

Modeling and Implementing Intelligent Educational Environments Using an Interdisciplinary Approach

Lucia M. M. Giraffa

giraffa@inf.pucrs.br

Post-Graduate Program on Computer Science (PPGCC) -Faculty of Informatics - PUCRS
Av. Ipiranga 6681 – Prédio 16 – 90610-900 -Porto Alegre – RS - Brazil

Abstract

This paper presents the results that we have been achieving with our research involving educational environments based on Intelligent Tutoring Systems (ITS) architecture using a MAS (Multi-Agent System) approach. A general guideline was idealized to be used as a reference for our research group to design and to implement intelligent educational software. Specially, interactive educational software modeled as a game. We believe that now we have the agent's technology that makes possible to build interesting solving problems environments. However, it is very important to remember that educational process is closely connected with country culture. Educational environments that work very well in a specific context will not necessarily perform in the same way in a different place. It can be viewed as a restriction in many ways: as applicability for other educational reality, pedagogical paradigm, and so on. However, we argue that educational process has a "kernel" of necessities and aspects to be observed. When you intend to work considering learner as central person in the educational process, under the "learn-to-learn" paradigm. So, it causes an immediately reflection in the way we design and model an educational software.

We discuss our ideas and guidelines used to model and to implement our systems under pedagogical viewpoint. We explore the possibilities of designing such systems using a MAS (Multi-agent) approach in order to explore the possibilities of this technique. In our proposal the domain is modeled with reactive agents and some of them we applied techniques of Machine Learning (Reinforcement Learning). The students and the set of pedagogical agents (task agents, assistants, and tutors) were modeled as cognitive agents using BDI architecture (belief, desire, and intention). Due to the space we are not describing the systems in details, but exemplifying how the guidelines were used.

1. Introduction

This paper intends to describe a general guideline idealized to be used as a reference for our research group to design and to implement educational software. Most of the ITS (Intelligent Tutoring Systems) and ILE (Intelligent learning Environments) were built for procedural subjects, such Mathematics, programming, and Physics. The difficulties to build ITS for other domains like Biology, Chemistry, and Ecology, are connected with the phenomena's representation. Our proposal explores the possibilities of using simulations built with agents to model these phenomena, create interfaces that make possible for students to visualize the results of their actions, and a help system to aid the student to understand the system behavior itself. Before you go further on this paper we would like to say that you will not find comparisons concerning our work with related

works, we did this in other previous works [2; 3; 4; 5; 6; 8]. Of course, we have been reading a lot of related work, but the goal of this paper is to describe the way we have developed our systems. Our intention is not to claim to ourselves the notoriety of having something that nobody had though before. Our intention is to exemplify that we can develop educational software with quality if we have a policy to guide our research. Our effort showed us that it is possible. We explore the possibilities of designing such systems using a MAS (Multi-agent) approach in order to explore the possibilities of this technique. However, we do believe that educational software cannot be significant for educational activities without considering the pedagogical aspects (methodology, strategies, educational paradigm, and so on). The artificial intelligent techniques are extremely important but they are not enough to guarantee the quality of the educational software.

The traditional paradigm adopted in schools was based on to teach knowledge made by specialist, and the role of the students was to learn how to solve problems with the knowledge and solution made by others. Nowadays, we have a new proposal, a new paradigm: teach the students to understand the problems, its features, and how s/he can create solutions for those problems. It is a turning point: changing the focus from the teacher to the student. In fact it is not a new idea. The new medias integrated by computers offer us an opportunity to create new environments/methodologies in order to improve the learning process, and overcome limitations of the traditional resources (blackboard, charts, graphics, and so on) founded in schools. We are not arguing that traditional methodologies are not efficient because they are "classical". We are saying that some subjects (as we mentioned before) are resource bounded, e.g., they are very difficult to explore using the traditional resources. Nowadays, our kids have such familiarity with computers and related technologies that, for them, it is easier to use software than to pay attention at the blackboard. The educational software research began on the 40's of 20th century. The earlier programs were very simple, and had poor interfaces. We have more than sixty years of research. Many good things have been done in order to create good and powerful educational systems. The technology evolution had a side effect on the new environments. If we consider only the introduction of AI (Artificial Intelligence) techniques we find a lot of interdisciplinary work. The ITS (Intelligent Tutoring Systems) was the first educational systems that had an architecture proposal. It caused a great impact on

