
KNOWLEDGE REPRESENTATION IN INTELLIGENT TUTORING FOR THE FIRST COURSE FOR LEARNING OF PROGRAMMING

Abstract - In this paper an overview of using of EduSof system (shell framework for realization of intelligent tutoring system) in designing of intelligent tutor for learning of basic programming is given. The aim in realization of LesPas system, intelligent tutor realized by using EduSof shell, is to help beginners in programming in solution defining, learning language construction while using adequate examples, as well as in implementation and testing of his own programs. Intelligent tutoring system LesPas does not only give reports on syntax, semantic and conceptual user errors. It also tries to understand users point of view while designing solution of given problem, to help, to give advice and accept original ideas, i.e. self learning is possible. Intercode representation of knowledge of basic programming enables pupil to direct creation of his solution. One mental model is given, the way of thinking, for better problem solving in basic programming.

1. Introduction

Computers are in education for more than twenty years [1, 2]. In the last couple of years in the world different research about artificial intelligence methods in different educational systems, i.e. for designing and making of educational software. The final aim of this approach is individualization of educational process, and that every pupil accepts contents according to his capabilities. In contrast to other educational means (graphoscope, diascope, television) computers bring into educational process more quality, i.e. possibility of direct communication between an user-pupil and a computer-teacher (for example question-answer).

Fast development of computer technology in the last decade, as well as great use of techniques and methodology of artificial intelligence (AI) in the realization and use of before all expert systems leads to the use of those methods in the education. In the late 80's in the world new research area is introduced-intelligent tutoring systems(ITS). This area joins work from methods and techniques of artificial intelligence, computer science and leaning and teaching.

The difficulty of designing and developing more useable and cost-effective intelligent tutoring systems has caused the realization of some new approaches in that field, the realization of intelligent tutoring shells. Our starting point and perspective on development of ITSs shell is motivated by issues of pragmatics and
usability. The advancement of AI methods and techniques makes understanding of ITSs more difficult, so that the teacher are less and less prepared to accept these systems. As a result, the researchers and the educational community is constantly becoming wider, in the field of ITSs. Also, the present ITSs need quite big development environment, huge computer resources and in consequence are expensive and hardly portable to the personal computers. Considering commercially available and widely used authoring systems for traditional computer-based teaching, we try to give the next step, the next paradigm shift which is needed to enable some of the advantages of ITSs in that these tools.

Also, traditional Intelligent Tutoring Systems (ITSs) are concentrated on the field (domains) they are supposed to present, hence their control mechanism are often domain dependent. More recent ITSs pay more attention to generic problems and concepts of the tutoring process, trying to separate architectural, methodological, and control issues from the domain knowledge as much as possible. This was, also, one of the main ideas behind the ITS shell (environment) called EduSof [3], designed to allow fast prototyping of ITSs in different domains.

However, the original design of EduSof mechanisms for representing domain and control knowledge has proved to be fragile in maintenance and further development. Therefore, a new version of EduSof has been designed, considering a new model of ITS we developed using object oriented approach. It is called GETBITS (Generic Tools for Building ITSs) [4] and is essentially a specific extension of a more general, recently developed model of knowledge bases, called OBOA (OBject-Oriented Abstraction). First, we introduced the control knowledge of an ITS. The control knowledge describes the ITSs problem-solving process, reasoning strategies used by the inference engine, organizational hierarchy of reasoning modules and agents, control regimes corresponding to particular steps of the problem-solving tasks, the methods of elaborating the answers, etc. To an extent, this knowledge complements the domain knowledge-ideally, control knowledge is an application-independent description of how the problem-solving and reasoning activities are performed during the various ITSs operation, i.e. during the learning process. The same is with all the modern Expert Systems (ES), which have both domain and control knowledge representation in their knowledge bases explitely.

In this paper one application of EduSof shell for realization of ITS for supporting the process of learning of programming. This sort of system enables beginners to establish, design, implement and test their first programs. Realization of that intelligent system is made using intercode technique. Tutoring system finds and gives reports on users syntax, semantic and conceptual errors. However, system tries to understand pupils attitude toward solving the problem and eventually partially or completely understands and verifies the whole result of given problem in learning of the programming, as well as elements of the programming language itself and basic concepts (data type, flow control), based on great experience of the author in tutoring in bases of programming.

2. THE ARCHITECTURE OF THE KNOWLEDGE IN EduSof

The original goal of developing of the EduSof system was to design a conceptual framework for representing of the objects, events, responses, reactions and relationships used in tutoring and learning. The system is featured with an interactive graphical environment in which the teacher can multiple objects at different levels of abstraction. These objects and relationships between them are used for defining, designing and creating of 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 a particular ITS. These mechanisms are responsible for dynamically customizing the machine's responses.

