UNIVERSIDADE DE SÃO PAULO

PROTEMA
Intelligent tutoring systems for mathematics

MARIA DAS GRAÇAS VOLPE NUNES
RICARDO HASEGAWA

No. 20

N O T A S

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2577

PROTEMA
Intelligent tutoring systems for mathematics

MARIA DAS GRAÇAS VOLPE NUNES
RICARDO HASEGAWA

No. 20

NOTAS DO ICMSC
Série Computação

São Carlos
Nov./1995

PROTEMA
Intelligent Tutoring Systems for Mathematics[1]

Maria das Graças Volpe Nunes[2]
Ricardo Hasegawa
{mdgvnune, rh}@icmsc.sc.usp.br

Departamento de Ciências da Computação e Estatística
Instituto de Ciências Matemáticas de São Carlos
Universidade de São Paulo
C.P. 668 - 13560-970 - São Carlos - SP
Brazil

Abstract
This paper describes the project Protema. It proposes a general model for intelligent tutoring systems in the domain of Mathematics and presents, as results, the choice of a suitable representation model for mathematical knowledge, the proposal of a correspondent system architecture for intelligent tutoring systems (Arqtema), as well as an authoring environment (Tootema) for building Arqtema-based tutoring systems in any subdomain of Mathematics. The general representation model of mathematical knowledge maps any mathematical theory into a complex relationship among concepts, results and examples. Arqtema consists of an extension of this model which includes a bug catalog with the most common students misconceptions as well as a set of related exercises. The authoring tool provides help to the author during the task of creating and structuring the domain knowledge. The remaining system's components are composed with the knowledge bases which are part of a common kernel of subdomain independent modules. Tootema consists of a graphical and a document editor, a browser, a simulator and an assembler.

Resumo
Este trabalho descreve o projeto Protema (Projeto de Sistemas Tutores para Matemática). Propõe-se um modelo geral para sistemas tutores no domínio da Matemática. Para tanto, ele se utiliza de um modelo de representação do conhecimento matemático, baseado no modelo proposto por Michener, que mapeia toda teoria matemática numa complexa rede de relacionamentos entre conceitos, resultados e exemplos. A extensão proposta prevê, para efeitos tutoriais, as categorias de exercícios e de erros frequentemente cometidos por aprendizes da respectiva teoria. Tal modelo constitui, no sistema, o módulo do domínio. Os demais componentes do sistema são o modelo do estudante, o módulo tutorial e o módulo da interface. Por tratar genericamente o conhecimento matemático, propõe-se, ainda uma ferramenta de autoria de sistemas para diferentes subdomínios matemáticos, Tootema, descrita também nesse artigo.

Keywords: Intelligent Tutoring Systems, Authoring Systems, Mathematical Knowledge Representation.

---

[1] This work has been partially supported by a grant from the HyperProp-ProTeM project of CNPq-Brazil # 680063/94-3
[2] This research is partially supported by a research grant from CNPq-Brazil #301365/91-1

1. Introduction

Intelligent tutoring systems (ITS) are educational programms that incorporate knowledge about what they teach, whom they teach and how to teach. The contribution from Artificial Intelligence researches lies on the knowledge representation process: the ability of representing knowledge in an explicit and adequate way, as well as having the control mechanism independent from domain knowledge, allowing the system to reason about that knowledge. In the scenario of tutoring systems, this means to be able to treat users individually, as well to deal with particular details of different domains.

It has been argueded that the task of designing and implementing intelligent tutoring systems requires the cooperation of profissionals from different areas, such as computer science, cognitive science, education, student modelling, besides those from the particular domain to be explored. Considering tutoring systems to be used in environments where the teacher has an active role, such as during a session in the classroom, teacher's contributions should be taken into acount during the design process. What can be derived from the literature is that the expected real author of such systems -- a teacher -- has not been directly involved in the design process, and few of them happen to bring major contributions to the project. As a consequence, more and more the researches results are restricted to academic purposes and seem to be weakly related to pragmatic aspects from the teaching/learning process [9].

Another good reason for providing authoring tools for ITS is that more than 50% of the design effort is wasted during the domain knowledge codification. Moreover, many of the rules for a good knowledge base development can be verified during the creation process [1].

Methodologies for designing ITS are recent and there is no agreement about the best ITS architecture to be adopted. Moreover, different domains seem to require different structures. Nevertheless, some generic authoring tools can be found in the literature, such as SIIP [7], PIXIE [15], IDE [14], SCALD-1 [11], KAFITS [10], Meno-Tutor [18], CMU-Tutor [4]. However, the price for generality is usually paid with usability, and generic tools happen to be left apart by exigent authors. The process of knowledge acquisition and representation is a hard task even when the domain is known. In the case of generic environments, this task becomes much harder. Authoring systems for specific domains are usually less complex than their counterparts for generic ones and they can provide the user with effective tools for helping the task of structuring the knowledge.

This work presents Protema, an environment for accessing and/or building intelligent tutoring systems in Mathematics. The systems created via the authoring component, Tootema, share the domain knowledge representation schema, which is based on a general model for representing mathematical knowledge [9]. Tootema's users are supposed to create new domain knowledge bases, or to modify some existing ones, in order to obtain an ITS in a new subdomain. Moreover, users can add code which is particular to that subdomain, and even set parameters of some other knowledge bases. Most of the code modules, however, are inherited from a common nucleus. The tool was specially designed to help the user in the task of structuring large quantity of information by providing graphical representation and simulation mechanisms. As a consequence, the required resources and development time to structure the instructional content are reduced. In this way, the consistency and the qualitiy of the created learning activities are automatically improved. On the other hand, new relevant contributions can improve the environment bringing the teacher into the design scenario.

The mathematical knowledge representation model is discussed in section 2. Section 3 presents the general architecture of the Protema tutoring systems. The authoring tool is detailed in section 4, and section 5 concludes the work.

2. A general representation model for mathematical knowledge

Michener's representation model for informal mathematical knowledge consists of a network involving three fundamental cathegories: concepts, results and examples [9]. Each category is divided into epistemological classes in which are defined some intrinsic relations. Dual relations can also be defined among different categories.

The concepts category has at least three epistemological classes: definitions, mega-principles and counter-principles (Figure 1). The last two classes contain the heuristic advice that we use to give (to others and to ourselves) while working in a theory. Mega-principles have the form of suggestions or generically truthful heuristics. For example, *"symmetrical matrices are good"* is a mega-principle from Linear Algebra. Counter-principles, on the other hand, are advices in the form of cautions that indicate possible sources of troubles. *"Do not divide by zero"* is a well-kown counter-example. A pedagogical ordering, which establishes that concept A should be introduced before concept B, structures the concepts of this category. Sometimes this relation reflects the fact that concept A is part of the definition of concept B; other times this reflects preferences related to the expository order of the concepts. More than one vision could be explored by different presentation orders.

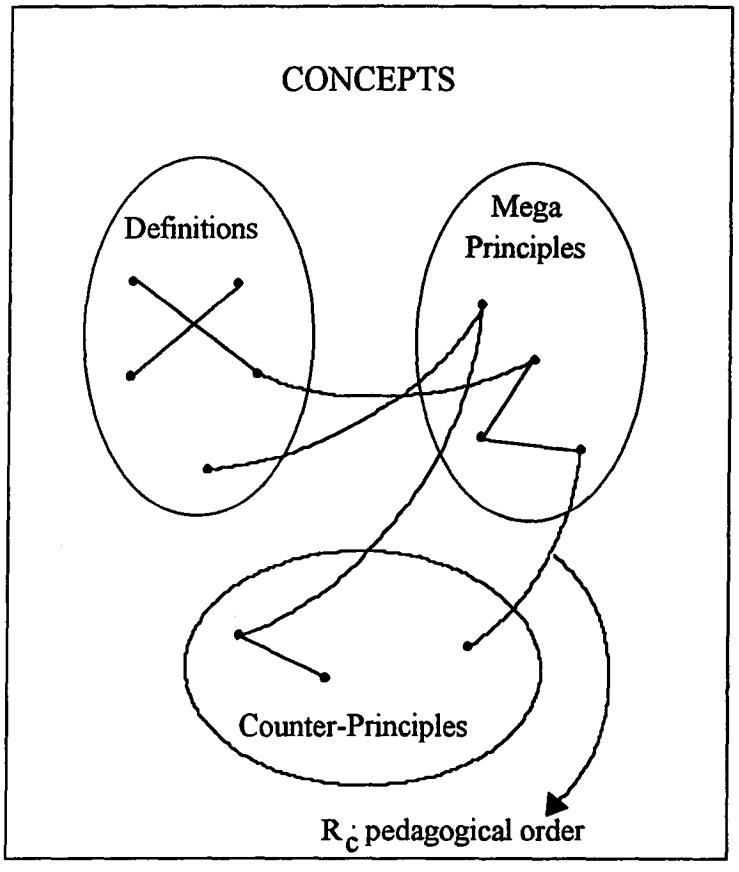

Figure 1 - The Concepts Category

The results are the traditional logical aspects of Mathematics: theorems and their proofs. The elements of this category are structured by the relation of logical deduction, what means that *A precedes B* indicates that result A is used to prove result B. The basic and culminating results are two of many subclasses of items that form the category of results. Basic results establish elementary properties of concepts. Culminating results are

those concepts towards which the theory has been driving to. The Fundamental Theorem of Algebra is an example of a culminating result in the study of polynomials.

