A Multi-Agent Based Framework to Build Intelligent Tutoring Systems

[1]Evandro de Barros Costa, [3]Hyggo Oliveira de Almeida , [1]Klebson dos Santos Silva, and [2]Angelo Perkusich

[1]Departamento de Tecnologia da Informação

Universidade Federal de Alagoas

Campus A. C. Simões, Tab. do Martins, Maceió -AL – Brazil, Phone: +55 82 214-1401

ebc@fapeal.br

[2]Departamento de Engenharia Elétrica, [3]COPIN

Universidade Federal da Paraíba

Campina Grande -PB – Brazil

Abstract. Intelligent agent approaches have been applied to designing and development of new ITS (Intelligent Tutoring Systems). However the design of such systems is mainly based on *ad-hoc* approaches. In this paper, we adopt a multi-agent approach which explores knowledge engineering and an important feature of software engineering based on design experiences: frameworks. Here, we describe a conceptual framework for designing of Cooperative Intelligent Tutoring System which adopts a multi-agent approach according to the model that we have proposed.

Keywords: Intelligent Tutoring System, Multi-agent Systems, Software Engineering, Distance Learning Environment

1. Introduction

The application of intelligent agents in many different, complex and dynamic domains has increased over the last years. This is mainly due to the flexibility, modularity, as well as the general applicability of this paradigm to a wide range of complex problems [HAYE 99]. The inherent knowledge complexity of Intelligent Tutoring Systems (ITS) is a key issue related to the increase in the use of the agent paradigm for the design of such systems. However, in most of the cases such design is carried on based on *ad-hoc* approaches.

In order to cope with such undesirable situation we introduce in this paper a framework based on a domain knowledge modelling approach together with software engineering disciplines to the design of multi-agent systems applications. Our focus is on the design of ITS in a distance-learning environment. The research reported in this paper in part of a larger project, named MathNet [COST 00], which is oriented to the design and development of a distance-learning environment based on a multi-agent approach. This project is based on the Mathema architecture [COS 96, COS 97, COS 98].

The major idea behind the MathNet system is to define and develop an environment that provides effective means to involve human learners, a tutoring system, and human teachers in productive cooperative interactions. In this paper our focus is to introduce the framework established to develop the society of agents, with emphasis on the design of a tutoring agent of this society. According to the proposed architecture and the conceptual framework. This architecture has been used to derive the abstract design of our framework. These tutoring agents are designed to offer effectiveness in the interaction process involved in teaching-learning activities. They provide personalised interactions with human learners working in the context of distance education and mainly focused on problem solving activities.

The remainder of this paper is organised as follows. In Section 2 we present the general architecture of the MathNet environment. We present our framework to model domain knowledge and its consequence in building a society of tutoring agents. Also, we define the Tutoring Agent architecture. A functional scenario to illustrate the agent architecture is presented in Section 3. In Section 4 we present a conceptual framework to model a Tutoring Agent. Finally, the conclusions are presented in Section 5.

2. The MathNet Environment and its Society of Agents

The general architecture of MathNet has been defined around a cooperative multi-agent ITS to provide human learners with cooperative learning. Learning is based on problem solving activities and their consequences leading to the accomplishment of other pedagogical functions, such as instruction, explanation, and hintsfor example. In

this context, we defined and developed a model for a computer-based cooperative interactive learning environment based on a multi-agent approach [COST01]. From an external view, the conceptual model of this system is organized as shown in Figure 1 and consists of five main entities: *Human Learner, Human Teacher, Human Expert Society (HES), Society of Artificial Tutoring Agents (SATA)*, as well as, an *Interface Agent* representing three different interfaces modules to assure interactions between theses entities: *Learner Interface, Teacher Interface, HES Interface*.