In the EduSof, a teaching sequence is organized as a list of a semantically connected elements - Lessons. They consist of a list of Concepts. Concepts, as a basic components of a teaching sequence, can be expressed by means of Text, Picture, Voice and/or Simulation. For each Concept, a list of Questions is given, used for testing of the required knowledge connected to this Concept. For each Question, a list of Alternatives is given, as possible answers to it, or the right Answer. For each Alternative, a Right Answer is provided. Also, the information about What Next the pupil will learn, in the case that a right or wrong answer is chosen, are given.
The EduSof system is organized in three separate modules. The crating module Teacher, serves for designing lessons by teacher of the subject, and the module Learn is a kind of interpreter of the knowledge introduced by the Teacher module, used by pupils while mastering the subject-matter and getting the pertinent knowledge. The third module can be used to test student for marking.

There are six modules in EduSof architecture: Expert (containing domain knowledge that an instructor enters), Student (student model), Tutor (knowledge of teachers strategies and pedagogical knowledge), Diagnostics (evaluation of student's knowledge), Interface (user interface), and the Explain module [5], our extension to theory of ITS. These modules together make EduSof capable of handling basic knowledge (lessons, text, pictures, etc.), additional information (for helping and improving the use of basic knowledge), learning mechanisms and student's reactions.

Currently, we consider authoring systems or ITSs shells in two different classes. At first, there are 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. All the adaptive behavior has to be programmed in by an experienced user. The other class of systems is the class of different research tools, such as IDE (Russell et al, 1988), ID Expert (Merrill, 1989), KAFITS(Murray and Woolf, 1992), COCA(Major and Reichgelt, 1992), GTE(Van Marcke, 1992), RIDES (Munroe et al., 1994), Byte-sized Tutor (Bonar et al., 1986) and EON(Murray 1996). These systems come with knowledge based structure and interpreters much like as expert system shell. Their difficulty is knowing how much flexibility to offer to their teacher and how to make that acquisition process as easy as possible. These systems have not made it out of the lab or seen wide use on multiple domains and according to T. Murray that happens because of one or more of these factors:

- They are based on a specific instructional approach;
- The domains of application are limited because the system was modeled from an existing task-specific intelligent tutor and generalized to similar domains;
- They are too complex because they are based primarily on theoretical concerns or AI techniques;
- They provide tools for structuring and using knowledge, but not for creating appealing student interfaces or learning environments.

In general, these ITSs, though some are fairly powerful and general, haven't been 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).

We try to solve these problems with authoring tools and ITSs shells so that instead a component architecture for ITSs shells (like most of these shells) we used the object-oriented approach while designing the new version. First, different kind of knowledge in its modules 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 are 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.

3. EXPERT KNOWLEDGE IN LEARNING OF PROGRAMMING

In the expertise of knowledge for realization of learning of programming basis as well as contents of this expertise there are two cases:

1) problem and formal representation required knowledge for etching of programming, and
2) problem of learning, itself and acquisition of that knowledge.

Although programming (or software development) can be considered as relatively young discipline, which includes that methodology of learning of programming is not completely developed, just the learning of programming has additional difficulty in handling and use of abstract as well as computed concept and
process. From one point of view, in programming defining and designing of object and program are used, which is similar to other areas that use design and construction, for example engineering. From the other point of view, the nature of these objects which are being defined and used, require strong theoretical base and acceptable programming that leads ultimately to mathematics. Although great results are achieved in programming theory, before all, in corresponding ways of program forming description, programming languages, programming transformations, predicate logic, correctness proving etc., at the present point of development of computer science we still have one practical and useful methodology which will cover the problem of construction and program realization and methodology convenient for unification of development process of the whole software.

From above mentioned reasons the problem of learning of programming is specially difficult, and from that expertise of knowledge needed for effective and correct lecture and learning of programming basics. The aim is defining of methodological aspects of programming learning, i.e. defining corresponding expert rules in knowledge realization, the abstraction technology is chosen and intercode, i.e. realization of ideas of visual programming with communication on formal language which is subset of Serbian.

4. PHASES OF PROBLEM SOLVING

Realization of problem of learning of programming in this intelligent tutoring system begins with defining of elementary problem to the programmer the beginner. After that the process of problem solving in steps is controlled and supervised, and eventual semantic or syntax errors are being taken care of. In mentioned process advises are given for more correct and effective solution and at the end the code is generated in needed programming language, in concrete case in PASCAL. Realization of so based programming learning is happening in three phases.

The first phase bring informal construction to solve given problem. Language on which the interaction between pupil and teacher is made is of very high level (subset of Serbian) and is very similar to the way that programmer writes informal instruction in an pseudolanguage or draws data flow diagram of program execution, describe to the beginners the process of solving of this problem. For example, pupil can construct informal instruction(sentence): “execute block and do it while sum is greater than 100”. After that solution analyze is made. Eventual conceptual errors are being discovered. Before going to the next phase, pupil must complete the whole development of solution in this language.

In the second phase, refined and more detail program blocks defined in the later phase, are being introduced. This is done by introducing of half-formal programming plans. Plan is similar to the structure in data flow diagram and describes how given aims are transformed into real program code. In the paper of Soloway and Ehrlich [6], Navrat and Rozinajova [7] and Bonar and Liffick [8] similar method is used. In the learning by using EduSof shell, the array of advancements is introduced in knowledge base, gained in work experience with programmer the beginner. Usually plans have different roles as well as correlation and influence on other plans. The access to problem solving using plans enables pupils to concentrate more on solution, use corresponding plans, without barrier on syntax complexity and programming language realization (data type, corresponding operators, ways of memorization, different language limitation) which must be taken into the consideration when explicitly programming.

