
Methodological Aspects of Knowledge Types for Intelligent Tutoring System Design

Ljubomir Jerinic

Institute of Mathematics, University of Novi Sad,
Trg Dositeja Obradovica 4, 21000 Novi Sad, Yugoslavia
jerinic@uns.ns.ac.yu

Abstract. The advancement of using the Artificial Intelligence (AI) methods and techniques in design the Intelligent Tutoring Systems (ITSs) makes the understanding them more difficult, so that the teachers are less and less prepared to accept these systems. As a result, the gap between the researchers in the field of ITSs and the educational community is constantly widening. While ITSs are becoming more common and proving to be increasingly effective, each one must still be built from scratch at a significant cost. Also the present ITSs need quite big development environments, huge computing resources and, in consequence, are expensive and hardly portable to personal computers. This paper describes our efforts toward developing uniform data and control structures that can be used by a wide circle of authors, e.g., domain experts, teachers, curriculum developers, etc., who are involved in the building of ITSs, i.e. the model of the ITSs framework, the GET-BITS model.

Introduction

Since the early ‘70s when instructional computer programs such as Smallwood's geometry program, Carbonell’s SCHOLAR for geography, Kimball's Integration Program, Leeds Arithmetic (Woods, et al.) for arithmetic, etc. were developed; there has been a steady increase in the use of computers to deliver instruction, i.e. to teach. An indication of computer based educational trends is evidenced by the rapid growth in the number of computers available to pupils/students in elementary, high and university schools over last three decades in almost all countries. During that period, very rapid expansion in research and development of the computer based educational programs and systems are also evident and equally dramatic.

The cost of education and training is significant. For example, according to the U. S. Department of Education Projections to 2005, the cost of primary through secondary education is expected to reach $232 billion dollars by the turn of the century. It has also been estimated that in the U. S., $47 billion dollars is being spent
annually on corporate training and $20 billion dollars spent annually for military training. The similar situation is in nearly all countries, but with the different figures. Therefore, anything that will reduce the cost of instruction per student or trainee can pay for itself within a reasonable period. If properly designed computer based education has the potential to do just this by reducing instruction time and producing better-educated and trained students.

In the past 30 years we were witnesses of the great number of research projects all over the world to do above task. But, we are also the witnesses that a few of the results of that project, i.e. the educational software, are in the classrooms. The question is why, and how to surpass that?

**Problem Statement – the Good Educational Software**

The problems in the realization, implementation and acceptance of the computers in the teaching process arise from the origin of the problem:

*What we need for the good computer program in educational purpose?*

![Diagram showing the crossroads of science - the good educational software (GES)](image)

**Figure 1.** The crossroads of science – the good educational software (GES)

As it is shown in the Figure 1, the computer programs in education need to have the collected and embedded knowledge from the artificial intelligence (AI), the computer science (CS) and various teachers’ sciences (TS). The solution could be Intelligent Tutoring systems (ITSSs) and the numerous tools, i.e. ITSs tools for the
designing the GES. The lot of papers, words and ideas have been written and spoken in the last three years on:

- congresses, tutorials, workshops, formal and informal discussions on AIED’95, ITS ’96, AIED ’97, PEG ’97, IFIP Working Groups meetings, IFIP World Congress ’98 and ITS ’98), and in
- magazines and journals which directly or indirectly covers the GES,

between the researches in the field of artificial intelligence in education (AIED), pedagogical groups (IFIP, PEG, etc.) and others groups who are interested in the realization the GES. There are more of general accepted research directions in the next period, and some of them are:

- Introduce the computers in the real classroom, i.e. to use the computers in education;
- Reduce the cost of designing the educational software;
- Find the way to include the teachers community in the process of designing and implementing the educational software;
- Work on standard vocabulary and ontology;
- Investigate of the new theory of learning by the computers;
- Make shared resources for development the educational software, etc.

In this paper our approach in the realization some of the above goals is presented.

**Intelligent Tutoring Systems**

The endeavors to design Intelligent Tutoring Systems (ITSs)1 [Woolf, 1992] have opened a new research direction in the computer application for educational purposes. According to many authors, ITSs are capable of helping to solve difficult problems in the process of knowledge transfer, i.e. teaching and learning. They enable an almost full individualization of teaching process, as well as the student's advancement in the stage of acquiring knowledge and application of the new knowledge according to their capabilities, aspirations, previous knowledge and alike. Also, ITSs could stimulate the use of computers in education as a new education tool and teaching process controller, and opened new approaches for the use of computers in education.

