynamically. In order to assure the integration, four type of information have to be considered in the specification of the inference algorithm (see Figure 8), which are: *Input/Output*(it represents the input and output data and their types), *Reasoning*, *Feedback*, and Statistics(pre-established data used to evaluate the efficiency of the algorithm).

3.2. Agents

The agents assure the adaptive way at the learning process. They are composed by Controller Agent, Mediator Agent, Persistence Agent, and an Agent Society, as shown in

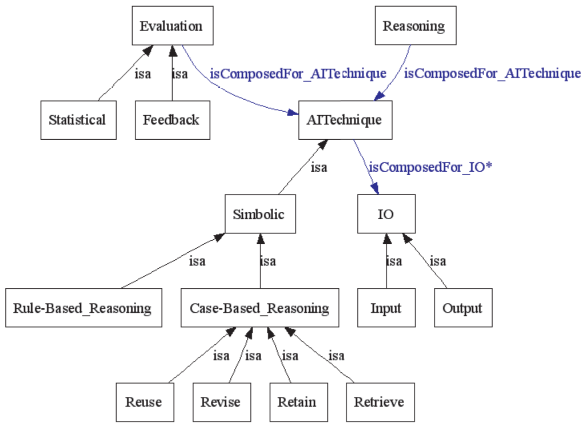

Figura 8. Inference Ontology

Figure 9.

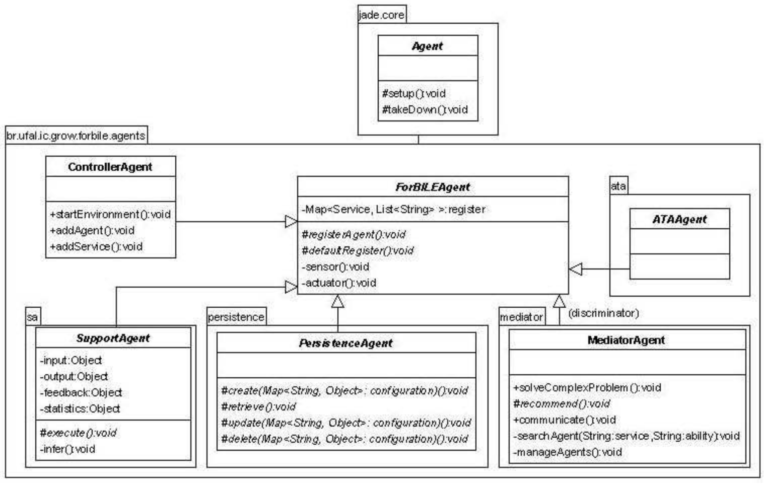

Figura 9. Class Diagram and package of the Kernel.

The extensibility of Jade occurs by the Agent Class, where ForBILEAgent extend it. ForBILE Agent is an abstract class and implements some default functionalities, like the register of services, sensors and actuators. The sensor is responsible for perceiving the environment and the actuator is responsible for acting in the environment.

Aiming to discuss functionalities of the agents presented in the framework, the next subsections specify each agent.

3.2.1. Controller Agent

The Controller Agent (CA) has three fundamental skills, which are: i) Start Agents: to build all the agents when the system is started ii) Add, remove, and update agents of the society; iii) Add, remove, and update the pair $< Service, Ability >$ of the agents: each agent can change their services and abilities dynamically.

3.2.2. Mediator Agent

The complexity involved in the interaction management of the agent society motivated the use of a mediator agent (MA) to coordinate as best as possible the interaction process. The usage of each functionality is described below:

1. In order to assure the construction of the Agent Society, management functionality was added. This behavior configure dynamically the interaction ontology ($< Service, Ability >$) for each Autonomous Tutoring Agent (ATA). In other words, when the CA creates the MA, the MA configures the ontology and send the list of ATA (Cognitive Agents) to be created;
2. Some of the recommendation (service) ways are: i) when ATA needs to interact between them; ii) when the student wish interact with other student; iii) when the student needs help of an expert in the domain. In order to guaranty this functionality, the developer has to follow two steps: i) implements the *recommend* method (*MediatorAgent* class) and ii) configure the protocol by defining the specific recommendation ability;
3. The complex problem solving process occurs due to the capacity of all the agents solve their tasks. It is invoked when an ATA agent requires cooperation of others ATA agents to solve a problem. The implementation of this functionality is provided by the framework.

With the functionalities cited above, it is demonstrated the reusability and extensibility in order to overcame the agent interaction, recommendation, and the complex problem solving.