The role of plans and their definition depends on chosen programming language, because there are different programming styles (imperative, declarative). Of course, that there is a group of plans common to for couple of styles. In the sole plans are a sort of atoms, i.e. axiom which are used while programming. In this plan based programming learning it is possible to test, i.e. pseudo-execute atom. Plans are being managed by using parametrised semantics rules net: “If there is use of STRUKTURNIH TIPOVA and there is no use of ULAZA then use DINAMI^KI TIP and GENERIJ”.

In the third phase translation of plans is being carried out. Plan base conception is translated into real code using corresponding programming language, in our example PASCAL. In this way generalized programming code can be corrected using simple editor, as well as testing using corresponding translator.

5. THE USE OF THE SYSTEM
Learning of programming in intelligent tutoring system begins with giving the problem to the pupil. This problem must belong to the beginner exercises. The tasks are being chosen that they present in simple and effective way basic concepts of imperative programming, like splitting or iteration. Pupil while solving given problem goes through three described phases.

In the first phase pupil controls the array of instruction in formal language, subset of Serbian language, use technique of modular development of program (bottom up or the other way around, depending on level of learning). Since the system is current phase realized for learning of imperative programming, block technique, step by step technique and structural programming technique is used (every block has one input and one output). In the next phase given construction is developed using plans and program is built using plan representation. In the last phase plans are being further developed and concrete program command in given programming language is generated. In current phase programming language PASCAL is chosen. In further work learning rules for programming languages PROLOG and LispKit Lisp are going to be realized.

In description of use, as well as this system use demonstration, suppose that following problem is given:

"Write a program which inputs from keyboard array of integers, until given integer number 999 is input and computes average of this array"

Pupil from the main menu chooses the option marked with Odabir pojmova na srpskom jeziku, which as submenus has beginning phrases: "Izražaj", "Îtampaj", "Izlaz", "Nastavi", "Uzmi", "Uštaj", "Radi dok", "Za" etc. By choosing of some construction a new submenu for further work is given. For example for option "Uštaj" submenu has "Jedan ceo broj", "Više celih brojeva", "Jedan realna broj" etc. There is description to the user like "[ta da uštajam?" On every level there are options for help, examples, additional information and/or explanation why that option is used right now. System for bad use of some construction explains why it is misused or warns user about possible errors which will come up later.

After first phase, as one of possible solutions the following array of sentences is gained, i.e. pseudo-commands:

Uštaj ... cele brojeve
Saberi ... ceo broj
Izbroj ... svaki ceo broj
...
Radi dok se 999 ne uštaj
Izražaj
Îtampaj

Every from above mentioned completed or uncompleted pseudo-constructions is automatically presented on the interaction window and can be changed with adequate option. After this phase is finished tutor symbolically compares representation of this version of program with list of demands in concrete anticipated solution. This list of demands is created in the phase of lesson preparing by the teacher as the part of problem description in corresponding language.

The list of demands is usually given for more different levels. These levels are adequate to different student models for which expert considers that the student can create. The differ from exact solution, in the sense that in them not all plans must be defined, they are freed from eventual borders and the let flexible pointing. For example, one of these student models for loop explanation, enables that pupil makes loop construction by describing just the first iteration, which is followed by the "I tako dalje...". Based on expert rules system tries to conclude weather the first iteration is good, and if this is the case informs the pupil how to complete the whole expression. In contrast, if the use and definition of this first iteration is not good it gives explanation with possible consequences. If the system is unable to conclude what the pupil wanted it informs the user about that with request for further specification.
Diagnostically the part of strategy used in this phase is based on comparison and finding of a student model for some problem which is being solved. Every model requires some plans made in subset of Serbian, as well as corresponding organization of these plans. They analyze the pupils solution on that informal level, tutor computes how this solution satisfies requirements for every realized student model. Further learning and tutor behavior is leaded and realized by using the most simple unsatisfied plan of the most simple student model with one or more unsatisfied plans.

In the phase two further work on the specification given in the phase one is done. Variables, connections between concepts and plans are introduced etc. For example when loop is being done for input from the example the following result is gained:

Po-etak Petlje  
U’itaj X  
Saberi Ukupno sa X  
Izbroj Koliko  
Kraj Petlje

In the third phase with source to source technique translation from intercode representation of pupils solution automatically PASCAL program is generated. Code produced in this way can be modified with the call to little editor as well as with the call to corresponding interpreter or tester.

6. Conclusion

In this paper the main interest is put on realization of one model for teaching of basics of programming. Intercode technique is used, i.e. pseudo-representation of programming code in the language close to Serbian is used. System is in the final testing phase. In the next development phase the similar methodology will be applied for learning of basics of programming in PROLOG with the aim that the corresponding plans and rules of deduction comparison with the tendency toward unification of these two intelligent tutors.

Literature