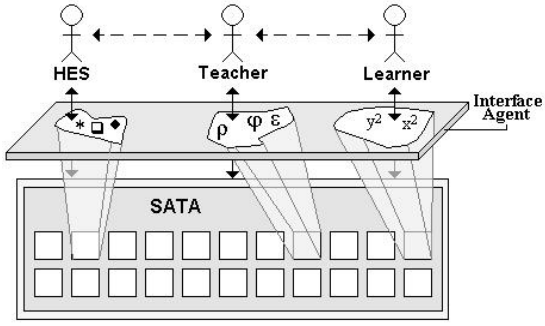

Figure 1. General Architecture of MathNet

Learner: an active human agent who is interested in learning about certain domain knowledge. For example, the domain of traditional music harmony [Cost 01].

Teacher: a human agent responsible for playing a facilitator role for promoting assistance to the learner and interacts directly with him.

SATA *(Society of Artificial Tutoring Agents)*: Responsible for assuring productive interactions with Learner/Teacher. This society represents the multi-agent ITS.

HES *(Human Experts Society)*: works as sources of knowledge to the SATA, being responsible for building and maintaining the tutoring agents from SATA.

Learner Interface: this interface allows the communication between the learner and the system. All interactions are achieved through a browser.

Teacher Interface: Allow communication between the teacher and the system, and as well as the learner interface, it is accomplished through a Web browser.

Expert Interface: Responsible for communication between HES and SATA.

Each one of these three categories of interface agents is defined to provide both a microworld related to a given domain knowledge and a set of primitive operators to manipulate the objects defined for this microworld, as for example a turtle graphic in Logo [Pape 94]. Moreover, we establish two levels of languages.

2.1. Domain Knowledge Modelling

Considering the need for improvement and effectiveness in the interaction process between the learner and the computer system, taking into account productive and adaptive interactions, we have addressed some quality requirements towards the

design of a good learning environment. These quality requirements include searching for good qualities in domain knowledge. On the other hand, a great part of researching in Cooperative ITS deals with questions related to: (i) model of learners which allow a Tutoring System to provide individualised actions and adaptive interactions which focuses on building open and inspectable learner model [BULL 95], [PAIV 95]. This leads to the possibility of learners to inspect the learner model, which the system has made of him, having the opportunity to discuss and change the results from this model. Other important question in ITS claims to (ii) multiple representation of domain knowledge [CUMM 91] and (iii) multiple pedagogical strategies [GIRA 99]. Based on these requirements, we have first worked on the improvement of the domain knowledge (DK), taking into account a trade-off between its richness and its structure [COST 98].

We propose a particular way to consider multiple views on *DK* and then providing it with a suitable organization in terms of computational structure. Then, a particular *DK* is viewed as a set of interrelated sub-domains. Considering a goal-oriented approach, in a teaching/learning situation, we define three dimensions for *DK*: *context*, *depth*, and *laterality*. The first one, the *context* provides multiple representations of knowledge by leading to different interpretations. The second one, the *depth*, provides different levels of language refinement of DK. The last dimension, the *laterality*, provides dependent knowledge that is, in this work, considered as prerequisites and co-requisites of a particular *DK*. Once established this organization, we define a *DK* decomposition into sub-domains and identify microworlds and tutoring agents from this decomposition to approach *DK*. This organization is defined according to the following scheme: *DK* can be associated with different contexts $(C_1, C_2 \ldots, C_n)$, where each C_i, with $1 \leq i \leq n$, corresponds to different depth levels defined by $(D_{11}, D_{12}, \ldots, D_{mi})$. Finally, to each pair $\langle C_i, D_{ij} \rangle$, $1 \leq j \leq m$, different laterality $(L_{111}, L_{112} \ldots, L_{ijt})$, with $t \geq 0$, are associated.

From the dimensional view of *DK* plus some considerations about the requirements mentioned above, it is necessary to manage a big complexity in the involved knowledge. To deal with this complexity and its consequences in the co-operative interaction process, we have adopted multi-agent approach. This approach seems to be natural to that end by offering various benefits from knowledge engineering. Also, it offers suitable techniques apparatus and methods from the field of Distributed Artificial Intelligence.