educational software design. The idea behind the ITS architecture is to assist the student in a personal way. It means, design a system that can be adapted according to the student behavior. However, the expectations were not enough fulfilled. There is a big gap between the system that can be implement, and the systems that are desired to be built. If we search on the Internet, and on the conferences proceedings we will see that it is rare to find ITS for domains like Chemistry, Biology or Ecology, and with a construtivist approach. This is due to the set of computational prerequisites like abilities to handle with the complexity of modeling their phenomena, to select the set of strategies related when the students make mistakes, and to infer his/her misconceptions (tutor behavior rules). It is necessary to remark that it is important for the student to understand why his/her action did not reach the desirable result. It is also important to design interfaces that can be used as a way to offer quick feedback for students, and adequate help systems. If all this aspects are assembled together you will have a very complex system to make. Design an ITS is not an easy task to perform.

This paper is organized as follows: section 2 presents the guidelines to design educational software. Section 3, presents some aspect related with the use of agent's techniques, and the testbeds implementation. Section 4, presents the final considerations and future works. Section 5 presents the references.

2. Guidelines to Design Educational Software

Since the beginning of the research we observed the gap between educational software design and other applications like commercial or industrial systems. Almost all the systems have a formal modeling and they use Software Engineering resources. However, with educational software projects it is the opposite. It is very rare (and recently) to find an educational environment which observes methodological issues, even under Computer Science (CS) or Pedagogical viewpoint. Due to this gap between educational community and other fields, the author decides to create some guidelines to model and design educational software. These guidelines were refined along the years, during many projects under our supervision. To develop educational software we divide the process in two parts:

The definition about the environment concerns to pedagogical viewpoint: This first part is very important to observe because the main decisions about the system will be done. The elicitation process will give elements for the second part. We argue the importance to clarify those elements before working with a more formal process.

The formal system modeling related with CS (methodologies). After deciding the main guidelines and specifications concerning the software pedagogical aspects, it is necessary to specify the sequence of activities to be followed during the developing process. It is also necessary to identify the application life cycle, and

compare it with the real evolution of the system. The developing methodology (life cycle) is different from the technique or method selected to specify the system itself. The methodology must have a certain degree of independence comparing to the set of chosen techniques. This part is not finished yet; we are still working on it. In this section we explore the steps related to attend the first part of the guidelines:

1. Select the subject (domain). We believe that it is impossible to create good educational software to general purpose or a large set of contents. We need to select a specific subject to be modeled in a deep way. Especially when we try to create environments based on Intelligent Tutoring Systems architecture, where we need to model students and tutor interventions related with students mistakes. It is very hard to create a good choreography. It means, the set of activities to be taken by an artificial tutor/assistant when it needs to help the student to understand why his/her action did not solve the problem.

2. Select the system users. It is important to define the age of the users (students). We believe that it is impossible to model a general environment that is suitable from "six to sixty" years old. Each group has special characteristic that must to be taken into account when we define, for example, interfaces, user interaction, the way to present some tips, and so on. _

3. Select the modality of educational software. It means decide what kind of software will be adopted: tutorial, drill and practice, simulation, educational games, and so on. For us the traditional taxonomy (and its variances) that divided the systems into tool, tutor and tutee are not suitable with the current reality. Those taxonomies were related with the classical educational paradigm that classifies software according to their resources and interfaces characteristics. Nowadays, it is very hard to classify an educational software using those taxonomies, because the interfaces integrates many resources, the systems use to have different activities and modules, and the systems offer different pedagogical possibilities to be explored by students and teachers. We think it is time to analyze the environments observing their potential to achieve educational goals. Today there is more than just a single program, they become environments with many resources and options to be explore, even by teachers (using or creating new methodologies) or by students (testing hypothesis, discovering facts, and so on).

4. Select team members. After select a subject, it is necessary a specialist for domain modeling. S/he is responsible to select exercises, strategies and tactics related to student's mistakes. Educational games modeling need also a specialist to help with the interface (the specialist in Computer Graphics and simulation), and last, but not the least, a specialist in Informatics applied to Education.