We could say that ITSs are computer-based instructional systems that have separate data bases, or knowledge bases, for instructional content (specifying what to teach), and for teaching strategies (specifying how to teach), and attempt to use inferences about a student's mastery of topics to dynamically adapt instruction. ITSs design is founded on two fundamental assumptions about learning. First, that individualized instruction by a competent tutor is far superior to classroom style learning because both the content and style of the instruction can be continuously

---

1 Intelligent Tutoring Systems, Intelligent Learning Environment, Knowledge-based Tutors, Intelligent Computer Assisted Instruction Intelligent Educational Systems, are all, more or less, the synonyms for the using the computers in the process of teaching and learning by the aid of the methods and techniques of Artificial Intelligence.
adapted to best meet the needs of the situation. Second, that students learn better in situations which more closely approximate the situations in which they will use their knowledge, i.e. they "learn by doing," learn via their mistakes, and learn by constructing knowledge in a very individualized way. ITSs use techniques that allow automated instruction to come closer to the ideal, by more closely simulating realistic situations, and by incorporating computational models (knowledge bases) of the content, the teaching process, and the student's learning state.

**Promises and disappointment of ITSs**

When first introduced more than 10 years ago [Psotka *et al.*, 1988, Frasson, 1988], ITSs were announced and avowed as the future of education and training. Unfortunately, despite of the success of some ITSs (Shute’s Smithtown and Air Force, Clancey’s GIDEON and NEOMYCIN, Wolff’s Meno Tutor, Anderson’s LISP tutor, Nwana’s FITS, Wagner’s SCHOLAR, Johanson’s PROUST, etc.), ITSs have not see the general acceptance yet. Now, 10 years later the ITSs community is still talking about the promise of this technology, while searching for the leverage that will encourage its wide spread adoption. Much of this has to do with the complexities involved in definition and design ITSs applications, as well as the paradigmatic changes required of training and education organizations in the way they practice instructional design in order to accomplish this authoring [Clancey, 1996].

In the last half-decade ITSs have moved out of the lab and into classrooms and workplaces where some have proven to be highly effective as learning aides. These systems have been shown to be highly effective at increasing students’ performances and motivation too. For example, students using the LISP tutor [Anderson and Reiser, 1985, Anderson, 1990] completed programming exercises in 30% less time than that receiving traditional classroom instruction and scored 43% higher on the final exam. In another example, students working with an Air Force electronics trouble shooting tutor for only 20 hours gained proficiency equivalent to that of trainees with 40 months (almost 4 years) of on-the-job experience [Lesgold *et al.*, 1990]. Also, students using Smithtown, and ITSs for economics, performed equally well as students taking a traditional economics course, but required half as much time covering the material.

Simply said, there are only a few ITSs in real classrooms. The question is why? In the next sections we try to identify some of the reasons which will give the answer on that question.

**The Significant Cost of Development of ITSs**

As the modern intelligent tutoring systems are becoming more common and proving to be increasingly effective; each one must still be built from scratch at a significant cost, the big developing team and the large computer resources. Contrariwise, the software reusability and modern trends in software engineering, defined as the
process of creating new applications using previously developed software, take the
main role in software implementation at the end of the ‘90s.

The primary goal of this technique is to improve both the quality and the
productivity of software [Frakes, 1994, Lim, 1994], and reduce the development
cost. The main reason for taking into account reusability issues in the ITSs field is
due to the fact that to build ITSs needs quite big development environments. The
implementations of an average ITSs require huge computing resources, lot of money
and time. Furthermore, the results are expensive and hardly portable to personal
computers. Taking into account that prototypes are built incrementally through
successive enhancements and refinements, the time and cost of development is largely
reduced if existing knowledge is reused.

The Theory-Application Gap

As research in ITSs continues to produce more refined systems, the gap between the
research community and educational community continues to become wider. The
educators understanding, and usage of these research results have been much slower
than the research progress. As this theory-application gap becomes wider, it becomes
more difficult for educators to participate in ITSs research and application. The result
of the research becomes increasingly academic and unconnected to the pragmatic
aspects of teaching and learning.

The next problem is that the educational programs are unlike among the different
countries. The required knowledge for learning some lesson or topic in some domain
is not the same even in one country itself (the north - south problems, country - town,
etc.), So, the need for some kind of flexible system is obvious, i.e. the system in
which some on-line changes made by the teachers is possible.

The Role of the Teachers

The teachers want to have the active role in design, in eventually update and
improvement of intelligent tutors. For these teachers, however, ITSs are the most
difficult one in terms of grasping the total aspects. Of course, they have a chance to
read articles about ITSs but what they can understand through the articles is confined
to the abstract level of comprehension. Experience in using ITSs directly through a
system, could transfer the inner mechanisms of ITSs. It would be a powerful tool to
help them in understanding and comprehending this new paradigm of teaching by
computers, as well as the possibility to create their own ITSs lessons.

This has prompted research in the field of intelligent tutoring systems tools, i.e.
the tools for helping educators to create the ITSs.
The Intelligent Tutoring Systems Tools

The intelligent tutoring systems are becoming more common and proving to be increasingly effective in their basic goal to individualize the learning process. But, as we point it out, the fact is that the definition and implementation of a typical ITSs needs quite big development environments and this process is very time consuming, i.e. the accomplishments of an average ITSs require huge computing resources, lot of money and time. Furthermore, the results are expensive and hardly portable to personal computers and as we know that kind of computers dominated in average schools.