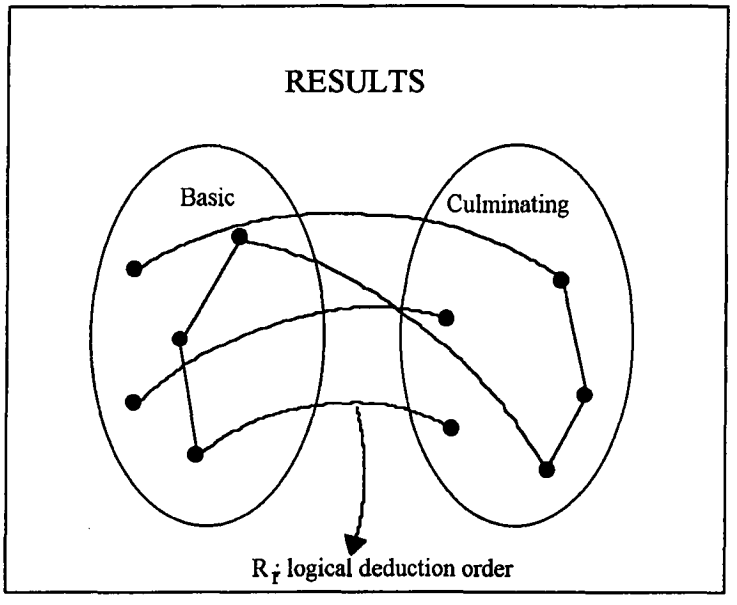

Figure 2 - The Results Category

The examples are structured by the relation of fabricational derivation in which an example A precedes an example B if A is used to construct B. For example, natural numbers N begets the integers Z, which begets the rationals Q, which begets the real numbers R, which begets the complex numbers C. Four epistemological classes can be detected in the category of examples. The start-up examples help one get started in a new subject by motivating the fundamental definitions and basic results. They can be understood on a stand-alone basis and are prospective in the sense that they can be easily generalized. The reference examples provide a common node through which many concepts and results are linked together. They are frequently refered during the development of a theory. For instance, the integers are always refered in number theory. The counter-examples are used to show that an assertion is not true, as well to show the limits of the properties of a theory.

Some analogies between the epistemological classes are possible: model examples, mega-principles and culminating results can be seen as the most important items of a theory; counter-examples and counter-principles are used to show the limits of a theory; basic results and start-up examples are usually initial points to introduce a theory. Moreover, some specific relations between items can occur in different spaces. We can define epistemological dual spaces by defining relations between the three category spaces. This way, each category item has two sets of dual items from the two other categories (Figure 4). The dual items of an example are the sets of concepts and results needed to discuss or fabricate it and, also, the sets of concepts and results motivated by it. The dual items of a result are the examples motivating it, concepts establishing its proof and also, the concepts and examples derived from it. The dual items of a concept are the examples motivating it, the results laying the groundwork for it and also, the examples illustrating it as well as the results proving things about it. Moreover, it is possible to define relations based on the dual idea: two items are related via dual idea if they share dual items. For instance, two items are conceptually (or illustratively, or deductively) dual if they share common concepts (or examples, or results).

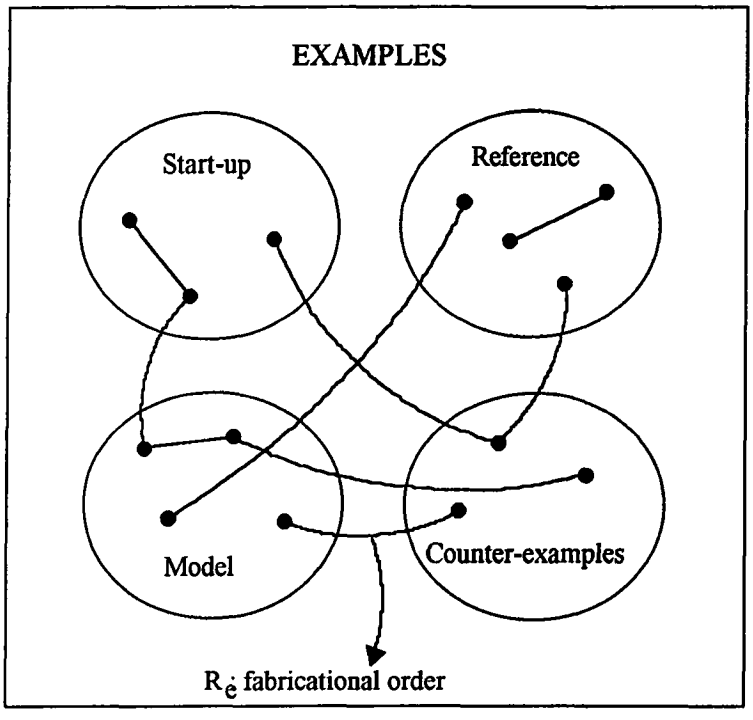

Figure 3 - The Examples Category

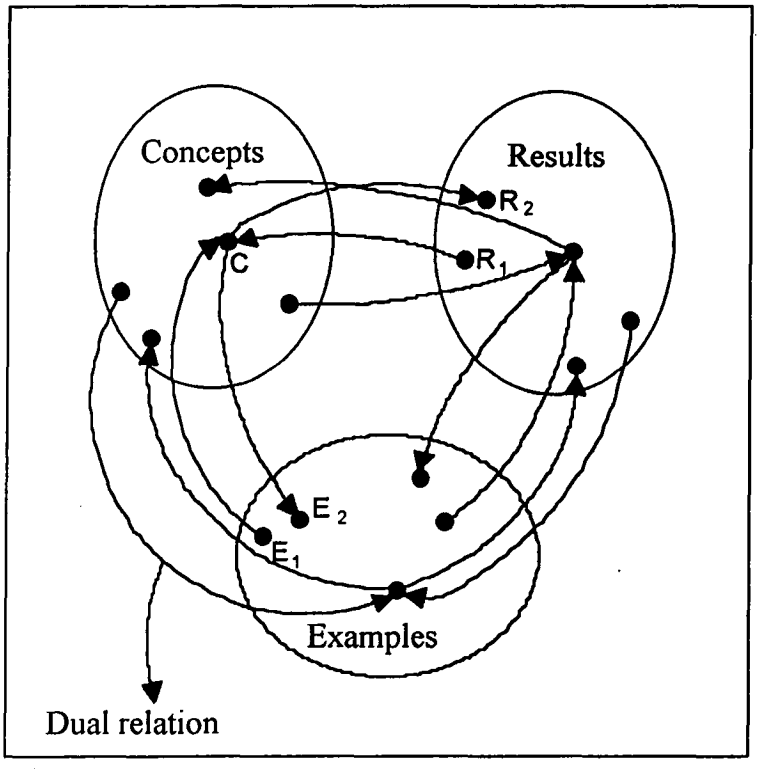

Figure 4 - The Dual Idea

The Michener's model is an interesting proposal for informally representing mathematical knowledge and it has been the basis of the intelligent tutoring system's architecture we present in the next section.

3. The tutoring system's architecture

The system's general architecture (Arqtema) consists of four knowledge bases containing the domain knowledge, a student model, the tutorial module and the interaction module (Intema), as shown in Figure 5. The tutorial module is the system's core and its role is to decide what, when and how to present the instructional material to that particular student, based on knowledge provided by the other three modules.

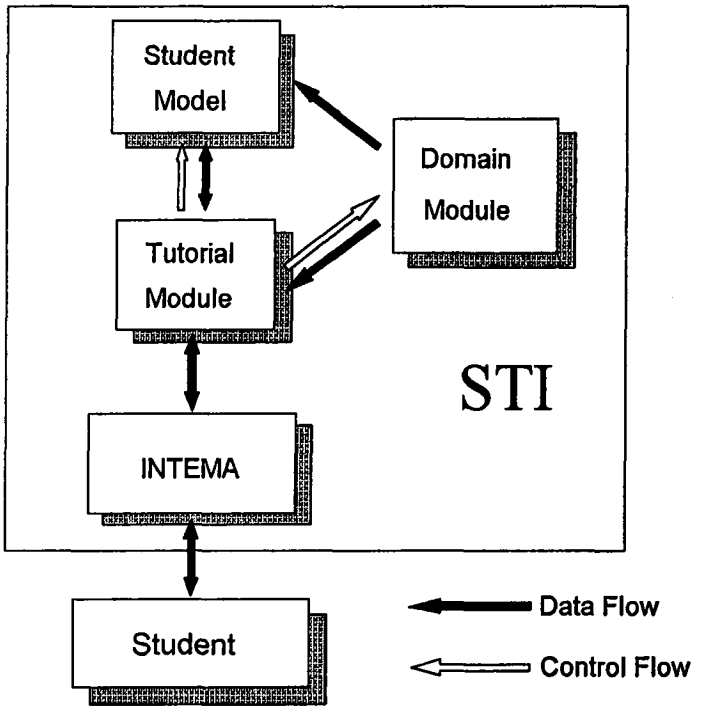

Figure 5 - The general architecture Arqtema

3.1. The Domain Module

The domain module is a two-layers structure (Figure 6). The topic layer consists of a network of topics about the subject matter. These topics are related by what we call *tutorial relations*. These relations are based on the suitable orders of presentation of the topics to the student under the author's perspective. They include relations based on information about prerequisites and complexity. For instance, in the domain of Logic, the presentation of First Order Language should take place before the presentation of well-formed-formulae. The topic layer is part of both tutorial and domain knowledge since topic links suggest not only a presentation order but also a conceptual link between topics of the subject. This meta-knowledge level allows for changing precedence relations without changing the domain knowledge.