5. Design the interface. The interface is the communication window between the user and the system. It must be carefully designed. We depart from the interface elements to guide all the system design. Our

experience showed us that it is very important to test the interface with the users and teachers to better guarantee that the interaction will happen according to our expectations. The interface design and implementation must be performed using an incrementally approach. We have been using the guidelines proposed by <http://www.useit.com/papers/heuristic>

6. Select the environment to implement the system (tool or set of tools). The choice of specific tools or environment to implement the system is related with environment characteristics. The experts of CS support are very important after the decision of the steps above. Currently, there are many possibilities to implement an environment. The specialist's advice is the best and safest way to avoid future disappointments. They have the knowledge to analyze the role project and to indicate what is better, taking in account the hardware where the system will be performed. Sometimes we have to drop the use of the ultimate technology because it is not available in schools. We must remember that our client is there. For sure, the gap between University computers and regular schools equipment's must be taken into consideration. Probably, it is not a local (Brazilian) problem.

7. System evaluation. We need to perform some kind of evaluation process in order to guarantee the quality of the system under educational viewpoint. We can design and implement a great system under Computer Science viewpoint. However, it might not be so good under educational viewpoint. As we said before we must consider at the same time both aspects. In our experience, we faced with some situation where the solution was not so creative or good enough under system perform (CS aspect). However, it was necessary in order to guarantee system quality under the pedagogical viewpoint. In an interdisciplinary team, sometimes it is very difficult to handle this. Sometimes the team members seem not to agree with the final decision that must be taken. Evaluating a system under pedagogical viewpoint is something quite hard to do. If you decide to measure the students learning acquisition it cannot be done in a short period of time. The student's learning evaluation is a process that always remains after concluding a prototype connected with an ungraduate final work, a Master Dissertation or a Ph.D. thesis. To measure how much a specific system helped the students to acquire knowledge and skills about something we need a rigorous scientific methodology with statistics analyze, group's control, and time, lots of time. So, what can be done in order to perform some kind of system evaluation under educational viewpoint? We have developed, also, a method to perform what we call the "evaluation of pedagogical aspects". It was applied in real school environment (classroom). We created an instrument (questionnaire), under Psychologist and Education supervision, to identify which system aspects are adequate or suitable to be explored by a special methodology, during the learning activities. The results for these evaluations made with our systems (see section 3) showed

us the potentiality of such instruments. They can be adapted in an easy way in order to evaluate different programs.

3. The guidelines in practice

We had focused our efforts to build educational systems modeled as solving problem environments. We chose to model educational software inspired on ITS based on multi-agent system architecture (MAS) and game like fashion interfaces. From ITS architecture we explore the idea of student model to personalize the interaction between system and users. From the games and the entertainment applications we explore the interfaces. From MAS approach we explore the domain modeling to create better conditions to be explored by the student, and the reasoning process of the cognitive agents (students, assistant, and tutors) were modeled using a BDI approach [2,3,4,5,7]. The agent's approach allows us to handle with the complexity of the phenomena in a better way. The intrinsically idea of MAS systems decreased system computational and modeling cost. Our research group has an important work made by [7] that allow us to model and to implement cognitive agents using a specific tool. This tool, named X-BDI, was a "milestone" for our group. Many following works have been done using this tool in order to program our BDI agents [3,5]. It also helps us to spread out the mentalistic approach. This section presents some testbeds that allow us to evaluate our guidelines and better understand the complexity intrinsically in this activity (developing educational software concerning technical and pedagogical aspects). Due to the space restriction only a general idea will be given of each system and their goals. More details can be obtained on the related URLs. The systems interface was written in Portuguese (Brazilian official language). However we have some documentation written in English (papers and technical reports).

ECO-LÓGICO is a game, which the domain application is Ecology (water pollution). In this ecological environment the student choose a character to play using six options: Mother Nature, Mayor, Citizen, Fish, Ecologist, and Tourist. After that, the student defines the game configuration (foreign elements that will cause pollution) by a selecting process from a graphical menu. The game was modeling to be played by one student. The system was tested in real situation, i.e., with students from an elementary school working with environment as a complementary activity during their classes. The results allow us to measure interface limitations and possibilities. It has been used for several schools in our city, and it is spread out in Brazil as well. The rules used on the domain model were used to MCOE system, the multi-agent version of Eco-Lógico.