3.3. Agent Society

The complexity regarding the adaptive teaching process motivated the use of educational and intelligent agents. For this, a heterogeneous agent (composed by autonomous tutoring agents and support agents) society was built in order to make this process as effective as possible, as follows below.

3.3.1. Autonomous Tutoring Agents

The Autonomous Tutoring Agents (ATAs) were modeled based in the Mathema Model [Costa et al. 1998], through the development of a top ontology (described in Subsection 3.1.2).

The ATAs are responsible for the teaching process. In order to build an ATA agent, two steps are necessary: First, specify the models (student, domain, and pedagogical) configuring an ontology [Bittencourt et al. 2006a]. Second, define which type of ATA agent is intended to be built. The types of ATA agent are: i) cognitive: it is always presented in instructional system and the developer has to implement functionalities like assessment and diagnostic. For this, the developer has to extend the *CognitiveAgent* class and implement the defined abstract methods. ii) others: these agents depends on the specific aspects of the application. These agents could be motivational, affective, metacognitive, etc. In order to provide other types of agents, the developer has to extend the *ATAAgent* class and implements the intended methods. Figure 10 shows an example of this extensibility.

With the functionalities cited above, the complexity in the implementation of tutoring agents is reduced through the ATAAgent class extension.

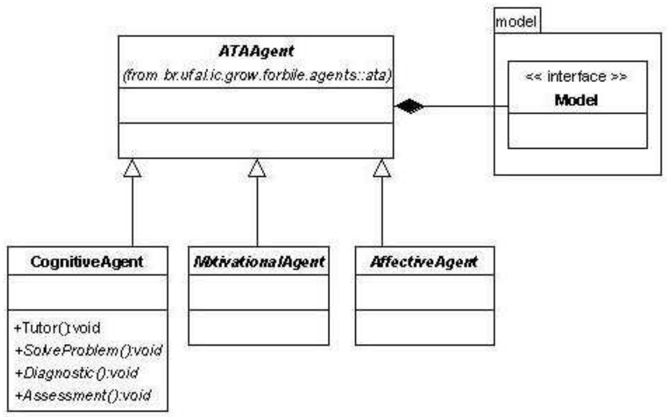

Figura 10. Class Diagram of ATAAgent.

3.3.2. Support Agents

Support agents have features used to infer in accordance with a prior constructed mechanism. In order to use support agents, the developer has to implement a component (inference mechanism), extend the SupportAgent class (Figure 11) and implements the *execute* method to use the component.

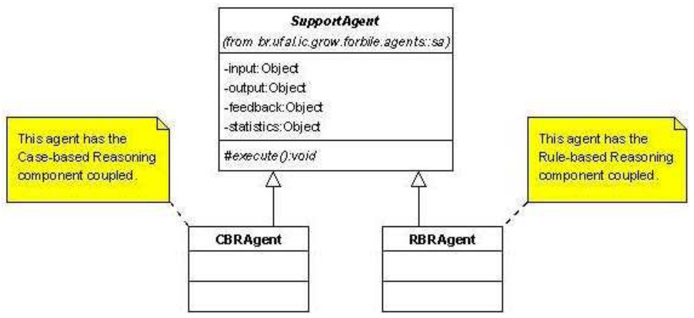
SupportAgent
(from br.ufal.ic.grow.forbile.agents::sa)
-input:Object
-output:Object
-feedback:Object
-statistics:Object
#execute():void
This agent has the Case-based Reasoning component coupled.
This agent has the Rule-based Reasoning component coupled.
CBRAgent
RBRAgent

Figura 11. Class Diagram of the support agents.

These intelligent agents improve the effectiveness of the adaptive teaching process. Furthermore, in the attempt to make easy the development of intelligent tutoring systems, two inference mechanisms were implemented and released with the system, which are Case-Based Reasoning (CBR) and Rule based Reasoning (RBR) [Bittencourt et al. 2006b].

3.4. Related Work

Many tools for building instructional systems have been created. A relevant analysis of the state of the art can be viewed in [Murray 2003]. However, recently, some new environments have been developed. One of them, considering the proposals related to the presented proposal are described below.

[Aleven et al. 2006] presents CTAT (Cognitive Tutor Authoring Tools). It has two types of tutors (Cognitive Tutors and Example-Tracing Tutors), where they represent different trade-offs between ease of authoring on the one hand and generality and flexibility of the resulting tutors on the other. However, the process for building interface agent is too slow. In addition, authoring facilities are not so intuitive because i) to build Cognitive

Tutors the knowledge engineer is required and ii) to build Example-Tracing Tutors, the author has to know graphs notion.