The lowest layer is an extension of Michener's model. It consists of a network including concepts, results, examples, exercises and a catalog of most frequent bugs which occur during the resolution of problems in the theory. Each submodule is in turn a network, according to Michener's model. Exercises are associated to concepts and results, and they are aggregated into classes of similar resolution types. Each exercise is in fact a pattern whose parameters are set in order to instantiate an exercise item. For instance, different exercises on second grade equations can be derived from a pattern which comprises all information about this kind of exercise. Parameters like coeficients and alternative solutions are provided only once for the pattern.

Knowledge about bugs are connected to the exercises in order to allow the system to decide what to do after detecting a student's wrong answer. At the moment, prototypes present only multiple-choice exercises. This restricts the interaction but makes easier the strategy of what to do next: each alternative is associated to a possible bug, which is, in turn, related to a system's action. These actions can be a system's suggestion of which topic or exercise should be revisited or done, or even an explicit presentation of some concept or result, depending on the information from the student model.

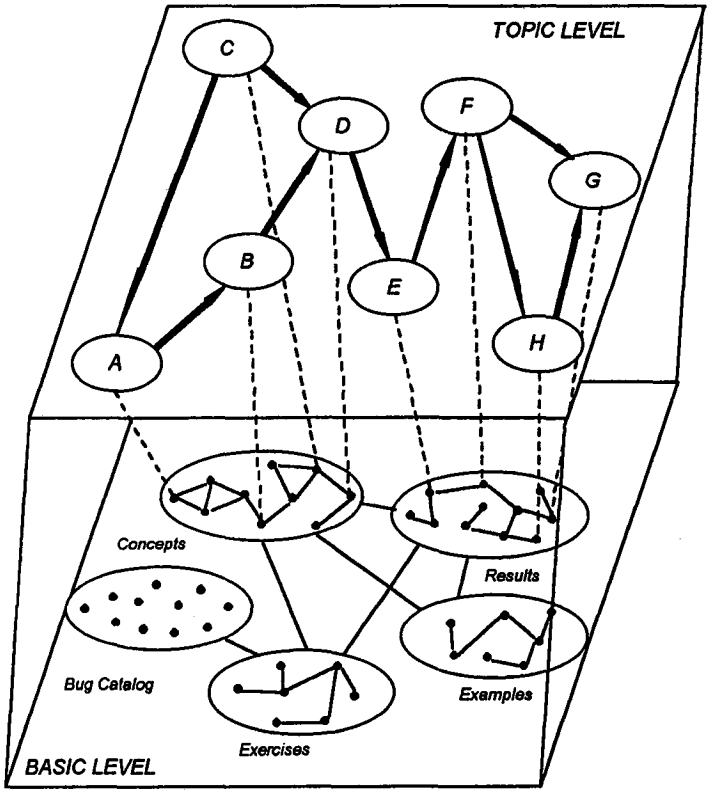

Figure 6 - The Domain Module

3.2. The Student Model

The key for the students individualization in a tutoring system is its knowledge about its users. The module that contains this knowledge is called student model. Most student models are of quantitative or numeric type and they express too little in terms of what a tutor should know. Qualitative models are potentially richer and are supposed to keep the processes used by students in the knowledge acquisition task, in terms of spatial, temporal and cause-effect relations [3]. However, it is far from simple to keep an accurate student model during the learning process. One strong reason is the well-known lack of knowledge about *how* the learning takes place.

In order to acquire some qualitative aspects of the user's knowledge and to keep control about what has been done in the past sessions, the Arqtema student model handles the following set of information: student identification; session number, topics covered per session, list of examples and information about exercises per topic; list of conceptual items already seen in the past sessions; list of conceptual items (concepts, exercises and examples) seen in the present session (there are different strategies for detected problems related to concepts which have been seen in the past rather than in the

present session); list of concepts assumed (by the system) to be known (by the student); list of bugs detected during evaluation process. All these parameters are represented as values of slots of a frame-based structure, and each one is associate to a procedure that indicates how this value can be obtained during the interaction with the student. The student model is updated by tutor's actions. Each student's action triggers a system's reaction which can change some values in the model. Some assumptions on the values of some parameters can be made by the system since the student can take the initiative in several points of the interaction (during the choice of topics, the choice of being or not evaluated, the need for help, etc.). However, this information is carefully handled by tutorial rules since the system can make a wrong assumption. They are considered as *weak* information.

The student model is of buggy type [2], which covers the cases in which the student's incorrect action does not result from the incomplete knowledge but from the incorrect versions of the student's target knowledge. Thus the student model indicates the remedial process which can take place after the diagnosis of an error or misconception. The diagnose process takes place during the exercises resolution: the predefined solutions for drilling exercises point to procedures which can suggest or even present some conceptual item (only those assumed as known have higher priority for being reviewed). Those items already seen are presented in alternative forms by the system. Figure 7 illustrates an instance of a student model.

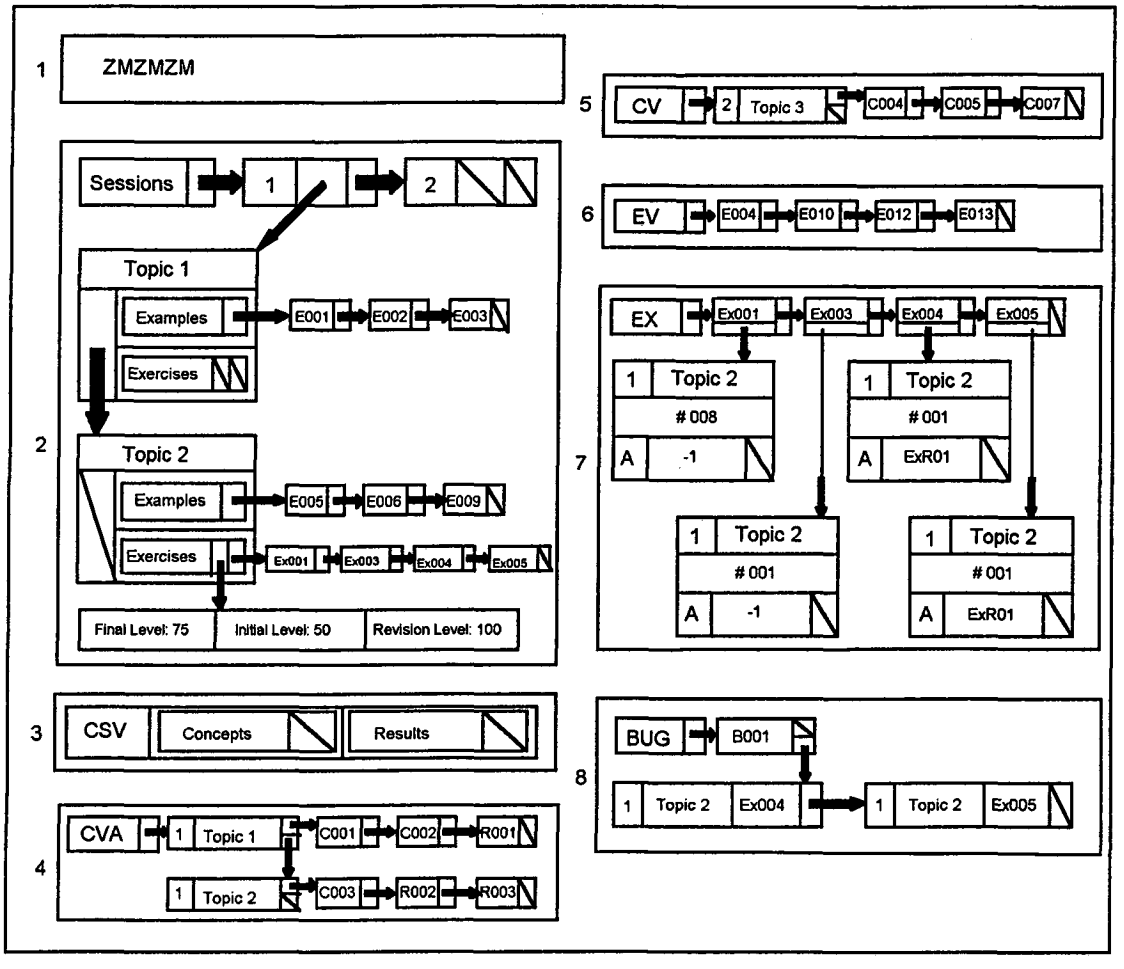

Figure 7 - An example of student model

3.3. The Tutorial Module

The tutorial module consists of a structured production rules system [13] which associates information from domain module to information from the student model. According to the present context, rules are triggered to decide the next system's action. The result of the application of a tutorial rule includes a set of actions related to the system's tasks -- the presentation of instructional material, of an exercise, of helping information, etc. -- and also the updating of the student model.

Rules are classified into categories which indicate what is the actual status of the dialogue with the student. These categories include the phases of initialization, presentation, exercises, questions, etc. The choice is determined by a controlled graphic interface based on buttons and menus. Some of the rules selected in one phase set the next set of rules to be considered in the following phase. This way, the search for the rules to be triggered is limited to a subset of the system.