Authoring tools or ITSs shells\textsuperscript{2} can speed ITSs development and reduce costs by providing essential infrastructure along with a conceptual framework within which to approach an educational problem. However, a commitment to a given ITSs tools entails other commitments that may or may not be suitable for a given application. For example, a tool may be appropriate for only certain kinds of task domains. Also, ITSs tools have not received the widespread reuse that one would expect from an economically successful technology.

At the current time we can consider ITSs tools in two different classes. Firstly there are the commercial systems like Authorware, ToolBook, Icon Author and SmartText. These systems have a number of advantages over programming languages such as help in lesson planning, and built-in widgets which make instructional presentation easier. What they lack is any AI. The entire adaptive behavior had to be programmed by an experienced user, i.e. computer professional. The other class of systems are the different research tools, such as IDE [Russell \textit{et al.}, 1988], \textbf{ID Expert} [Merrill, 1989], KAFITS [Murray and Woolf, 1990], COCA [Major and Reichgelt 1992], GTE [Van Marcke, 1992], RIDES [Munroe \textit{et al.}, 1994], \textbf{Byte-sized Tutor} [Bonar, 1988], EON [Murray, 1997], etc. These systems come with knowledge base structure and interpreters much like and expert system shell. Their difficulty knows how much flexibility to offer to the teacher and how to make that acquisitions process as easy as possible. These systems have not made it out of the lab or seen wide use in multiple domains, and that happens because of one or more of these factors:

\begin{itemize}
  \item They are based on a specific instructional approach;
  \item The domains of application are limited because the system was modeled from an existing task-specific intelligent tutor and generalized to similar domains;
  \item They provide tools for structuring and using knowledge, but not for creating appealing student interfaces or learning environments;
  \item They are too complicated and complex because they are based primarily on theoretical concerns or AI techniques;
  \item Their usage still need the big developing team and huge computer resources, so the product is still expensive, and
\end{itemize}

\textsuperscript{2} In the rest of the paper, to avoid some misconceptions and different terminus, we denote the authoring tools and/or ITSs shells with ITSs tools, without exception.
These ITSs tools are very inappropriate for the changes of their elements (they simply copy the philosophy of expert systems shells), i.e. the definition of the elements are buried in the source code, so some changes could require the complete redesign of the shell.

In general, these tools, though some are fairly powerful and general, were not designed with significant user input, and do not address the practical issues encountered when educators actually use these systems (with the exception of COCA, which underwent some user testing and some not clear uses of EON).

The GET-BITS Model – Possible Solution?

In view of the above trends and research issues, and the mentioned problems in the earlier sections:
- individualization of the teaching process;
- student's advancement in the stages of acquiring knowledge and application of the new knowledge according to their capabilities, aspirations, previous knowledge and the like;
- searching for the leverage that will encourage the wide spread adoption of the computers in education;
- reduce the cost, the big developing team and the large computer resources needed for educational software, and
- overcome the gap between the research community and educational community,

the goal of designing the EduSof system was to construct a conceptual framework for representing the objects, events, responses, reactions, and relationships involved in tutoring. Also, the aim was to build a highly usable knowledge acquisition interface for rapid prototyping and easy creation, modification, deletion, and testing of both teaching concepts and tutorial strategies. Finally, our intention was to produce a system that enables teachers to build their own lessons without any computer or AI specialist, the ITSs environment which works on ordinary PC or similarly hardware, on different platforms and with which the production of the ITSs systems will be cost-effective. The EduSof is based on the GET-BITS (Generic Tools for Building ITS) model of intelligent tutors is described [Devedzic and Jerinic, 1997].

The GET-BITS Approach of Designing the ITSs

Our try to solve these problems with ITSs and ITSs tools is that instead a traditional expert systems shells approach (like the most of above ITSs tools) we used the object-oriented approach combined with the component based architecture in designing the new version of EduSof system [Jerinic and Devedzic, 1996]. The previous version of EduSof suffered from some deficiencies, which we tried to overcome in the new version.
First, different kinds of knowledge in the modules of EduSof were designed separately, although all of them conceptually had much in common. Second, decomposing the system functionally to the above modules made it hard to make additional changes and extensions when they were needed. Adding new knowledge representation techniques when needed required substantial changes in several modules. Finally, whenever there was a need for a change, not much of the relevant software could be used without any change. To design the EduSof in the GES manner, we first examine what we need for the developing the GES as the ITSs and ITSs tool, and from the source of the learning/teaching – the thinking circle.

What Would be Nice to Have When we Develop ITSs?

