TOBIE: an Implementation of a Domain-Independent ITS-Architecture in the Domain of Symbolic Integration

Julita Vassileva

Software Engineering Department
Institute of Mathematics
1090 Sofia, P.O. Box 373, Bulgaria

Universität der Bundeswehr München
85579 Neubiberg, Germany
e-mail: jiv@informatik.unibw-muenchen.de

ABSTRACT
An implementation of an ITS-shell architecture in a specific domain is described. Various features of the architecture are demonstrated on examples from the domain. Special attention is paid to the issues of domain-knowledge representation, student modelling (incl. diagnosis), the pedagogical decisions taken by the system in both a tutoring and coaching style of instruction.

1 Introduction

During the second half of the last decade one of the focuses of interest in Intelligent Tutoring Systems (ITS) research was finding architectures applicable to different domains. A practical stream in this direction is the development of ITS-shells [Vass90], (Major & Reichgeld, 1992); (Nicaud, 1992); (Murray & Woolf, 1992).

The major problem in developing an ITS-shell is to find a unified way of representing the knowledge to enable the system to model the student and to take pedagogical decisions. Therefore, the main goal of our work was finding a unified way of representing domain-specific knowledge that integrates **expert knowledge** about the domain (e.g. the goals and sub-goals in solving a problem), **pedagogical knowledge** (e.g. the curriculum), **diagnostic knowledge** (how to interpret student's actions) and, in addition, a **unified scheme across different domains for taking pedagogical decisions** was created. These issues are presented in the paper. Although all examples are related to the domain of integration, in which the prototype TOBIE was implemented, they can be easily generalised.

2 Architecture

The architecture of the system consists of the following components: a domain-independent Pedagogical Component, an Authoring Component (Administrator), a model of the student's domain knowledge (we shall call it briefly Student Model) and a model of the student's individual characteristics (see Figure 1).

The "kernel" of the system is a Domain Knowledge Base. It contains

- a black-box domain expert program for solving problems;
- an integrated representation of all domain-dependent knowledge that is needed by the invariant Pedagogical Component to lead instruction and to create and update the two student models.

According to the target level of instruction and the author's viewpoint on the subject, this knowledge can be organised in various ways.

2.1 Knowledge Organisation

With "knowledge organisation" we denote:

• **Decomposition** of knowledge into primitive elements (learnable units, beliefs, concepts, problem solving steps etc.). The elements of knowledge decomposition can be chosen in various ways according to the

different levels of knowledge in a given domain, or to different viewpoints. There are special "bug"- elements that correspond to diagnosable errors, misconceptions or bad-plans.

• **Configuration** of these elements by defining different types of links, like links of precedence, analogy, generality etc. The configuration of these elements can also be done in various ways. It can express:

1) higher-level domain-expert knowledge (e.g. problem solving strategies);

2) domain-dependent pedagogical knowledge (e.g. in what order to teach the elements of knowledge).

The Domain Knowledge Base can contain various types of knowledge organised on different levels (see Figure 2). Since the main unit in representing the domain-knowledge organisation is called a Teaching Operator, this architecture is called a Teaching Operators-Based Instructional Environment (TOBIE).

In the next section we describe the representation of domain knowledge in TOBIE that allows wide applicability, modularity, ease of extending and some psychological pertinence.