The following procedure summarizes the tutorial actions. After the student's identification, the system recover the student model (or initialize a new one) and determines what is the suggested sequence of topics for that user. In fact, this sequence is a subgraph of the topics network, which is built by finding a topological order in that network from the start point selected by the student [16]. The system presents the concepts related to the topics as well as the associate examples and exercises selected by the user (Fig.8). The system can evaluate the student if some exercises are done. The student model is always updated by the tutorial actions

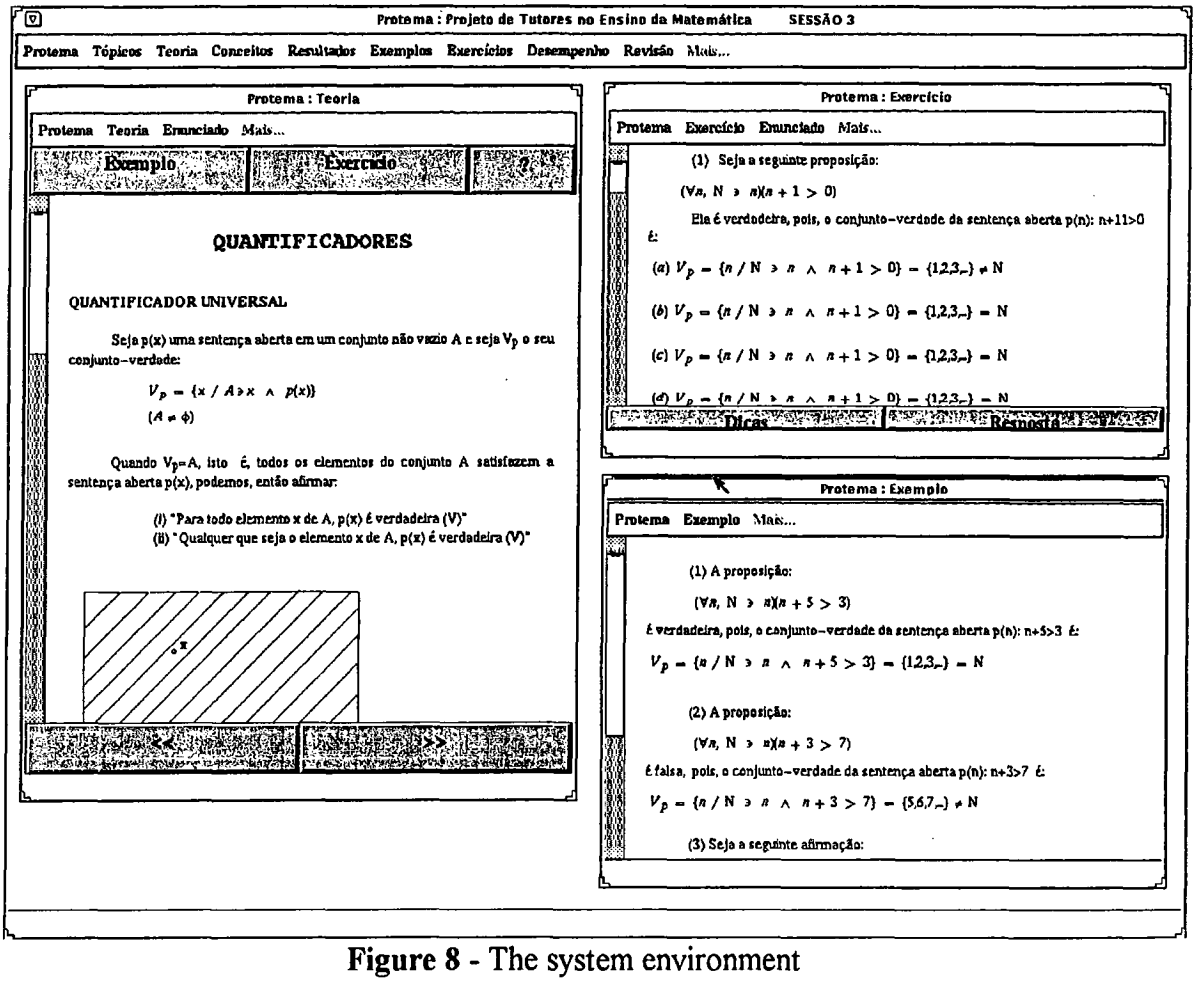

Figure 8 - The system environment

Tutorial actions involve system's decisions about what and how to present next. This is done based on the students performance in the proposed exercises and from the relations found in domain module. Therefore, complementary theory items (concepts definitions, examples, solved exercises) which are related to the main topics of the subgraph initially built can be selected for presentation to help the student in the learning process. For example, if the student has demonstrated dificulty in solving an exercise, the system can strategically decide to illustrate a related concept. Moreover, depending on the students status, some presentation characteristics, such as the level of detailing, are also considered by tutorial actions. For these cases, the system author has to define the item content according to a simple grammar, since the goal is to provide domain independent tools.

3.4. The Interaction Module

Intema is the communication module of Arqtema. It is responsible for the interaction of the users (student and teacher) with the system; its major function is to provide explanations to the users. The typical user of an ITS system is a student who is dealing with concepts not very well understood, what makes the interaction a very complex task. Moreover, explanations about certain ways of reasoning constitute by itself a way of teaching. These facts make us believe that explanation is extremely important and decisive in helping the student to understand the system, to elaborate domain's concepts and to (re)build her/his knowledge. Additionally, the more the explation takes into account the user's profile, the more effective it can be.

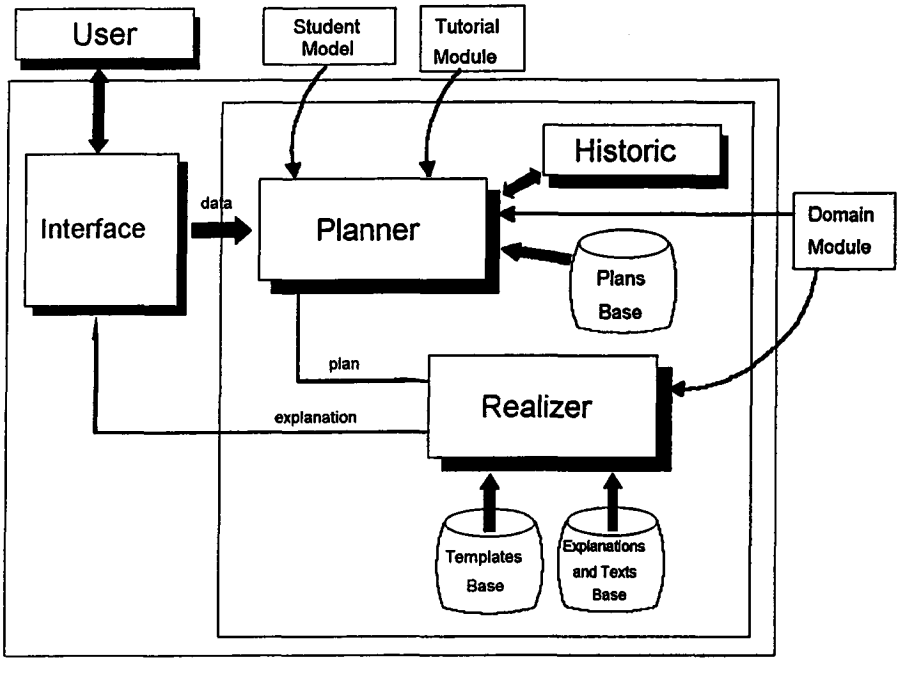

Figure 9 - INTEMA's architecture

Intema's architecture (Figure 9) is composed by a planner, a realizer and an interface module. The planner builds a plan by instantiating a selected plan with information from the domain module. The selection is based on information about the student, about what has already been explained, and about the context of the session (set by the menu option selected). The realizer module, based on information obtained from the plan,' composes

the final text with templates and chunks of predefined texts. Figure 10 illustrates an explanation provided by the system after a student's mistake.

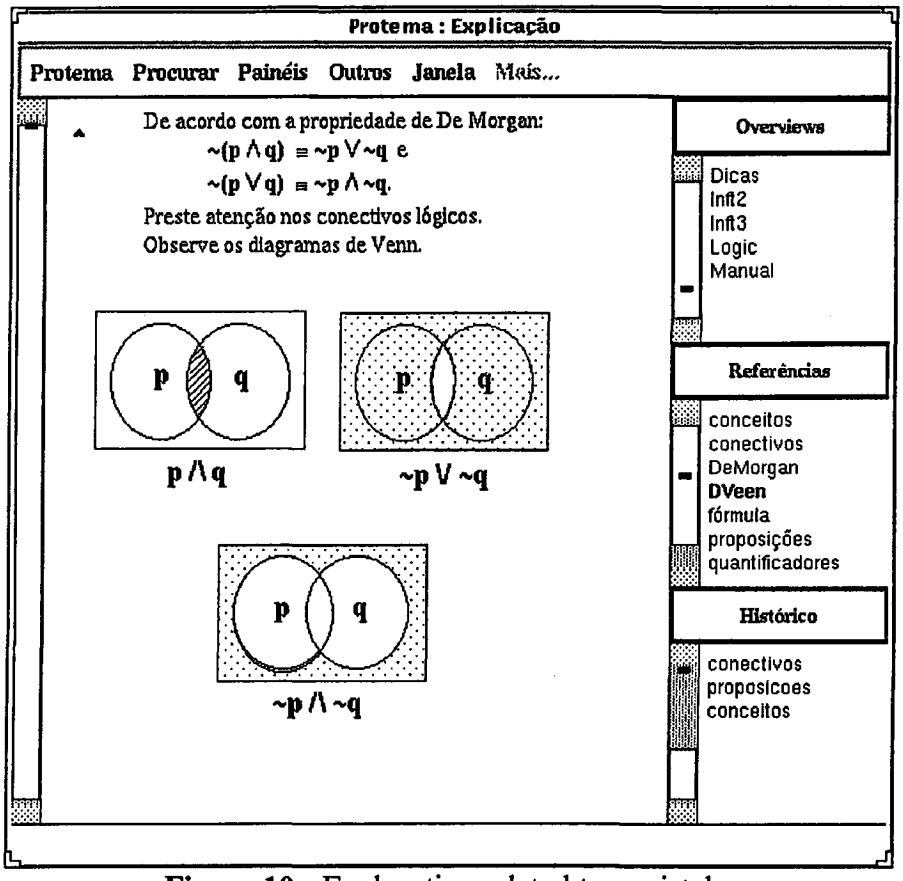

Figure 10 - Explanation related to a mistake

Intema also provides interaction with the teacher who can access iformation about the student's (individual ou comparative) performance, as shown in Figure 11.