The ITSs models used in most of mentioned systems, shells, and authoring tools, as well as the corresponding knowledge models differ only to an extent. However, the design methodologies employed vary a lot, and sometimes even remain blurred for the sake of the system functionality alone. On the other hand, using a shell for developing an ITS brings more systematic design, but can also become a limiting factor if the shell doesn't support a certain knowledge representation technique or design strategy that may be needed in a particular system. Many researchers have also noted that current ITSs are usually built from scratch, and moreover, knowledge embedded in ITSs does not accumulate well specifying functionality’s of software modules of current ITSs often implies a lot of difficulties.

So, our starting position is what we need and what would be nice to have to design the ITSs. It would be very nice if:

- we could easily assemble our ITSs, shells, authoring shells, agents, etc., from existing and pre-tested pieces of software, without the need to develop and implement them from scratch;
- we could have our shells and toolkits offering us only the tools and options that we really need; we don't want our shells and toolkits to lack a tool or an option that we really need in a given project, but we also do not need a whole bunch of unnecessary tools and options from them;
- we could easily replace any piece of software in an ITSs by a similar (and possibly new) one, without any serious harm to the rest of the system; this would allow us, for example, to experiment with several versions of our system, each one having a certain functionality implemented in a different way (i.e., by a different piece of software);
- in order to develop a new piece of software that we find out to be necessary in our project (and it happens frequently) we could have some other piece of software to start with; the other piece of software should, of course, be logically and functionally similar to the desired one whenever it is possible;
- we could automatically refresh and update the repository time after time, putting in it some new pieces and deleting some pieces of software
that are no longer needed, based on the functionality and use-statistics information;

- the access to the repository could be as easy as possible, e.g. through Internet or an Intranet; in other words, if we could easily get, put, organize, and update software in a remote repository.

In short, it would be very nice if we could concentrate more on design of ITSs, and automate their implementation and maintenance as much as possible. It would be very nice if we could dedicate our work mostly to cognitive aspects, learning and teaching issues, and effectiveness of ITSs, and having most of the software changes in them done quickly.

The Learning/Teaching Process – the Thinking Circle

To resolve our goals and above assumptions in designing the useful ITSs tool, and to overcome the described problems and deficiencies in the former approaches in ITSs and ITSs tools, we start from the origin of teaching and learning.

First, we start to investigate the human problem solving mechanism and the human learning process. What are the characteristics of human thinking in relation with learning? We start with currently known features of intelligent systems. The intelligent systems may be roughly defined as computer programs that use knowledge represented explicitly to solve problems, or to help the user in making decision, or to help the people in learning. Based on this definition, one can argue that the most important focal points are knowledge, process of solving problems, and learning. It means that to realize in the computer any intelligent program, we must solve the following problems. How to represent the knowledge? How to use that knowledge? How to improve and increase that knowledge?

As such we regard knowledge as being everything that an individual knows about a specific universe at a given time. Knowledge for us, people, is important because almost everything that we do is in some way based on the knowledge that we have stored in our brains. In other words, people apply knowledge to achieve their goals. Therefore, the process of problem solving or thinking is also a routine of applying knowledge, here, with a purpose of finding out problem solution.

This observation suggests that application might be the most important but it is not the only human activity involving knowledge, which is significant in this context. Really, application is just one of the four activities that together build the thinking process, or the process of how the humans solve some problems, and in general how the people learn. In Figure 2 is shown the thinking circle, i.e., a sequence of human activities through which knowledge passes when people apply it to solve problems. As such, the first important phase in this context is the process of getting knowledge, i.e. the methodology of enriching someone's expertise. This involves collecting knowledge from many sources, assimilating this knowledge and storing it.

As knowledge is being assimilated, the second phase comes in play. Knowledge must be represented and organized in some form in our brains so that we could find and apply it when needed. This phase is denoted as the process of knowledge representation and memorization.
Once stored, it is than ready to be applied in the next phase, the human problem solving process itself. This means the active transformation of stored knowledge creates new one that is, in turn, used to achieve a goal. When the goal has been reached, people usually automatically incorporate the new knowledge that they have inferred during the solving process. However, assimilation is not enough to keep this new knowledge available over many years. It is also important to transmit it to the other people, or to know to explain why it that knowledge deduced is.

![The thinking circle](image)

**Figure 2.** The thinking circle

The transfer of new knowledge generally involves explaining to people how one comes to this acquaintance. Explanation is therefore the fourth and the last phase in this context. As people are having things explained, knowledge is again being taken so that the whole cycle is completed, than.
Second, the nature of education is being transformed in many classrooms where innovative teachers are integrating new computer technologies and the new methodologies. Learning (teaching) activities are becoming more student-centered, **constructivist**, and authentic. During the few past decades, constructivism as an instructional model has gained the attention and support of many educators. A basic premise [Piaget and Chomsky, 1979] of constructivist theory is that students build new knowledge and skills on pre-existing knowledge and skills, actively constructing meaning, collaboratively, in a rich contextual setting, as they follow learning paths that suit their own interests and needs. Learning (as well as the teaching) is not result of taking notes of experience that will be executed without structuring, which is the consequence of the acting. Also, there is not a priori, or inborn cognitive structures – only the functioning of the intelligence is inherit and that function give the creation of the structure only if successive actions, which are performed on the objects, are being organized. From that arise, epistemology of the origin, methods, and limits of the human knowledge (i.e. learning or teaching) could not be empiricivist or performacist, but only constructivist with incessantly elaborating the new operations and structures. So the simple question gets up: **“could we follow those principles in designing ITSs and ITSs tools?”** The answer is why not?