Then, we define a *DK* decomposition into various sub-domains *d* via the following criteria: from each C_i we define m sub-domains by the following pairs: $\langle C_i, D_{i1} \rangle$, $\langle C_i, D_{i2} \rangle, \ldots, \langle C_i, D_{in} \rangle$, with $1 \leq i \leq n$, and to each one of them we associate various support sub-domains d_{ijk}, $0 \leq k \leq t$, responsible for knowledge in laterality dimension. Therefore, we have defined two kinds of sub-domains: (i) from $\langle C_i, D_{ij} \rangle$ view we define the sub-domain d_{ij} and (ii) from the lateralities identified to each $\langle C_i, D_{ij} \rangle$ view, we define *L* knowledge support domains d_{ijt}. Then, we can identify the agents through the following criteria: to each sub-domain d_{ij} we define an agent Ag_{ij}

endowed with knowledge on this subdomain and on the microworld that we define associated with this subdomain. The same procedure is used to each lateral support domain d_{ijt} , defining a lateral support *agent* Ag_{ijt} having specific knowledge on this domain and each microworld defined. In short, we have three steps: (i) identify subdomain, (ii) define a microworld associated with this subdomain, and then (iii) define an agent to work on this subdomain and on this microworld. These agents are connected by means of dependency relations. These relations allow the identification of different interactions among agents.

2.2 The Society of Agents and Agent Architecture

The society is an open multi-agent system. It is made up of a collection of tutoring agents that may, through established protocols [COST 96], co-operate among themselves to achieve the teaching/learning activity to promote the learning of a certain human learner. This society is designed to be open and dynamic in the sense that it allows maintenance operations such as the entry and the exit of agents, besides eventual modifications in the knowledge and in the inference mechanisms of an agent [COST 99]. Any agent is an ITS that have the necessary knowledge to achieve pedagogical tasks related to a particular domain. These agents are cognitive and possess properties such as autonomy, goal-oriented, and social ability [FRAN 96].

The architecture of a tutoring agent is composed by three main components: a *tutoring system*, a *social system*, and a *distribution system*. The tutoring system (*TS*) controls the cooperative interactions between a tutoring agent and a human learner/teacher. The social system (*SS*) coordinates the cooperative activities, reflecting the social behaviour of a given agent. The distribution system (*DS*) executes the sending/receiving of messages through the communication environment. In what follows we the architecture and the functionality of a tutoring system as a component of an agent is detailed. The interested reader may refer [COST 96, COST 97] for details about the social and distribution systems.

3. Architecture and Functional Scenarios of Agent Systems

3.1. Tutoring System

The tutoring system (*TS*) is responsible for the cooperative interactions between a tutoring agent and a human learner/teacher, in learning activities. This module is composed by the components described below.

Mediator: Responsible for interacting with the Learner through the Interface Agent. To do this it is formed by two mechanisms. One is responsible for interpreting the actions from the Learner and then selecting a particular Reasoner. The other mechanism maps a microworld concerning the domain knowledge.

Reasoners: This module is responsible for providing all the pedagogical functions. It is composed by the submodules described in the following subsections.

Expert Module: This module is responsible for problem solving and explanations of the solutions. The functionalities provided by the this module are played by the following sub-modules.

Inference Engine : responsible for reasoning on two knowledge bases: Correct Rules Base and Mal-Formed Rules Base (which contains rules about some kind of misunderstanding of the domain).

Tutor Module: This module is responsible for directly interacting with the Learner by selecting pedagogical resources from a curriculum structure defined over the domain. For accomplish their functionalities it is divided into sub-modules, described below.

Evaluator : *Responsible for evaluate the learner answers.*

Pedagogical Tasks Manager **:** Manages the pedagogical knowledge stored in the database, choosing the resources (concepts, examples, exercises, etc) needed to the learning of a certain learner.

Remediator: Responsible for choose the next action of the system according to the learner cognitive diagnostic.