The first system prototype was developed for a Sun workstation, using Andrew Toolkit for Unix [4] and language C. The Andrew's power is its capacity of creating multimedia documents via special editors. The interface elements includes windows, menus, buttons and hypertext-based help functions, among other resources of a graphic interface.

More details about Arqtema can be found in [12], [16] and [8]. The following section introduces Tootema, the authoring tool for building Arqtema-based intelligent tutoring systems.

4. The Authoring Tool

An authoring tool for ITS is similar to a shell for expert systems. It should provide a generic system kernel, through which the user defines the knowledge bases for a new domain. The resulting system is an ITS for the domain structured by the user. Moreover, the author can produce a new ITS from an existing one through mechanisms of domain exchange. The main requisites for an authoring tool can be summarized as:

- Modularity to avoid highly coupled components and spread code.
- Usability and Flexibility to make easier the creation task for teachers.
- Explicit knowledge representation to facilitate alteration of knowledge bases, and to provide reasoning on the knowledge.

- Parameterized procedural knowledge to adequate the alterations made, without accessing system's details.

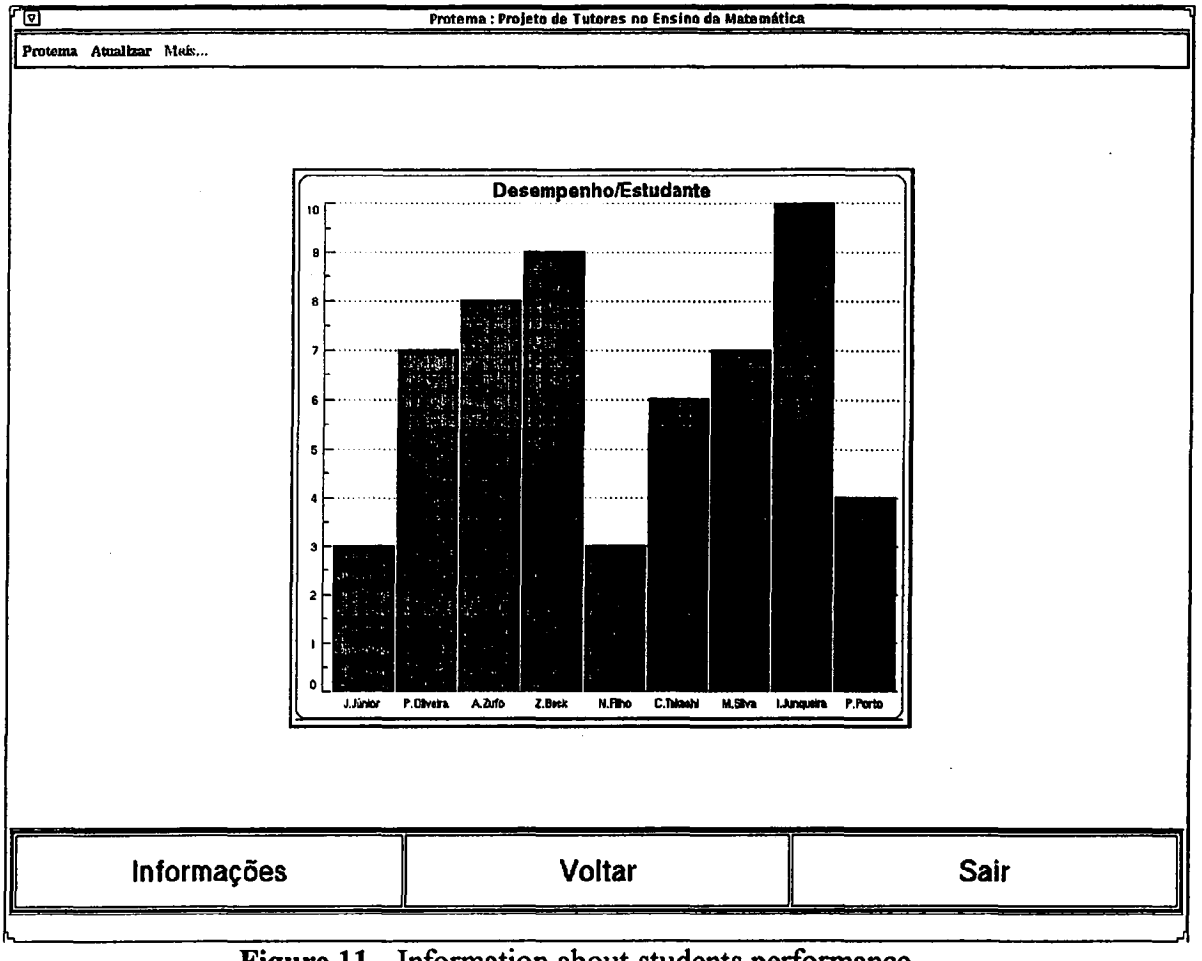

Figure 11 - Information about students performance

Designing and structuring instructional material, especially for extensive or sophisticated courses, constitutes a complex information management task. The author (usually a teacher) must determine what content should be covered in a course, usually decide how it should be presented, and produce the course material. Based on the human process of creating instructional material, a generic authoring model consists of the following stages [6]:

- Creating and collecting material. This involves the identification of main concepts to be taught; the major topics to be covered; the classification of the material by topic. References are identified and preliminary course material are cerated. Examples of products of this stage include concept definitions, set of examples, explanations, illustrations, etc.
- Contextualising the subject matter. Once the main subject matter has been identified, the material may be related to other subjects the students have studied or will study. The separation of this process from the previous one enables the adaptation of the material to different types and levels of students.
- Identifying learning activities and tools needed. This step involves designing a set of exercises or activities for students to be linked with topics in the course material. Examples include questions to be answered and problems to be solved. Learning

activities can be organized by topic and can be ordered by factors such as difficulty level.

- Identifying characteristics of the students. This identification can dettermine different styles of material presentation and different types of explanation, among other things.
- Identifying the assessment methods. The means by which the course is to be assessed are determined in this stage. The formal assessment may contribute to a student's overall grade, and the informal assessment may be used to provide feedback to the teacher on the student's performance. This can be achieved by examining the results of student's activities, enabling the teacher to adapt, for specific or a whole group of students, the structure, the pace or the approach taken to teaching a topic.
- Presenting the topic matter. At this stage, the material is organised into a related set of lessons. Each lesson will consist of one or more learning activities. This organisation may vary according to student characteristics which become evident as the course is taught.
- Refining the project. This stage involves the efficiency evaluation of the knowledge bases and the modification of inefficient portions. This process includes conventional tests as well as interviews with potential users.

Each stage described above specifies a set of tasks for the teacher. Some of these tasks can take advantage from the use of the computer as a support tool. We believe the tool we are to present here can support many of these tasks.

Tootema is a system for helping the author (namely, a teacher) in the task of creating and managing an Arqtema knowledge base. By knowledge base we mean not only the domain module but also the tutorial module and the student model. The tool provides resources for structuring large amounts of information, representing graphically the structure of the base, accessing and editing all domain elements in a multimedia environment [5]. Our goal is to bring the teacher into the development process of systems used in teaching-learning enviroments.

The tool is composed of five main components (Fig. 12): Interfaces with the teacher or the programmer, graphical editor, documents editor, browser, simulator and assembler, which are presented in the following sections.

4.1 The Teacher User Interface

Four main goals were considered for the building of the tool:

1. To allow the development of the knowledge bases in an easy and functional environment. Multimedia editors and graphical editors can offer a basic and direct way to insert in and modify knowledge from the Domain Module, independently from the way this knowledge is internally represented. Tootema's editors enable the teacher to move quickly through the stages of edition and tests, while building part of the knowledge base.
2. To prevent the teacher to add inconsistent information into the knowledge base. Tootema uses a controled menu-oriented interface for data acquisition.
3. To allow the visualisation of the knowledge base structure. This is an important requisite in the task of building the domain module. The user is supposed to want to see the final presentation sequence. Moreover, the graph enables the teacher to select directly the components of the knowledge network. Tootema offers a graphic editor for visualisation of the knowledge network and a simulator which allows the teacher to see how a topic will be presented to an user from a particular class.

4. To help the user to solve problems with the tool. Tootema has an on-line tutorial that helps the teacher with information about all components of the tool (Fig. 13). It shows, through examples, how to specify an information to be added into the knowledge base. Moreover, it provides mechanisms to locate and to consistently add information.