---

**Figure 3.** The “LEGO” system – GET-BITS model

- Domain knowledge
- Explanation knowledge
- Control knowledge
- Levels of abstraction

More...
The GET-BITS Model

The simplest way to describe the GET-BITS is to imagine the well-known children building blocks (for example LEGO™ system), Figure 3. The main idea is that we define and realize the software pieces, which represent the knowledge representation techniques, control knowledge, action methods, etc. involved in tutoring and learning. These blocks have their connection, horizontally and vertically, and they are used to make the upper levels, i.e. the units for learning/teaching, systems, integration, etc. The building blocks could be simply changed with new one, if the connections to the upper level or left-right blocks are preserved and saved. These blocks could be used to realize the ITSs, the ITSs tools, and/or some parts, modules for testing some new technique of tutoring, student modeling, etc.

To develop the GET-BITS in the above manner we start with the representation of the knowledge first. The essence of the model is a unified abstraction of different knowledge representation techniques and different models of human knowledge. All kinds of knowledge (e.g., domain, control, action, explanatory, etc.), as well as all types of knowledge representation formalisms (e.g., rules, frames, logic, etc.), can be viewed from the perspective of an abstract and fairly general concept of knowledge element.

Second, the power of using AI technology to build an educational software package results from the possibility of representing abstractions and models in the computer system. This means that it is possible to encode some abstract entities such as "when the pupil is confused give more explanation," or "give topic overviews only for the beginning level," or "give response for a right answer."

<table>
<thead>
<tr>
<th>Level</th>
<th>(I)</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>(S)</td>
<td>System</td>
</tr>
<tr>
<td>Level</td>
<td>(B)</td>
<td>Blocks</td>
</tr>
<tr>
<td>Level</td>
<td>(U)</td>
<td>Units</td>
</tr>
<tr>
<td>Level</td>
<td>(BO)</td>
<td>Primitives – basic, operatuions</td>
</tr>
</tbody>
</table>

Figure 4. The levels of abstraction in GET-BITS

Third, we can define appropriate class hierarchies for developing ITSs and the five levels of abstraction on all segments and parts of GET-BITS and all models and systems inherit from them Figure 4. That's what GET-BITS are about. It also starts from the concept of knowledge element, and derives meaningful subclasses that are needed for building a wide range of ITSs. However, classes for knowledge representation are not the only tools needed to build an object-oriented ITS. Apart from knowledge of various kinds, in each such ITSs there must also exist some
control objects that functionally connect the system's modules, handle messages, control each session with the system, monitor student's reactions, etc. In other words, such objects provide control/action and handle dynamics of ITSs. **GET-BITS** also specify classes of these control/action objects.

The **GET-BITS** model is supported by a number of design patterns and class libraries developed in order to support building of intelligent systems and ITSs in particular. In fact, designing and developing an ITSs based on the **GET-BITS** model is a matter of first developing an ITSs framework, such as **EduSof** (Figure 5) and then using it for development of the ITSs itself (**LeaPas, Hart,** etc. for example). Also, the proposed model could be used to design the ITSs, like **Flute, ForLan,** etc. directly using the pieces of knowledge, i.e. blocks from the **GET-BITS** model. In spite of the fact that this means starting the project without a ITSs shell, it is a relatively easy design and development process, because of the strong software engineering support of the design patterns and class libraries [Jerinic et al., 1998].

Finally, the **GET-BITS** could be used to design the intelligent user interfaces, the separate part of ITSs such as student model **COSMO,** or **Topic Editor,** as well as the tools for human-computer interaction, or web-based and distance learning.
Along with the high modularity and reusability provided by the class libraries, potential design flexibility is another important advantage of using the GET-BITS model. Development of a GET-BITS-based ITSs tools means putting together only those pieces of software from the class libraries that are really needed for a given application. In mentioned ITSs tools, if any additional class is needed, it must be designed and developed by the tool (i.e. shell) developer. Fortunately, the class hierarchies and design patterns of GET-BITS provide a firm ground to start from in such an additional development. Additional subclasses can be derived either directly from any of the already existing classes. The classes of the GET-BITS model are designed in such a way to specify “concept families” using the least commitment principle: each class specifies only the minimum of attributes and inheritance links. That assures the minimum of constraints for designers of new classes.
The GET-BITS and EduSof ITSs Architecture

As the ITSs are complex software products and represent the connection of several disciplines: education, AI, computer science, cognitive science, psychology, pedagogy, etc, the traditional ITSs [Woolf, 1992] consists of:

- **Expert Model**
- **Domain knowledge module**
- **Student model, Tutor or Pedagogical module**
- **Diagnostic or Misconception module** and
- **Interface or Communication module**

We extended these ITS concept and their components with Explanation module [Jerinic and Devedzic, 1997]. In Figure 6 we connected the traditional and extended definition of ITS components with human thinking cycle, presented in Figure 2.