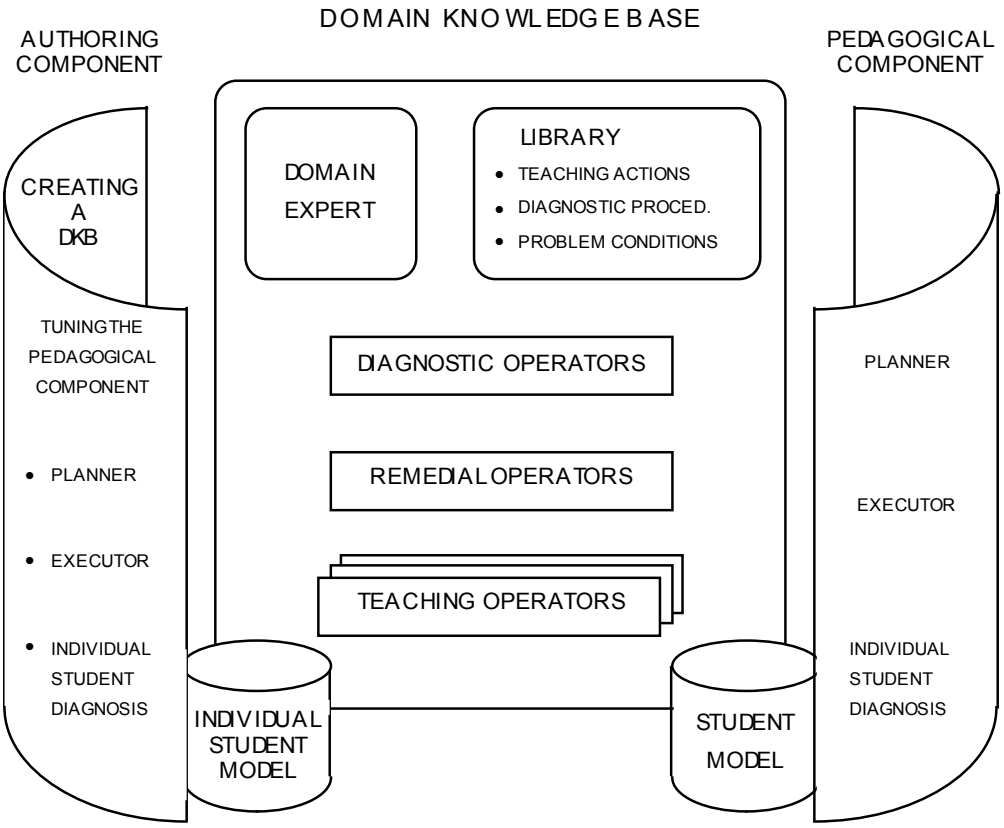

Figure 1: The general ITS -architecture

2.2 Teaching Operators (TOs)

A Teaching Operator is defined as: *TO: SM* ---> *SM* , i.e. a TO on a given level of knowledge organisation is a function over the space of possible states of the student's model. The idea of Teaching Operators is connected with instructional planning (Peachey & McCalla, 1986), (Wenger, 1987). We have generalised this notion (Vassileva, 1990b). The TOs are structures representing in an integrated way the different types of domain-dependent knowledge about:

- the domain-specific teaching actions;
- when it is appropriate to apply them;
- what possible errors could be done.

A TO is a rule-like structure, which conditions and effects are the names of elements on a given level of organisation of domain knowledge. A TO is invoked, if the student model contains its conditions. After a successful execution of a TO, the diagnosed effect-elements are added to the student model.

Every TO is associated with a procedure called "teaching action" that carries out the interactions with the student. It can be either a teaching or a problem-solving step, depending on the level of organisation of the TO.

The context of a teaching action is narrow enough to allow including the knowledge needed for diagnosing the student's answers. Therefore, a list of diagnostic procedures called "passing criteria" is included in the TO's structure. These procedures analyse the student's answer according to the context of the teaching action. There are two groups of procedures. The first group tries to test the correctness of the student's answer and add the effects of the TO to the student model. The procedures from the second group are activated one by one, when no procedure of the first group has succeeded. Their purpose is to diagnose the bug or misconception that caused the error. If any of them succeeds, the corresponding bug-element is added to the student model instead of the TO's effects. Since several TOs can share the same diagnostic procedures and teaching actions, they are stored separately in a "Library". The "passing criteria" and "teaching actions" parts of the TOs contain pointers to the actual procedures.

The success of TOs with different types of teaching actions can give evidence about some individual characteristics of the student. For this reason a list of parameters called "evaluation criteria" is attached to the TO's structure. These parameters describe the teaching action from a pedagogical point of view. They are used by statistical mechanisms in the Pedagogical Component for creating and updating the Individual Student Model.