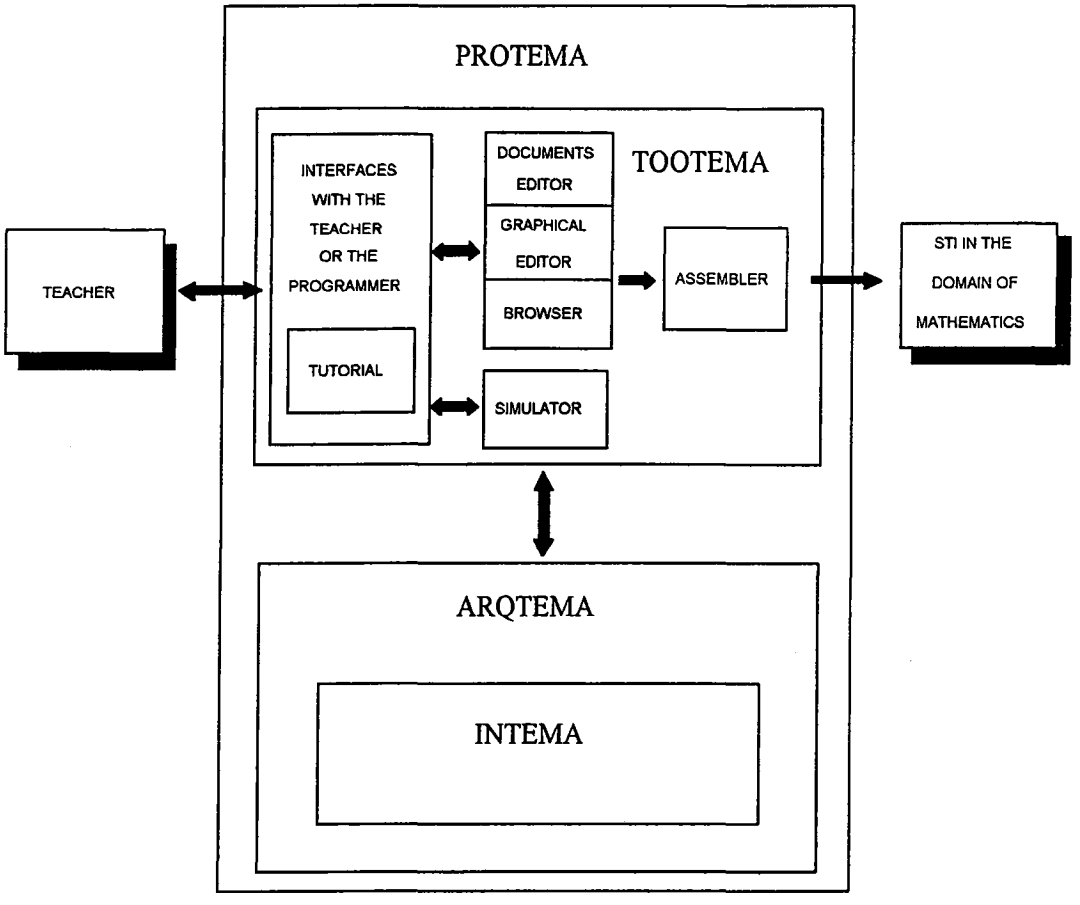

Figure 12 - Components of TOOTEMA

4.2. The Programmer User Interface

Tootema provides an interface designed specially for users who can programm in language C. The goal is to allow the user to build an interface for a new application by using common objects from the existing library. The user selects and groups interface components using Arbcon, a tool from Andrew Toolkit, and then the C source code of the prototype and the specification file are generate. The main goal is to allow the user to enhance the Protema environment with different applications.

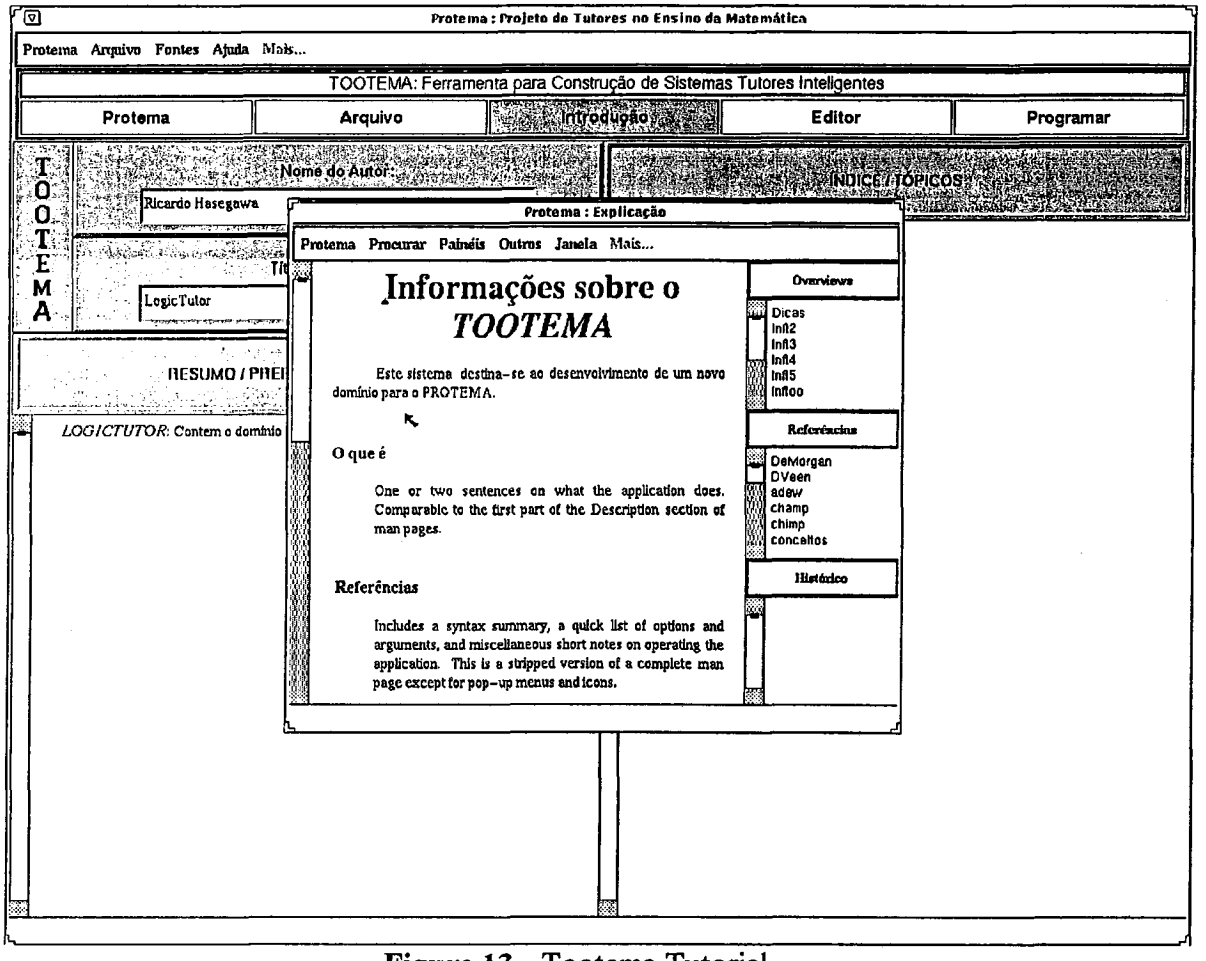

Figure 13 - Tootema Tutorial

4.3. Editors and Browser

For tasks of locating, adding, modifying and visualising information from the knowledge base, Tootema provides some edition components for helping the user to solve these problems. The user can add and recuperate information from knowledge base by two different ways: 1) by editing or consulting directly the knowledge base, through a mechanism which enables the expression of the information characteristics, or 2) by navigating through the base structure, looking for information not easily described. These two methods are not excludent; rather they can be used in a complementary way. The components which are responsible for these tasks are the editors and the browser.

The browser enables the user to search, create, edit, visualise and delete items information, and can be activated by clicking an item from de items list (right down window in Figure 14), or an element from the graphical network (left window in Figure 14), or even selecting one of those functions from the menu options. If the user selects the item via graphical structure, the corresponding contents is presented in the document editor for eventual alterations.

There are two editors in Tootema: the document editor and the graphical editor. The multimedia document editor allows the edition of knowledge base nodes, and also enables the creation of the initial knowledge base structure. The graphical editor enables the user to visualise the knowledge base structure.

The author can first list, via the document editor, the broadest topics in a traditional sequential way, and the system automatically creates the associated nodes which, in turn, can be successively refined to achieve a hierarchical knowledge structure. The access to

the nodes contents can be done by the browser, by the document editor or by the graphical editor. So, if the user chooses to select the node by clicking the corresponding object in the graphical structure, the document editor will present the corresponding node content, and the browser will highlight the node in its class (Figure 14).