<http://www.inf.pucrs.br/~raabe/eco-logico/>

MCOE (Multi CO-operative Environment): The conception of the 'MCOE' was based on an ITS architecture proposed by Giraffa [2,3]. This architecture is composed by a hybrid society of agents that work to

achieve a common goal: to fight against the pollution resulting from foreign elements (pollutants) and to maintain the ecosystem equilibrium. We have reactive agents (bottom of the lake, microorganisms - plankton, water, plants and three types of fish) and cognitive agents (the Tutoring agent, and students represented by the characters). The cognitive agents are modeled using a mentalistic approach, where the term agent means a computer system that can be viewed as consisting of mental states such as beliefs, intentions, motives, expectations, obligations and so on. The model of each student and the tutor contains a set of basic beliefs, desires and expectations¹. From this set of mental states emerges the dynamic selection done by the tutor to select a personal teaching strategy. We designed the system to be played by two students. The environment was tested and compared with Eco-Lógico and, the results showed us that using agents improved users interaction and enlarged the pedagogical possibilities.

www.inf.pucrs.br/~giraffa/mcoe/mcoe.html.

TCHê: This educational software (game interface) has the purpose to support students from elementary schools (first grade) to understand how to solve basic Mathematics problems. The game was designed to present a set of activities that can be performed during a configurable time. The time adjust is an important approach adopted since MCOE system that guarantees flexible tuning according different students profiles. The set of activities is divided in four scenarios (seashore, city, mountains, and countryside). We used agent techniques to model and to implement the scenario elements (reactive agents). <http://www.inf.pucrs.br/~giraffa/tche/index.html>

E-MCOE (Extended MCOE): It integrates our previous result reached with Eco-Lógico, MCOE, and TCHê. The domain was improved with Machine Learning techniques (Reinforcement Learning algorithms) proposed by [1]. The fish now can learn different behaviors like avoid the predators, search for food, etc. It caused an impact in the way that we model the strategies to be used by the tutor, and generates a set of information that were not modeled in advance in the system. E-MCOE was built using an extended architecture used to design MCOE system. It includes a new agent named Mediator created to help the tutor to handle with this new information. To better programming the cognitive agents with BDI architecture using the X-BDI tool we developed a technique to specify and to implement the BDI agents [5]

The design of these environments described above created the necessity of having technical solutions in order to attend the pedagogical aspects. It allows us to improve our previous research in Distributed Artificial Intelligence field. That is another aspect of our interdisciplinary team members, besides we have people from Education, Psychology, and Communication, we have people from different areas of Computer Science. This interaction

allows us to co-operate in different ways and enrich our experiences.

4. Final considerations

It is very important to remember that educational process is closely connected with country culture. Educational environments that work very well in a specific context will not necessarily perform in the same way in a different places. It can be viewed as a restriction in many ways: as applicability for other educational reality, pedagogical paradigm, and so on. However, we argue that educational process has a “kernel” of necessities and aspects to be observed when you intend to work considering learner as the central person in the educational process, under the “learn-to-learn” paradigm. The kernel is related to the way the system is designed, and the set of beliefs about education that the community has. Independent of the country or culture, the steps suggested on section 2 can be taken in account. On ITS the steps are expressed on the strategies and tactics used to conduce the interactions between tutor and students. In a MAS approach it will be expressed by a society of agents that performs the roles related to learners and tutor. For us it is a matter of selecting roles in the multi-agent society connected with the pedagogical choices. How many agents do we need? What are their goals? Are they permanent? Are they temporally? You can just answer questions like these after decide the pedagogical issues of the systems.

The guidelines suggested on section two were followed in a rigorous way for all systems mentioned on section 3. Each software took in average 18 months to be modeled, implemented (prototype), and tested. All prototypes and related documentation are available to be downloaded on the mentioned URLs. Those homepages mentioned above were created to offer an opportunity for the Education research community to check our results and to collaborate with us through their critical analysis. All the systems were spread out in different schools (in different Brazilian states).