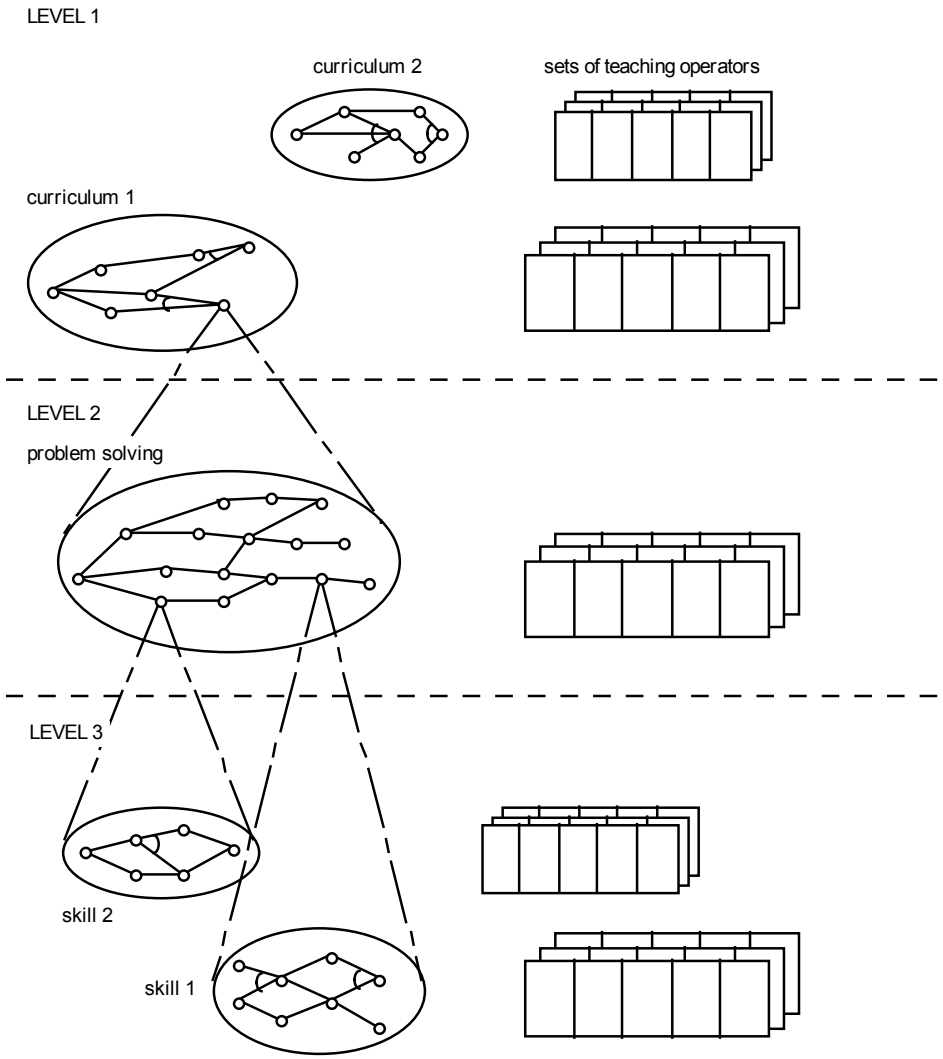

Figure 2: Three levels of knowledge organisation in TOBIE

2.3 Remedial Operators (ROs)

RO: SM ---> SM* , where *SM** denotes the set of possible states of the student model that contain elements, corresponding to bugs, misconceptions (bad plans) or undiagnosed errors. The purpose of ROs is to remove the bug-elements from the student model, therefore the conditions of ROs are bug-elements. The result of a successful execution of a RO is that its conditions are removed from the student model. The action of a RO can be any appropriate teaching action: text, providing a hint or help, letting the student see a possible next step in the solution, explanation or giving a new problem. Of course, the student may not be able to recover - he can make another error. Again a list of diagnostic procedures (passing criteria) checks and if this is the case, a new bug-element is added to the student model. The unused effects-part of a RO can represent additional conditions. It can contain a list of elements, indicating when this particular type of remedial for a given bug will be most appropriate (e.g. intermediate step reached during problem solving; facts from the history of the tutoring session, contained in the list of bugs or in the student models on the same level for the same type of problems).