**Domain knowledge** module contains information the tutor is teaching, and is the most important since without it, there would be nothing to teach the student. Generally, it requires significant knowledge engineering to represent a domain so that other parts of the tutor can access it. One related research issue is how to represent knowledge so that it easily scales up to larger domains. Another open question is how to represent domain knowledge other than facts and procedures, such as concepts and mental models. This module entails a general problem in AI: How do we represent domain knowledge? In GET-BITS, the domain knowledge expert module is represented with the knowledge network of semantically connected frames. That knowledge network represents the content to be taught in terms of lessons, concepts, rules, tasks, questions, actions, and examples and their interrelationships. It can be taught of as more than just a network of facts. The knowledge network in GET-BITS could be coupled to a domain expert, or could be a collection of text, pictures, simulation procedures, and/or voice.

**Expert model** is similar to the domain knowledge in that it must contain the information being taught to the learner. However, it is more than just a representation of the data; it is a model of how someone skilled in a particular domain represents the knowledge. Most commonly, this takes the form of a runtime, or executive expert model, i.e. one that is capable of solving problems in the domain. By using an expert model, the tutor can compare the learner's solution to the expert's solution, pinpointing the places where the learner had difficulties. The expert module consists of the domain knowledge to be tutored. This component of the ITSs is assigned task of generating problems and evaluating the correctness of the student's solutions. In GET-BITS, the expert module is represented with the control knowledge and with the concept of transaction.

**Interface module** is the mechanism by which the student and the tutor communicate. No communication can exist without this component, and therefore it has been a part of every ITS developed to date. Presenting this module as a component of the ITSs architecture reflects the current view of the importance of a well-designed user interface to any interactive program. The general goal of the user interface is to use the available devices, usually keyboard, mouse, monitor, etc., to display to the student the necessary information, and use these devices to obtain
students' responses. Interactions with the learner, including the dialogue and the screen layouts, are controlled by this component. How should the material be presented to the student in the most effective way? This component has not been researched as much as the others, but there has been some promising work in this area. In GET-BITS we use the menu-driven, the icon managed, and easy-to-use user interface.

**Tutor module** is responsible for deciding on how and when the domain knowledge is to be presented to the student. This component provides a model of the teaching process. For example, information about when to review, when to present a new topic, and which topic to present is controlled by the pedagogical module. As mentioned earlier, the student model is used as input to this component, so the pedagogical decisions reflect the differing needs of each student. Because the tutor needs to know information about the student to be able to make these decisions, the tutoring module relies heavily on the student-modeling module. In GET-BITS the tutor module consists of various teaching strategies, the rules that decide what should be taught to the student given the current state of the student model. This is realized by parameterizing the semantic network of frames, with the combination of If-Then-Action rules, as the part of the control knowledge.

**Error or diagnostic module** in GET-BITS consists of the rules used to identify student misunderstandings, updates the student model, manages the teaching rules, if it is necessary, and stores the student common mistakes and errors.

**Student model** stores information that is specific to each individual learner. At a minimum, such model records how well a student is performing on the material being taught. A possible addition to this is to record misconceptions as well. Since the purpose of the student model is to provide data for the pedagogical module of the system, all of the information gathered should be able to be used by the tutor. In the GET-BITS model we extend the student model with the psychology type, i.e. the relevant knowledge about an individual student concerning the flow of information in a ITSs is referred to as **Student behavior**. As a high level component, it can be seen as an aggregation of two other components: **Student history** and **Student model**. Although some authors discuss whether student history should be naturally included in the domain knowledge or it should exist as a separate module, that problem is not of high importance in the GET-BITS model. Student behavior component and its parts look the same regardless of the part of the knowledge base where they are included, their relations and communication with other components are always the same.

The primary purpose of the part of the knowledge base that we refer to as the **explanatory knowledge** is to define the contents of explanations and justifications of the ITSs learning process, as well as the way they are generated. Explanatory knowledge is related to both domain and control knowledge, and often that knowledge is treated as a part of the other two components of knowledge bases. However, in the GET-BITS model it is treated as a distinct knowledge component, in which the knowledge about the learning process explanation, knowledge elements

---

3 As it is shown in Figure 6, the part of ITSs in GET-BITS denoted as the **psychology type** have influence on the other parts of ITSs, specially to communication module.
explanation, control strategies explanation, and other types of intelligent assistance is represented explicitly and is treated in its own right.