Learner Modelling Module: It is responsible for acquiring, maintaining, and representing informations about individual learners. These informations are useful in order to tutor module makes pedagogical decisions. This module implements a learner model, which allow a Tutoring System to provide individualised actions and adaptive interactions. It takes into account the possibility of learners to inspect the learner model, which the system has made of him, having the opportunity to discuss and change the results from this model. Also, it may implement a distributed diagnosis mechanism based on the multi-agent platform available. For achieve their goals, it is divided into submodules, described below.

Historical Manager **:** Responsible for organization of all pedagogical knowledge viewed by the learner.

Profile Manager: Responsible for updating the learner level, considering the new resources and his performance in problem solving.

Cognitive Diagnostic: Responsible for inferring the learner knowledge about a specific domain previously taught for the system.

Inspection: Responsible for inspection mode. In other words, if the learner disagree about his profile (controlled by the Profile Manager), then the system have to show those exercises whose learner answers were wrong, and ask for the resolution of similar problems.

Resource Bases: This module is responsible for providing the necessary knowledge to be support to the reasoners.

Let us consider a scenario where the Tutoring System has the interaction control. Once the Learner is connected to the system, the Tutor module starts an interaction session with him by assuring his login.

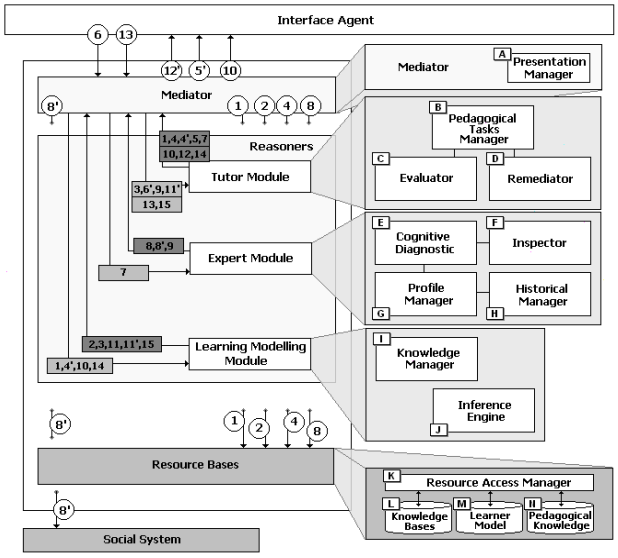

Figure 2. Functional and Internal View of a Tutoring System Architecture

1 – The **Tutor Module** through the **Pedagogical Tasks Manager (B)** requests **Learner Modelling Module** the next resource to be showed to the Learner.

2 – The **Learner Modelling Module** through the **Profile Manager** (**G**) recovers information from the **Learner Model** (**M**), via **Resource Access Manager (K)**, and sends the informations to the **Historical Manager** (**H**). Then **Historical Manager** (**H**) check whether the Learner covered, in the last pedagogical session, all the resources contents established.

3 – Case the content has not been covered as a whole; **Historical Manager** (**H**) returns the kind of resources remaining to the **Pedagogical Tasks Manager (B)**. Also, it returns a list of resources already presented to the Learner in order to **Pedagogical Tasks Manager (B)** can generate the review content.

Otherwise, **Historical Manager** (**H**) sends the profile informations to the **Cognitive Diagnostic (E)** and returns to **Pedagogical Tasks Manager (B)** a list of resources before presented to the Learner, generating review content.

The **Cognitive Diagnostic (E)** processes the informations based on a set of rules defined by the expert (HES) via **Knowledge Manager (I)**, obtaining the current learning level from Learner.

Following the sequence, **Cognitive Diagnostic (E)** sends this learning level to **Remediator (D)**. Finally, **Remediator (D)** sends back to **Pedagogical Tasks**

Manager (B) what kind of resources should be presented to the Learner based on his level.

4 – Having the kind of resources, **Pedagogical Tasks Manager (B)** retrieve a resource from **Pedagogical Knowledge (N)**, through **Resource Access Manager (K)**, and check whether that resource has ever shown to the Learner, via **Historical Manager (H)** (step 4'). Case the Learner has ever seen it, **Pedagogical Tasks Manager (B)** retrieve other until find one that Learner never saw.