For example, if the student has not selected an appropriate transformation for an integrand of a given type, several ROs can help him choose the appropriate transformation, some of them requiring additional

conditions, e.g. that the student has already successfully chosen the correct transformation when he obtained an integrand of the same type before. The action of such a RO will not show the appropriate transformation directly, but will make use of the past experience e.g. "Remember what you did before, when you obtained the integrand...".

2.4 Diagnostic Operators (DOs)

DO: SM_{L1} *---> SML2*, where *SML1* and *SML2* are the sets of possible states of the model of student knowledge on possibly different levels of knowledge organisation. The purpose of the DOs is to find the signs of bugs, misconceptions, bad-plans by analysing the contents and structure of the student model. The conditions and the effects of a DO can be elements on different levels of knowledge organisation. The passing criteria are procedures that analyse the structure of the "condition-level" model and try to find configurations of elements that are significant for bugs on the "effect-level" student model. The action of a DO can be empty, i.e. it may not involve any particular teaching action. It can also be a procedure that invokes the system with a set of TOs, representing the "effect-level" domain knowledge. On this level the system may try to remove the bug by an appropriate RO. An example of a DO is given in section 4.1.2.

3 Domain Knowledge Representation

The TOBIE-architecture allows a integrated and unified representation of domain-dependent knowledge. "Integrated" means that different types of knowledge, needed by the Pedagogical Component and for modelling both the student's knowledge and his individual characteristics are represented together with operators (TOs, ROs, DOs). "Unified" means that knowledge on different levels of organisation and from different viewpoints within a fixed domain or in various domains is represented by means of operators having the same structure.

3.1 Representing curricular knowledge

Knowledge about the curriculum of the subject is needed by the Pedagogical Component to dynamically plan instruction. It can be easily represented with a set of TOs. The first level of knowledge organisation in our prototype corresponds to the curriculum. The elements of knowledge decomposition represent the methods of integration the student must learn. The TOs encode the order in which these methods should be taught. The teaching actions are explanations, examples and demonstrations of how a given method is performed etc. In the example below the teaching action is solving a sequence of problems of a given type. Several sets of TOs can be created on this level corresponding to different curricula on the subject of integration. Here are some of the elements in the decomposition of domain knowledge - they denote that the student knows:

1L1 - standard integrals;
1L2 - elementary methods for integration;
1L3 - integration of expressions of the type $1/(ax^2+bx+c)^n$;
1L4 - integration of rational functions;
1L5 - integration of rational functions of sin and cos ;
1L6 - integration of Abel's functions etc.
An example of a TO on this level is given in table 1.

Table 1: A TO on the first level (L1).

Conditions: 1L1, 1L3 **Effects:** 1L4 **Action:** Procedure which creates a sequence of problems and presents them to the student **Passing criteria** Invoke the expert to solve the problem. Compare the answer with the student's answer. If they are identical, the name of the problem is added to the SM_{L1}. If they differ, the system is activated on Level 2 (L2) to trace the way of solving the problem. A SM_{L2} is created. Depending on its state, after returning from L2 the name of the problem is added or not to the SM_{L1}.

Eval.criteria: intelligence: medium or high; concentration: high, confidence: high, motivation: high.

DOs on the first level analyse the structure of the student model after every change and try to find a configuration indicating success. In such a case the list of problem-names will be deleted and 1L4 will be added in the 1-level student model, otherwise instruction continues until the end of the sequence of problems and the TO fails to obtain its effect.

In fact, on this level no actual domain expertise is represented. A system with a set of TOs for only this level will not behave more "intelligently" than a traditional CAI program, containing a script of instruction (but it is generated dynamically during instructional planning, see section 5.1.).

3.2 Representing knowledge for planning the problem solution

Knowledge representation with TOs is also appropriate for representing articulate (glass-box) domain expertise. Since the TOs are productions, this way of representation is applicable for tasks that are decomposable into independently solvable sub-tasks. Typical tasks of that type are those for strategy finding (in planning, games, synthesis of programs), problems whose solutions are unordered sequences of actions (e.g. symbolic integration), problems for creating logical deductions and theorem-proving. Even some problems whose solution can be represented with an ordered sequence of actions could be well presented with TOs, since the search for some sub-sequences can be done in an arbitrary order (e.g. the towers of Hanoi problem). Therefore the class of domains in which problem-solving knowledge can be represented by TOs is very wide.