[de Almeida et al. 2004] presents a Framework for building virtual communities, providing several interactive tools, such as blog, forum, e-mail, rss, digital library, and others. However, this framework does not support intelligent agents.

4. Case Study

This section presents a case study conducted in order to evaluate the proposed framework. Our framework was used in the development of a legal ITS, called Themis [Bittencourt et al. 2006d]. This ITS provides Law students with real cases, rules and different point of views with a given body of knowledge. The main idea is to engage Law students into interactions with the system based on the resolution of Legal problems and their consequences on other tutorial activities, concerning the Penal Law. The interaction happens in two ways: i) when the system sends subject content and a problem to be answered by the student and ii) when the student sends a problem to be solved by the system.

An important aspect of Legal domain is the problem specification, because it takes into account learning resources, like doctrine, Jurisprudence or Legislation[2]. A problem is defined by a 3-Tupla $\langle P, I, F \rangle$, where: i) *P*: it represents a real penal situation; ii) *I*: it represents an interpretation set of the problem *P*. The interpretations are based on two views: Lawyer View and Prosecutor View; iii) *F: P x I*: it represents a theoretical recital of the relation *P x I*, and it can be a doctrine, Jurisprudence or Legislation.

In addition, Case-based and rule-based reasoning are used as problem solving mechanisms. These mechanisms are motivated by the "legal structure" which is based on the legislation and jurisprudence.

The system has five agents, MediatorAgent, CognitiveAgent, CBRAgent, RBRAgent, and PersistenceOWLEMathemaAgent.

The steps to be followed by the developer are: i) Configure the ontology communication protocol in order to assure the interaction between the agents; ii) Extend the CognitiveAgent class and implement the ProblemSolving method, according to the specification of legal problems;

Moreover, in the problem process it is necessary the interaction between CBRAgent, RBRAgent and CognitiveAgent in order to solve the problem. In addition, this interaction is overcamed by the ontology communication protocol of the mediator agent. The Figure 12 shows the interaction between the agents.

4.1. Evaluation

The proposed framework was implemented and validated in two real scenarios. The system is being used by the Federal University of Alagoas (UFAL) and Catholic University of Brasilia (UCB). An application in medicine domain have been used at UFAL and UCB, and another in legal domain have been used at UFAL.

The main improvements identified with the use of the system were the solution of some problems like high development cost, complexity to develop AI algorithms, AI techniques integration, scalability, difficulty for share materials, and others.

However, the main difficulties identified were: i) Ontology version: as the universities (UFAL and UCB) are at different places, each one has its own ontology. The solu-

[2]This information were structured in an ontology, however it is not the focus of the paper.

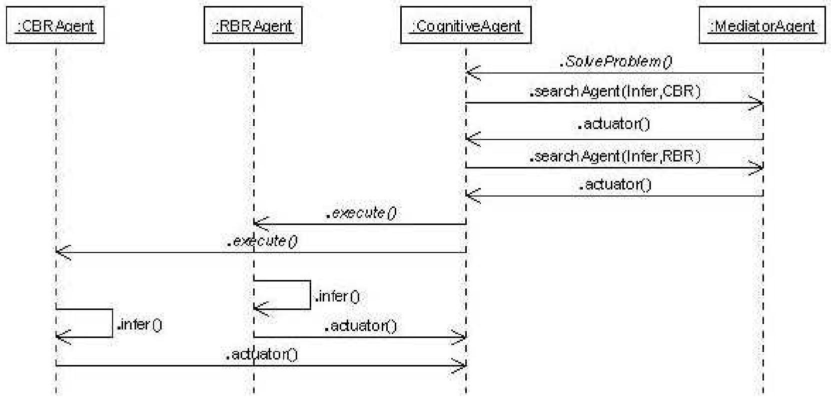

Figura 12. Sequence Diagram approaching the solving problem process

tion found was the use of a Protégé plug-in called PromptTab which was used to compare the ontologies and fixe then; ii) Slowness: The use of ontologies by the agents made the persistence process really slow. The better solution was the use of a computer with more processing power; iii) Ontology exchange through Jade messages: a serious problem was the exchange of ontology objects through Jade messages. The ProtegeOWL-API have been used to generate java classes. Although the objects are serialized, they became a null reference when it arrived to its destination place. So, a region to message exchange was developed and it is controlled by a semaphore algorithm.