Among the knowledge elements that this component can include are:

- **Canned text** associated with rules and other knowledge elements in the other two components, templates that are filled by the explanation module when required (in order to generate the full explanation text in a given situation);
- **Presentation functions** needed for explanation of certain knowledge elements (e.g., some knowledge elements are best explained by using tables, others require the use of graphics, etc.);
- **User models** necessary for generating explanations in accordance to the user's knowledge level;
- **Criteria** used by the explanation module when deciding about what knowledge elements to include in the explanation and what to skip, as well as what level of details must feature the explanation.

It must be stressed that apart from the explanatory knowledge, the explanation module uses extensively the knowledge from the other two parts of the knowledge base as well when generating explanations. Therefore, the explanatory knowledge may also contain explicit descriptions of explanation control strategies. These can be specified in the form of control rules like, for example: "If the explanation type is WHY, then 1/ show the current goal, and 2/ show the current domain rule instantiation, and 3/ show the meta-rule that was last applied".

**The EduSof Framework**

The goal of developing the EduSof system was to design a conceptual framework for representing objects, events, responses, reactions, and relationships in tutoring and learning. The system is featured with an interactive graphical environment in which the teacher can manipulate objects at different levels of abstraction. These objects and relationships between them are used for defining, designing and creating a lesson. The intended users of EduSof are educators and instructional experts, not programmers. The framework is domain independent, and includes mechanisms for representing domain knowledge and control information, thus generating particular ITSs. These mechanisms are responsible for dynamically customizing the machine's responses.
Knowledge organization in EduSof

The knowledge that student must learn in EduSof is presented as the finite hyper-automata $L(F, K, F_0, E)$ where $F$ is the world of fact that student learn, $F_0$ the initial fact – state from which the lesson begin, and the $E$ is a set of final state/fact and the $K$ is the function for action after student $S$ represented with the student model $S$ learn or not learn the current fact $f_i \in F$. The action function is defined by $K: F \times K_{S} \times S \rightarrow F$. The system made the plan of the new goal in learning according the value of $K_n(S, T)$ for the current fact that student learn and the student model $S$, i.e. made the new action knowledge for that moment (Figure 7).

Figure 7. The lesson in EduSof

The main object classes are the Lessons and the way of their presentation. Units of knowledge that can be taught, repeated, summarized, etc., represent Lessons. They are categorized according to knowledge type, for example: text, picture, simulation, more examples, and so on. The concepts have pointers, including various types of prerequisite, part-off and related-misconception links to other concepts, forming the lesson network. They have pedagogical information such as summary, motivation, examples, tasks, etc., which point to presentation. The presentation is the array of questions or tasks, represent expository or inquisitor interactions with the student. They are composed of a task, such as a multiple-choice question, or problem solving

---

4 We assume that the main object of learning is not the tree, like many ITSs authors does, simply because that we believe that teaching must be guided process, i.e. the lesson must have the beginning, the end and the way to learn the lesson elements – topics.
exercise, and an environment for doing the task, such as a picture or simulation of some system. Presentation also contains the possibilities for responding to the pupil, such as hints, congratulations, elaborations of the answer, etc. That knowledge representation is concerning some particular and concrete’s type of knowledge elements that are necessary for the realization of the Expert module in any ITSs shell. Any lesson is consisted of one or more issues, modeled by the class Topic. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained at the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The issues that student must learn (class Topic) are realized with separate type of knowledge elements in the GET-BITS system. Any topic could be introduced with text information (Information), graphically (the pictures and/or the diagram - Image), and/or with the simulation of some event (Simulation). Also, the additional or further explanation (Explanation) for that theme, or some suggestions (Hints) could be defined. The class Topic in GET-BITS is made for specifying and defining the issues or notions needed for lesson creation.
Figure 8. The parts of EduSof

Abstract class TQ served for description of the common elements for the two comparable and connected classes, one for the definition of tasks or assignments (class Task) and the other class for the realization of questions or problems (class Question). That class has the fields WhyPtr, HowPtr, WhatPtr, etc. With those fields the lesson creator made their explanations for the task and/or question. The instances of that class are given to the student during the learning process.
The EduSof system and the appropriate knowledge database are designed as a ITSs framework. The structure of knowledge used in them is dynamic. The knowledge base is realized using the frame system, based on the GET-BITS model. Teachings and examining methods for employing lessons or knowledge are widely opened for lesson creators (usually teachers), and are independent of the system. EduSof has a great capacity of representing knowledge through teaching sequences in almost all fields of educational process.