5 – **Pedagogical Tasks Manager (B)** sends that resource to the **Presentation Module (A)** in the **Mediator**, that build a graphical component that contains the resource and sends it to the **Interface Agent** (step 5').

6 – Case the resource be a problem, the Learner must answer the problem interacting with **Interface Agent**. Then, **Interface Agent** sends the answer to **Mediator** that forwards it to **Tutor Module** (step 6')**.** Otherwise, goto step **1**.

7 – **Tutor Module**, through **Evaluator (C)**, check whether the answer is correct: it sends to the **Expert Module** the informations about the problem and the Learner answer.

8 – The **Expert Module** by means of the **Inference Engine (J)** works on **Knowledge Bases (L)** through **Resources Access Manager (K)** in order to try validating the solution.

Case the current agent do not be sufficient to validate the solution in an isolated way, it tries to cooperate with other tutoring agents. This is done by the **Inference Engine (J)** requesting **Social System** to searching for agents in **SATA** (step 8').

9 – Once the solution is validated, **Inference Engine (J)** returns to **Evaluator (C)** both learner performance and the path of prove followed by the system.

10 – The **Evaluator (C)** sends to **Profile Manager (G)** the performance obtained in the evaluation process. It also sends to the **Interface Agent,** by means of **Mediator** (using **Presentation Module (A)),** performance and path by which traced during the prove which identified the performance.

11 – The **Profile Manager (G)** updates the **Learner Model (K)** through **Resources Access Manager (K)** and requests **Pedagogical Tasks Manager (B)** to show the Learner his current learning level and performance concerning the resources worked (step 11').

12 – **Pedagogical Tasks Manager (B)** requests **Mediator** (using **Presentation Module (A)**) to show these informations via **Interface Agent** (step 12'). At this moment, the Learner may discuss and change (by a negotiation mechanism through try again to solution a given problem) the learner model interacting with **Mediator**.

13 – In case of learner disagreement, **Mediator** forwards the request to the **Pedagogical Tasks Manager (B)**.

14 - The **Pedagogical Tasks Manager (B)** requests inspection to the **Learner Modelling Module**. This module, will verify through the **Inspector (F)** whether the learner had some problem during problem solving over similar problems.

15 – Case the **Inspector (F)** find some occurrence of these problems, a new resource will requested to **Pedagogical Tasks Manager** to be presented to the learner as a second opportunity in problem solving process (similar problems). In case agreement by the learner regarding to the Learner Model result, a new resource is requested to the **Learner Modelling Module.** (goto step **1**).

3.2. Social System

This module is composed by the following components.

Task Allocator: responsible for choosing agents to solve tasks received from Tutor module. This module returns a set of pairs $<T_i, LA_i>$, where T_i is a task and LA_i is a set of agents able to solve T_i.

Social Knowledge Manager: contains the agent's knowledge about other agents in SATA.

Coordinator: controls the requests of cooperation to solve tasks, taking into account possible dependency between them. In other words, the solution of a task T_i can depends of another task. So, this module is responsible to manager this situation.

Cooperation Manager: responsible for execute a cooperation. To do that, this module contacts other agents in SATA using pre-defined interaction protocols.

Dialog: represents an instance of communication established with another agent. This module maintains the context of the connections using in this communication.

The figure 3 illustrates the interaction between the social system components.

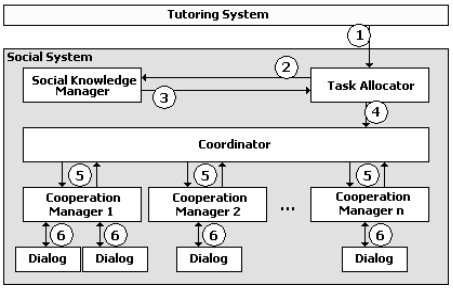

Figure 3: Functional and Internal View of an Social System Architecture

1- **Task Allocator** receives a set of tasks.

2, 3- **Task Allocator** consults the Social Knowledge during process of search agents able to solve the tasks received previously

4, 5 – **Coordinator** receives the set of pairs $<T_i, LA_i>$. It then controls the creation of **Cooperation Manager** processes.