5. Conclusion

This paper described the first steps towards a functional design of an ontology-based framework aiming to give support to developers to rapidly build multiagent ITS for particular domains. This framework has been successfully applied on the constructing of ITS in two heterogeneous domains: Legal [Bittencourt et al. 2006d] and Medicine[Bittencourt et al. 2006c]. It has been used java, jade and Protégé technologies. The main contribution of the proposed paper is to make the easier and more efficient the way to develop intelligent tutoring systems.

At the time of this paper we were working in an improvement of the two mentioned applications by including other services and updating some mechanisms, as for example: machine learning techniques. One of them has been designed to improve the selection of pedagogical actions by using reinforcement learning. The other is concerned student model by using neural network. Furthermore, we are planning another application oriented to a formal domain, probably will be mathematics.

Referências

Aleven, V., McLaren, B. M., Sewall, J., and Koedinger, K. R. (2006). The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In *Intelligent Tutoring Systems*, pages 61–70.

Bittencourt, I., Bezerra, C., Nunes, C., Costa, E., de Oliveira, R. N., Tadeu, M., and da Silva, A. P. (2006a). Ontologia para construção de ambientes interativos de aprendizagem. In *XVII Simpósio Brasileiro de Informática na Educação, SBIE, Brasília*, pages 559–568.

Bittencourt, I., Costa, E., Nunes, C., Tadeu, M., da Silva, A. P., de Oliveira, R. N., and Sibaldo, M. (2006b). Um arcabouço para construção de sistemas baseados em agentes inteligentes. In *XX Semana Paraense de Informática, SEPAI, Belém*.

Bittencourt, I. I., Rocha, R. T., Abreu, C. G., Pinho, R., Ferneda, E., de B. Costa Lourdes M. Brasil, E., and da Silba, A. P. B. (2006c). Um sistema tutor inteligente multiagentes em anatomia Óssea do crânio. In *Publicação aceita no Congresso Brasileiro de Infromática Biomédica*.

Bittencourt, I. I., Tadeu, M., Costa, E., Nunes, R., and Silva, A. (2006d). Combining ai techniques into a legal agent-based intelligent tutoring system. In *Eighteenth International Conference on Software Engineering and Knowledge Engineering - SEKE, 2006, San Francisco*, volume 18, pages 35–40.

Chen, W. and Mizoguchi, R. (2004). Leaner model ontology and leaner model agent. *Cognitive Support for Learning - Imagining the Unknown*, pages 189–200.

Chepegin, V. Usermodelling. http://smi-protege.stanford.edu:8080/ KnowledgeZone/OntologyMetadata?ontologyid=22.

Costa, E., Perkusich, A., and Ferneda, E. (1998). From a tridimensional view of domain knowledge to multi-agents tutoring systems. pages 61–72.

de Almeida, H. O., Tenório, L. E. F., Costa, E., Barbosa, N. M., Bublitz, F. M., and Barbosa, A. A. (2004). Um arcabouço de software livre baseado em componentes para a construção de ambientes de comunidades virtuais de aprendizagem na web. In *Proceedings do XV Simpósio Brasileiro de Informática na Educação, Amazonas, Brazil*.

du Boulay, B. and Luckin, R. (2001). Modelling human teaching tactics and strategies for tutoring systems. *International Journal of Artificial Intelligence in Education*, 12:235–256.

Fontoura, M., Pree, W., and Rumpe, B. (2001). *The UML Profile for Framework Architectures*. Addison Wesley.

Kumar, V., Shakya, J., Groeneboer, C., and Chu, S. (2004). Toward an ontology of teaching strategies. In *Proceedings of the ITS'04 Workshop on Modelling Human Teaching Tactics and Strategies, Maceió, 2004*, pages 71–80.

Major, N., Ainsworth, S., and Wood, D. (1997). Redeem: Exploiting symbiosis between psychology and authoring environments. *International Journal of Artificial Intelligence in Education*, 8:317–340.

Murray, T. (2003). *Authoring Tools for Advanced Technologies Learning Environments: Toward cost-effective adaptive, interactive and intelligent educational software*, chapter An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art, pages 493–546. Number 17. Kluwer Academic Publishers, Netherlands.

P. Dillenbourg, J. S. (1992). A framework for learner modelling. *Interactive Learning Environments*, 2:111–137.