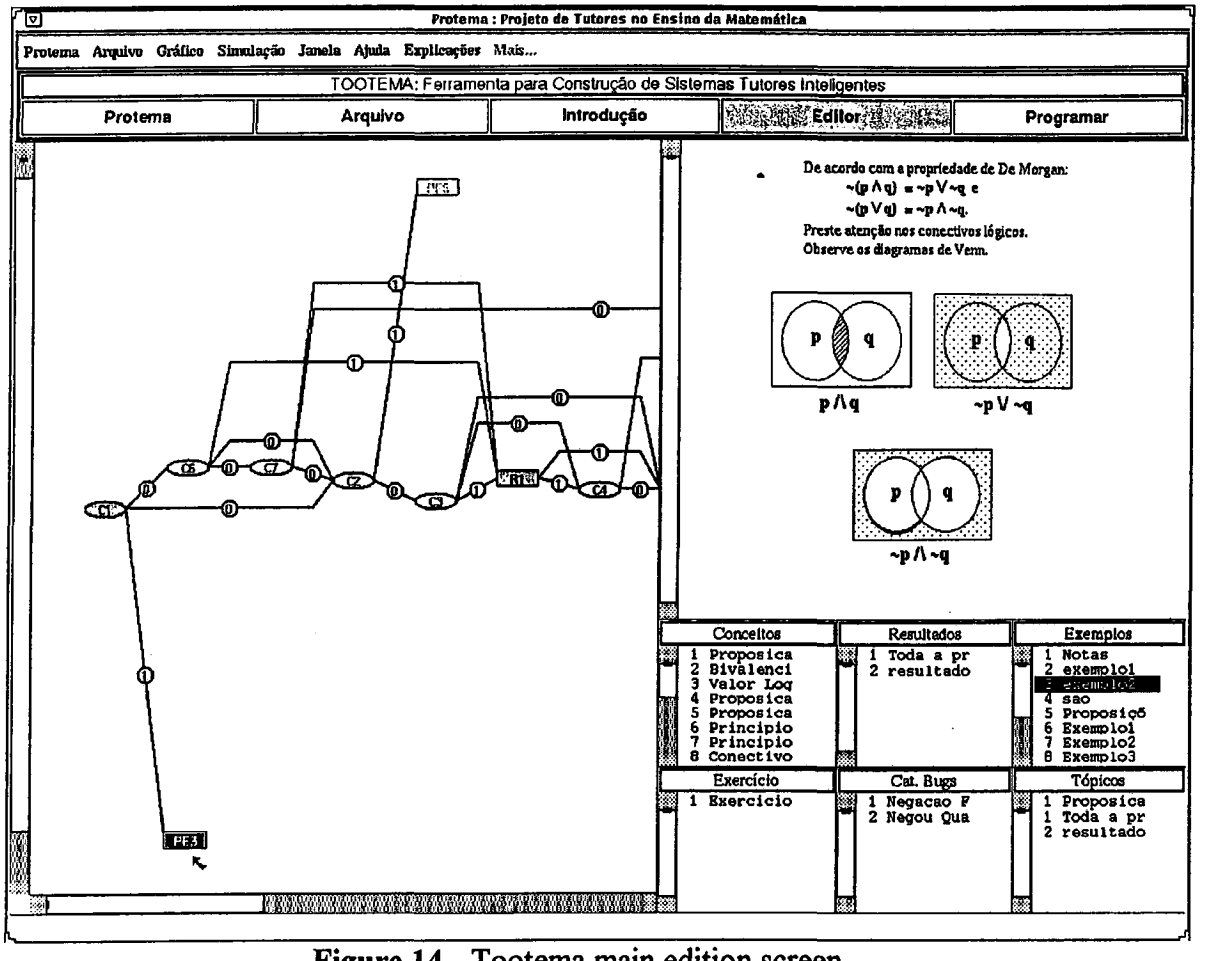

Figure 14 - Tootema main edition screen

The graphical editor also takes into account some aesthetics aspects such as the minimalisation of the edges crossing and of the area occupied by the drawing. Rather than redefines the nodes positions as long as more space is required, the algorithm used, through the the concept of node scope, directly determines the new positions of the nodes, without repetition of calculation [17]. Consistency tests are also made, such as whether the node just inserted by the author has provoked a cycle. Most of the relations defined by Michener's model defines partial sets, then acyclic graphs.

4.4. The Simulator

The simulator's task is to show to the author how a topic will be presented to the student. In spite of the importance of this function, it was not found in any authoring system from the consulted literature. In Tootema, the author selects a topic and sets some parameters in the student model, and the simulator presents the network structure resulted from the automatic process for preparing a session. By interacting with this system, the author can antecipate its behavior with the aim of detecting and editing the contents of the modules components. These components include the theory items in the domain module, some parameters in the student model and the tutorial rules in the

tutorial module. For the last two modules, some help is still necessary for the task of code alteration. Figure 15 illustrates the use of the simulator.

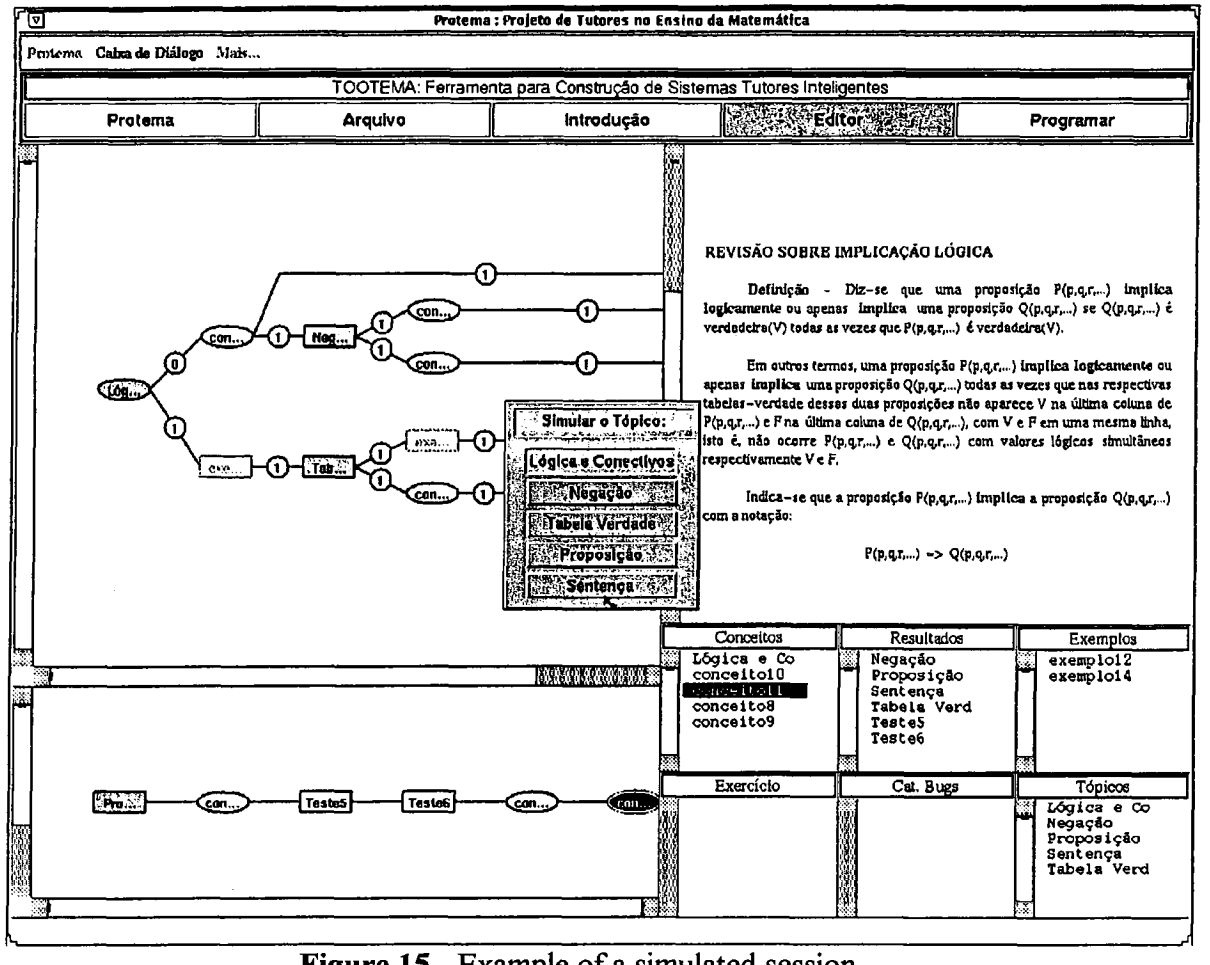

Figure 15 - Example of a simulated session

4.5. The Assembler

The assembler's main task is to assembly the reusable procedures and the elements from the knowledge base which will compose the resulting system for a specific domain. The student model as well as any domain specific program should be also added by the assembler. This tool also enables de user to create particular features for interfaces of the resulting system by using functionf from the Arbcon library, a module of Andrew Toolkit. For example, a calculation machine can be designed as a node of the knowledge base, as shown in Figure 16.

5. Conclusions

In this paper we have described a framework for the building of ITSs for Mathematics. It consists of a system architecture for intelligent tutoring systems in Mathematics domain which is based on a general mechanism for representing and manipulating the domain elements. The paper also describes the design of an authoring tool which uses the basic architecture to construct a tutoring system for a given mathematical theory. The current prototype of the authoring environment has been used by Mathematics teachers who give us the desired feedback for improvements. The

authoring process is complex and takes long time, but the resulting systems have usually overcome the initial expectations. The mathematical knowledge representation model has been proven adequate for it provides a familiar authoring process for most of Mathematics teachers.

Further work will be concentrated on interface issues such as more flexible ways for the dialogue with the student. One such way involves the use of some usual schema of reasoning presentation, since most exercises in Mathematics follow a common pattern. Based on this pattern, the system could also infer valuable new information about the student, therefore enhancing the student model. Moreover, additional tutorial rules could be formulate to handle this new information.

Ongoing tests on the use of the authoring tool by teachers as well as of the resulting systems by the students have demonstrated that simple approaches, such as the one described in this paper, can produce positive results.

The Andrew toolkit and language C for Unix consituted the basis of the implementation. Recent efforts have been made for the migration to Windows plataform in order to make the tool avaiable for more people.