6- Each one of the cooperation processes requests help to other agents in SATA. An instance of module **Dialog** will be created for each one of agents.

4. Conceptual Framework

We introduce a framework to build ITS based on the identification of the common characteristics observed in applications developed in the context of MathNet environment [COST 01] and on its architecture. We present in this paper a high-level description of such framework. Observe in the class diagram shown in Figure 6 that classes with stereotype abstract define the hooks of the framework. Due to space constraints we are not detailing these hooks in this paper. The application of the framework is based on the composition of the functionalities of the modules of an agent. Therefore, new functionalities can be added by inserting new modules in such a way that they do not cause problems in the execution of the modules already defined. Moreover, these modules will be compatible with and will be part of the system as a whole. In what follows we have done some considerations about the modules in the architecture. The *Mediator* controls all messages exchanged between reasoners. The request and reply messages are sent to the mediator, and then forwarded to the suitable reasoner. Based on the architecture already presented, we define fixed points in reuse of this architecture for design new ITS´s. The definition of agent (section 2.3) should be remained, then, tutoring, social and distribution systems represent fixed points in the framework. Moreover, their internal components are also fixed in terms of funcionality, but the implementation of this funcionalities is not fixed. This is possible due to *Hooks* placed in personalization points on the functional components of the framework (social, distribution and reasoner components). Here, we are not interested in presenting this hooks with low-level details concerning design phase ready to derive implementation aspects. An example of the use of hooks is shown in Figure 5, where a component may be any functional component in the proposed framework or an abstract component, in case of additional components:

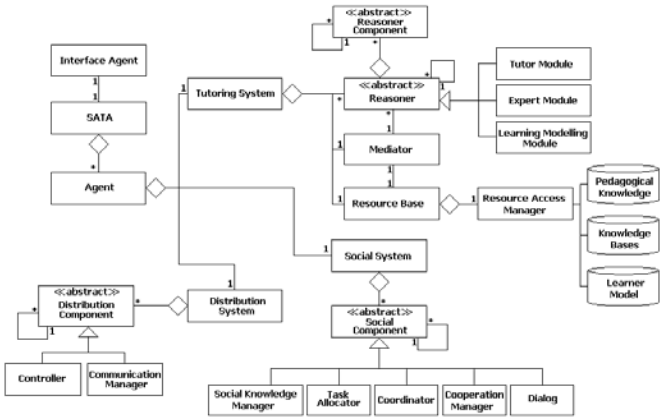

Figure 5. Class diagram of the framework

In the diagram illustrated in Figure 6, abstract classes (with abstract stereotype) define the default behaviour of the functional components and represent the hooks of

framework. The functionalities are preserved for the entire because the components behaviour are also preserved. New components may be added, or changed for others with for example better performance in runtime, therefore there is no need to stop the execution on an agent in order to change components.

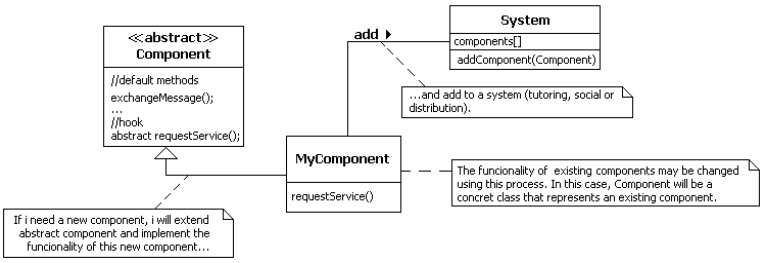

Figure 6. An example of hook

5. Conclusion and Future Work

In this paper we have presented a framework for building a society of tutoring agents from a domain knowledge model. Based on the architecture defined to a tutoring agent, a framework for designing a tutoring system from a tutoring agent has been defined. The first results from this framework have been used in the development of four multi-agent ITS prototypes. These prototypes have been developed and experimented in domains such as Musical Harmony [COST 01], Information Structure [FARI 00], Algebra and Classical Logic [FLEM 97].