![Diagram showing EduSof system](image)

**Figure 9.** The Tea module – creation of the lesson, the Topic editor

The EduSof system is organized in more separate modules (Figure 8). The creating module, i.e. the Topic Editor (Figure 9) Tea serves for designing of lessons by the teacher of the subject, and the module The user – teacher simply by the mouse chose the appropriate object on the left side of the screen and developed the lesson. In EduSof, a teaching sequence is organized as a list of semantically connected elements - Lessons. They consist of a list of Topic (Figure 10). Topic, as basic component of a teaching sequence, can be expressed by means of Text, Picture, Voice and/or Simulation. For each Topic, a list of Questions, Task or Problems are given. This object of learning could be used for testing the acquired knowledge connected to that Topic. For each Question, Task or Problem, a list of Alternatives, Explanations, Hints, More Information, etc. are given.
For each Topic element, the Control/Action mechanism (Figure 11 - Connections) to decide What Next the pupil will learn, in case that a right or wrong answer is chosen, as well as Explanation, Buggy Model and the learning/teaching strategy.

The abstract class PQ describes common elements of two other classes having many in commons and derived from the PQ class. One for specifications of problems that the student is supposed to solve (exercises or assignments) - the Problem class. The other class serves for putting questions that require simple answers from the student during an assessment session - the Question class. The PQ class has the fields WhyPtr, HowPtr, WhatPtr, etc. With these fields lesson creators can make their explanations related to a problem and/or a question. The instances of the Problem and Question class are presented to the student during the learning process.
Discussion and Conclusion

The purpose of the proposed GET-BITS model of knowledge bases and knowledge base management is to make a basis for applying the ideas of object-oriented software design methodology to ITSs knowledge organization, representation, and access. It covers all-important aspects of knowledge bases, like their contents, knowledge representation techniques, using, updating, extending and maintaining the knowledge, etc. However, it is important to stress again that the GET-BITS model should be regarded as an open framework for developing ITSs knowledge bases, rather than as a closed set of design rules and organizational hierarchies. In fact, the model has several open ends. The first open end is the possibility of extension by including new knowledge representation techniques. Such an extension can be done extremely easy, without reorganizing the entire GET-BITS model. As another open end can be easily extended to support new transaction types. This is particularly important if it is required to support some specific complex transaction. Moreover, it is possible to make an extension to support a certain structured query language for examining knowledge bases, an analogy to the SQL in the database technology.

The presented method for building a knowledge base with explanation facilitates, suitable for the educational purpose is currently under development and the realization in the present form. In sum, explanation part of GET-BITS is capable of playing the role of:

- a “research resource” that helps a student find information he/she needs;
- chooses between alternatives when they are available;
- adds related information that motivates, enriches, and enables the student's understanding of the primary material selected;
- orders all of this for a coherent presentation in a multimedia environment.

The GET-BITS model of intelligent tutoring systems, presented in the paper, allows for easy and natural conceptualization and design of a wide range of ITSs applications, due to its object-oriented approach. It suggests only general guidelines for developing ITSs, and is open for fine-tuning and adaptation to particular applications. ITSs developed using this model is easy to maintain and extend, and are much more reusable than other similar systems and tools. The model is particularly suitable for use by ITSs shell developers. Starting from a library of classes for knowledge representation and control needed in the majority of ITSs, it is a straightforward task to design additional classes needed for a particular shell.

The presented method for building a knowledge base suitable for the educational purpose is currently under development and the realization in the present form. In the current state of development we test usage of EduSof in some summer schools in various subjects (chemistry, history, geography etc.). With introduced approach in building the intelligent tutors we realize the one of the goals we set down before the start of the realizing of the above concepts. The teacher's independents in design of the intelligent tutoring lessons were almost 70% after an hour of explanation how to use the EduSof. The rest 30% is used for the professional programmer’s time for designing the simulation procedures and pictures. It is supposed that the teachers have
some knowledge of using the computers, a little knowledge in computer graphics, and some help of professional programmers' in making the simulation's procedures.

In the 1997/98 school year the LeaPas, the ITSs made by the aim of EduSof and GET-BITS is used in teaching the “Introduction to programming”, course on the first year of computer science studies at the University of Novi Sad. We use one group, about 20 students, and through practical exercises they used the LeaPas instead the classical method “chock and table, teacher explanation and try on computer alone and ask the professor for help”! At the final examination the average success of testing the students that are training with the traditional talking or class (the teacher are talking to the group of students from 50 to 200, and students practice alone with assistants on computers) method is between 40 and 60 form 100 points. When the teacher with same teaching strategy use the diagnostics test (to improve and modify his/her approach to that teaching subject), the results are better, i.e., about 74 from 100. But, when the students have individualized training with LeaPas, the results are 90-91 from 100.

Further development of the GET-BITS model, as well as the EduSof, are concentrated on development of appropriate classes in order to support a number of different pedagogical strategies. The idea is that the student can have the possibility to select the teaching strategy from a predefined palette, thus adapting the ITSs to his/her own learning preferences. Such a possibility would enable experimentation with different teaching strategies and their empirical evaluation. Another objective of further research and development of GET-BITS is support for different didactic tools, which are often used in teaching.

References