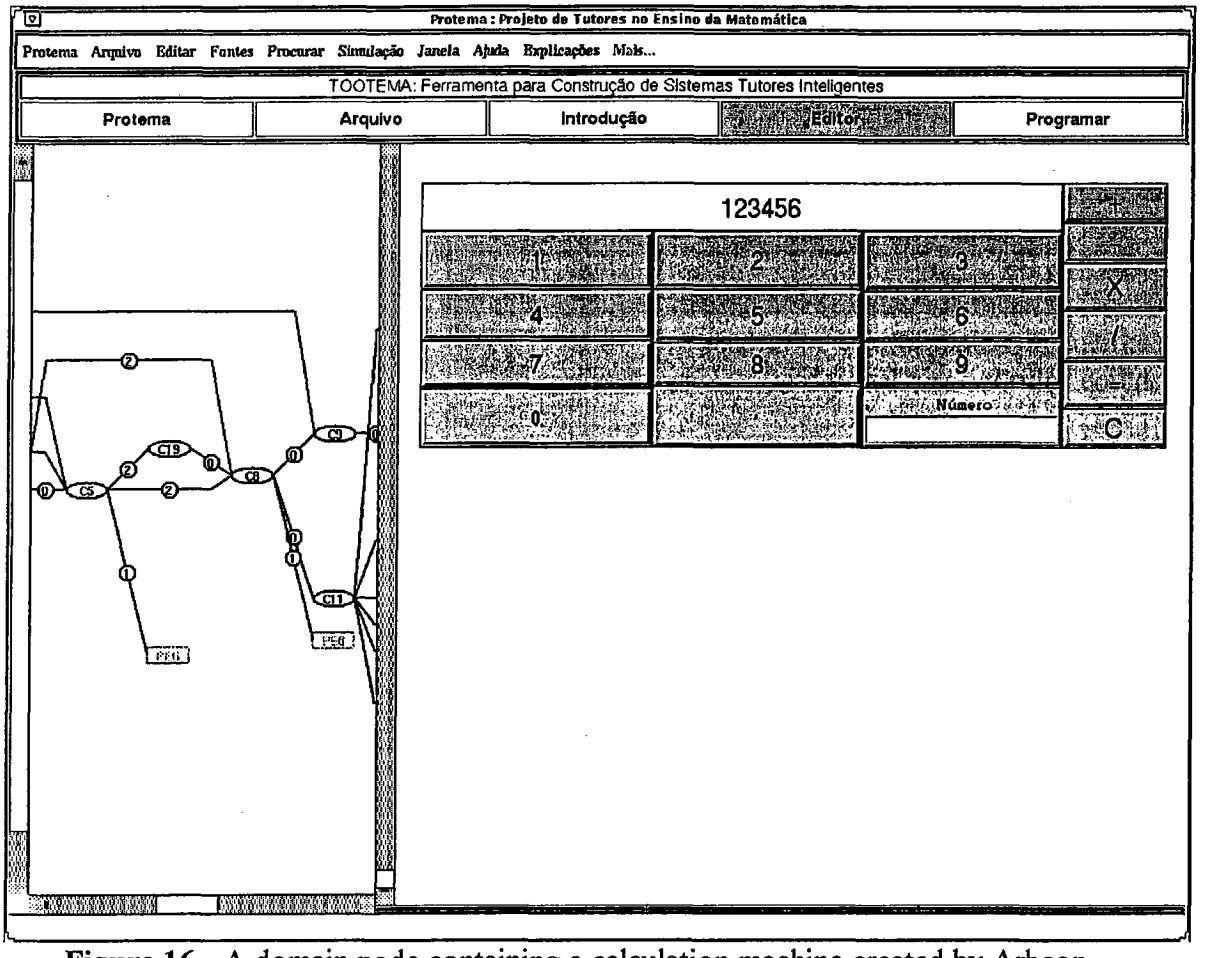

Figure 16 - A domain node containing a calculation machine created by Arbcon

Acknowledgements

We would like to thank Rosana Takehara and Marcus Vinicius for helpful discussions and comments on the authoring tool, beyond their valuable contribution to the ARQTEMA and INTEMA projects.

References

[1] Anderson, J.R. *The Expert Module.* In Polson, M.C. & Richardson, J.J. eds. *Foundations of Intelligent Tutoring Systems*, London, Lwrence Erbaum, 1988, pp. 21-53.

[2] Brown, J.S.; Burton, R.E. *Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science*, 2, 1978, 155-192.

[3] Clansey, W.J. *The role of qualitative models in instruction.* In J. Self (Ed.) *Artificial Intelligence and Human Learning.* Chapman and Hall Computing, 1988, pp. 49-68.

[4] Hansen, W.J. *The Andrew Environment for Development of Educational Computing.* Computers and Education, vol. 12, nro. 1, pp. 231-239, 1988.

[5] Hasegawa, R.; Nunes, M.G.V. (1995) *TOOTEMA: a Tool for building Intelligent Tutoring Systems for Mathematics*. Proceedings of the VI Brazilian Symposium on Informatics in Education, Florianópolis, SC.

[6] Hume, T. *Issues in the Design of An Authoring System for Educational Hypermedia Applications.* Proceedings of Seventh International PEG Conference - AI Tools and the Classroom: Theory into Practice. Edimburgh, UK, 1993.

[7] Macmillan, S.A.; Emme, D.; Berkowitz, M. *Instructional Planners: Lessons Learned.* In Psotka, J.; Massey, L.D.; Mutter, S.A. eds. *Intelligent Tutoring Systems - Lessons Learned,* New Jersey: Lawrence Erlbaum, 1988, pp. 229-256.

[8] Maltempi, M.V.; Nunes, M.G.V. *INTEMA: an explanation generator for Intelligent Tutoring Systems.* Proceedings of the XI Brazilian Symposium on Aartificial Intelligence, Fortaleza, CE, October, 1994, pp. 457-469.

[9] Michener, E.) Understanding understanding Mathematics. *Cognitive Science,* 2(4).

[10] Murray, T. & Woolf, B.P. *A Knowledge Acquisition Tool for Intelligent Computer Tutors.* SIGART Bulletin, vol.2, nro. 2, pp. 9-21, 1991.

[11] Nicolson, R. *SCALD - Towards an Intelligent Authoring System.* In Self, J. ed. *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction,* London: Chapman & Hall, 1988, pp. 236-254.

[12] Nunes, M.G.V.; Takehara, R.S.; Mendes, M.D.C. *A Network-based model for Intelligent Tutoring Systems.* X Simpósio Brasileiro de Inteligência Artificial, Porto Alegre, Brasil, oct. 1993, pp. 277-288.

[13] Rich, E. Knight, K. *Artificial Intelligence*. McGraw Hill, 1993.

[14] Russel, D.M.; Moran, T.P.; Jordan, D.S. *The Instructional-Design Environment.* In Psotka, J.; Massey, L.D.; Mutter, S.A. eds. *Intelligent Tutoring Systems - Lessons Learned,* New Jersey: Lawrence Erlbaum, 1988, pp. 203-228.

[15] Sleeman, D. *PIXIE: A Shell for Develping Intelligent Tutoring Systems.* In Lawer, R.W. & Yazdan, M. eds. *Artificial Intelligence and Education.*, vol.1. Ablex Publishing, 1987, pp. 239-263.

[16] Takehara, R.S. (in Portuguese) *ARQTEMA: um modelo genérico para Sistemas Tutores Inteligentes em Matemática.* MSc. Dissertation, ICMSC-USP, 1995, 109p.

[17] Vilela, P.R.S. (in portuguese) *Uma ferramenta para o auxílio visual ao teste e depuração de programas.* Msc Dissertation, FEE-UNICAMP, Brasil, 1995, 92p.

[18] Woolf, B. & McDonald, D.D. *Building a Computer Tutor: Design Issues.* IEEE Transactions on Pattern Analysis and Machine Intelligence, September, 1984, pp. 61-72.

NOTAS DO ICMSC

SÉRIE COMPUTAÇÃO

019/95 OLIVEIRA, M.C.; TURINE, M.A.S.; MASIERO, P.C. - A statechart - based model for hypertext.

018/95 PIMENTEL, M.G.C. - Alternative operations for browsing hypertext.

017/94 ROMEIRO, N.M.L.; CASTELO FILHO, A. - Análise Comparativa de Métodos Numéricos de equações algebrico-diferenciais.

016/94 MAGALHÃES, A.L.C.C.; SIQUEIRA, M.F.; OLIVEIRA, M.C.F. - Operadores de Euler na modelagem por fronteira: conceito, aplicação, estudos de casos.

015/94 ODA, C.S.; MOREIRA, E.S. - ASNMP graphical network monitor with automatic topology discovery.

014/94 FELIPE, L.S.G.; FRANCO, N.M.B. - Sobre a ordem de convergência para as equações integrais de volterra de segunda espécie tipo Abel com soluções não suaves.

013/94 PIMENTEL, M.G.C. - A framework for user-hypertext interaction.

012/94 TURINE, M.A.S.; MENDES, M.D.C.; NUNES, M.G.V. - TEGRAM: a geometry tutoring system based on Tangram.

011/94 SPOLON, R.; SPOLON, R.; SANTANA, M.J.; SANTANA, R.H.C. - Desenvolvimento de um gerador de aplicação para simulação de sistemas discretos.

010/94 SAWAKI, J.; MONARD, M.C.; RODRIGUES, S.R. SABNAG: - um sistema baseado em conhecimento para suporte aos usuários da biblioteca NAG.