Further research includes extending these frameworks by defining them in terms of implementation approach. Moreover, we have been working on the definition of a methodology based on the conceptual framework. We have worked aiming the three phases: analysis, design, and implementation.

References

[BULL 95] Bull, S.; Pain, H. "Did I say what I think I said, and do you agree with me?": Inspecting and questioning the student model. Proceedings of AI-ED 95 - World Conference on Artificial Intelligence in Education, Washington, DC; August 16-19, 1995, pp. 501-508.

[COST 95] Costa, E.B.; Lopes, M.A.; Ferneda, "E. MATHEMA: A Learning Environment Based on a Multi-Agent Architecture". In J. Wainer and A. Carvalho, editors, Proc. of 12th Brazilian Symposium on Artificial Intelligence, Volume 991 of Lecture Notes in Artificial Intelligence, pages 141-150. Springer-Verlag, Campinas, Brazil, October 1995.

[COST 96] Costa, E.B., Perkusich, A. "Modelling the Cooperative Interactions in a Teaching/Learning Situation". In Proceedings of Third International Conference on Intelligent Tutoring Systems - ITS'96, Montreal, Canada, June, 1996.

[COST 97] Costa, E.B. "Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura Multi-agents". Tese de Doutorado. UFPB, Campina Grande-PB, 1997.

[COST 98] Costa, E.B.; Perkusich, A.; Ferneda, E. "From a Tridimensional view of Domain Knowledge to Multi-agent Tutoring System". In F. M. De Oliveira, editor, Proc. of 14th Brazilian Symposium on Artificial Intelligence, Volume 991 of Lecture Notes in Artificial Intelligence, LNAI 1515, pages 61-72. Springer-Verlag, Porto Alegre, RS, Brazil, November 1998.

[COST 00] Costa, E.B. "MathNet: Uma Abordagem via Sistemas Multi-Agents para Concepção e Realização de Ambientes Interativos de Aprendizagem Cooperativa Assistidos por Computador. Revista de Informática na Educação da Sociedade Brasileira de Informática na Educação. SBIE, Abril de 2000.

[COST 01] Costa, E. B., et alli. "SHART-Web-Um Sistema Tutor Multi-agents em Harmonia na Web". SBIE´2001: Workshop on Multi-agent Interactive Learning Environment, Vitória, Espírito Santo.

[CUMM 91] Cumming, G.; Self, J. "Learner Modelling in Collaborative Intelligent Educational Systems". In: Peter Goodyear (ed.), Teaching Knowledge and Intelligent Tutoring, Norwood, N.J.: Ablex, 1991.

[FARI 00] Faria, T. de F. e Bittencourt, G. "Um ambiente interativo multi-agents para o ensino de estrutura da informação", XI Simpósio Brasileiro de Informática na Educação (SBIE'2000), Maceió, AL, 8 a 10 de novembro de 2000.

[FLEM 98] Flemming, E., "Um Sistema Tutor Distribuído no Domínio da Lógica", Universidade Federal de Santa Catarina, CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, Área de concentração: Sistemas de Informação, 29 de abril de 1998.

[FRAN 96] Franklin, S.; Graesser, A. "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.

[GIRA 99] Giraffa, L. M. M. ; "Uma arquitetura de tutor utilizando estados mentais". Tese de Doutorado, Universidade Federal do Rio Grande do Sul, 1999.

[HAYE 99] Hayes, C. C.; "Agents in a Nutshell – A very Brief Introduction". In IEEE Transactions on Knowledge and Data Engineering, Vol. 11, n. 1, January/February 1999.

[PAIV 95] Paiva, A., Self, J., Hartley, R., "Externalising Learner Models". In: Proceedings of AI-ED 95 - World Conference on Artificial Intelligence in Education, Washington, DC; August 16-19, 1995, pp. 509-516.

[PAPE 94] Papert, S.: Mindstorms: "Children, Computers, and Powerful Ideas". Basic Books. New York, 1994.