The use of agent’s techniques helps us to achieve this degree of sophistication necessary to our projects. In case of ITS and in Intelligent Learning Environments (ILE) we can consider agents as Pedagogical with some fundamental properties: autonomously, social ability, proactiveness, and persistence. Some of them can be reactive, continuously performing, capable to learn and a character represents most of them. This set of properties gave characteristics for the pedagogical agents that are very important under pedagogical point of view. In the reactive agents this properties improve the quality of domain representation, increase the feedback for the student’s actions and their strategies in order to solve the problem. The possibility to show the environment in different aspect using agent’s creates the feeling that each time the system is loaded you have a new challenge with unpredictable situations to solve. Even the characters are the same; the lake and its elements will not be equal as previous section. The user has a quick feedback that

¹ The expectation is considered for us as a future belief.

allows him/her to make reflections about his/her actions and its consequences. By the side of cognitive agents, the autonomy guarantees the conditions in order to achieve their personal goals. In an educational environment the agent has to analyze its own actions and the reflections of this on the environment (proactiveness property), and took in account to act again. The social ability are implicit in the keyboard actions (press a selected tool from a character), and codify in the set of beliefs connected with the student's actions.

The inclusion of the student model built using BDI architecture (it is an internal MAS composed by the mental states represented as an agent) give more than just sensorial information for the tutor. It describes the current state of the student and at the same time did not labeled the student according previous stereotypes. The student is considering the way hi/her is. More details can be obtained in [2; 3; 6; 7; 8].

Future works will concern building cooperative environments on the Internet. Now we are migrating to cyberspace because it is a matter of survival. The distance education is a reality in our University (and it is spreading out all over the world). The demand of environments to support the teaching-learning activities on the Web is increasing a lot. Once more the agent's approach will bring us more tools and facilities to overcome and handle with many topics. Some of them are the possibility of modeling many agents that can work as student personal assistant, task agents to collect information for students and teachers, agents to help to organize the students interaction, and so on. We will face the same problems, on the Internet, like we have faced on traditional classes, and new ones will be found. As future works, we are also working on two aspects described on the beginning of section 2. We are formalizing a methodology to specify educational software in a MAS approach (Software Engineering viewpoint), and developing a Web application for C language classes using agents and the pedagogical knowledge that was acquired with our previous works.

Both works are related with our new projects involving three private universities of Rio Grande do Sul – Brazil.

Acknowledgments

I wish to say thanks to my students and colleagues from Faculty of Informatics of PUCRS that collaborated with me in different ways during the last ten years. They made possible to implement those ideas that I had in my mind. Special thanks to CNPq, CAPES, and FAPERGS to have sponsored all these works.

5. References

[1] Callegari, D. 2000. Applicant aprendizagem por reforço a um ambiente para o ensino de ecologia. Master Diss. Dept. of Computer Science, PUCRS.
[2] Giraffa, L.M.M; Vicari, R.M. 1998. Tutor behavior in a multi-agent ITS guided through mental states activities. In: Workshop of Pedagogical Agents. 49-54.ITS'98- Fourth

International Conference on Intelligent Tutoring Systems. AIED.

[3] Giraffa, L.M.M; Mora, M.; Vicari, R.M. 50-60 Modeling the MCOE Tutor using a Computational Model. In: Lectures Notes on Artificial Intelligence - SBIA'98. Berlin: Springer Verlag.

[4] Giraffa, L.M.M; Mora, M.; Vicari, R.M. 1999. Modeling an interactive ITS using a MAS approach: from design to pedagogical evaluation. 153-158 In:3rd International Conference on Computational Intelligence and Multimedia Applications- ICCIMA'99. Los Alamitos: IEEE Computer Society.

[5] Giraffa, L.M.M. Uma arquitetura de tutor utilizando estados mentais. Porto Alegre: CPGCC/UFRGS, 1999. (Ph.D. Thesis)

[6] Giraffa, L. M. M., Móra, M.C., Zamberlam, A.O.2001. Working on student models (really) based on mental states. 379-383.In: 4TH International Conference On Computational, 2001, Los Alamitos, IEEE Computer Society.

[7] Móra, M. C.2000. A model of executable agent. Ph.D. Diss. Dept. of Computer Science, UFRGS.

[8] Vicari, R.M; Giraffa, L.M.M. 2002.The Use of Multi Agent Systems to Build Intelligent Tutoring Systems In: International Journal of Computing Anticipatory Systems. The American Institute of Physics (AIP), V.1.