One of the motives for choosing the subject of elementary integration for the first application of the TOBIE-architecture, was that symbolic integration is a classic example of a domain where heuristic production rules can be used for encoding articulate expertise. All that was needed was to encode some of the heuristic rules, invented by Slagle (Slagle, 1963), with ROs and in this way to create an articulate domain expert for provision of help, advice or for correction of the student's solution path.

The uniform way of knowledge representation allows using the same inference mechanisms at different levels of knowledge organisation. For example, a planning program could serve both as an instructional planner at a curriculum level and as a planner of the problem-solution at a level, corresponding to the decomposition of the main goal into sub-goals. In this way, a given set of TOs and a domain-independent inference mechanism compose a simple articulate domain expert at a given level of organisation.

Building an articulate expert is a non-trivial task. The architecture, however, is entirely modular and allows combining "pieces" of articulate and compiled expertise. The articulate "pieces" of expertise are stored as sets of ROs. A set of ROs may not be able to solve a domain problem entirely (the rules are heuristic, there is no guarantee that the transformation that they recommend is the best choice for the specific problem). They are used when it is necessary to help the student locally, at a fixed stage of the problem solving process. A similar idea for combining articulated with compiled expertise has been implemented in WEST [(Burton & Brown, 1982)], but in a different way. By combining pieces of articulate and compiled expertise a lot of requirements as robustness, reliability, speed, consistency and completeness of the set of TOs etc., which are difficult to ensure for the articulate expert, become not crucial.

In our prototype there is a small articulate expert in integration. A program-analyser of the type of the integrand is developed as an extension to the compiled expert - the muMATH package - to link it with the articulate expert and with the TOs. The articulate expert is based on heuristic rules for choosing an appropriate transformation for a given type of integrand. They are encoded as ROs that are applied only when the student is asking for help. If he continuously asks for help at every stage, he will see the problem solution step by step. Below is given a more detailed example of a RO on the second level of knowledge organisation in the integration tutor.

The second level of organisation corresponds to solving a particular integration problem. Problem solving consists of sequentially applying specific transformations to specific types of integrands. The student can usually solve a problem when he chooses an appropriate transformation at every intermediate stage and performs it correctly. "Appropriate transformation" is defined as one that leads to a simpler expression (for definition of "simpler" see (Slagle, 1963)). A similar way of instruction in integration is performed by the Kimball's Integration Tutor (Kimball, 1982). The elements on the second level denote different types of integrands. ROs encode the appropriate transformations that could be applied to them. Some of the elements in the second level are listed below:

2L1 - a standard type of integrand is obtained;

2L2 - an integrand that can be decomposed into a sum;

2L3 - a function of a linear function of the argument;
2L4 - a rational function is obtained;
2L5 - an integrand of type $1/(ax^2+bx+c)$ is obtained;
2L6 - an integrand of type $1/(ax^2+bx+c)^n$ is obtained.
An example of a RO on the second level is given in table 2:

Table 2 A RO on the second level (L2)

Conditions: 2L4 **Effects:**2L2 **Action:** Choose "decompose to partial fractions". Perform transformation. Display result. **Passing criteria: none** (The student can't make an error since he doesn't do anything.) **Eval.criteria:** intelligence: low; concentration: low; motivation: low; confidence: low

In our opinion the student must have the initiative while solving problems. Therefore, the system has to monitor his work unobtrusively by letting him choose at every stage the appropriate transformation and either asking him to carry it out, or making the domain expert perform it and display the result. In this way the student can either focus his attention on performing transformations or on solution planning. At any stage, however, the student can select a transformation that is not among the "appropriate" ones. In this case no error is encountered, because there is always a possibility that it can lead to a shorter solution in the particular case. The transformation is performed and the resulting type of integrand is added to the student model. In this way the system "follows" the student in his way of solving the problem. An example of two TOs which have different actions and different evaluation criteria is given in table 3.

Table 3: Two examples of TOs on the second level differing with respect to their action and evaluation criteria.

Conditions: type (2I) **Effects:** type (2R) **Action:** Display an integrand of type 2I and a menu of possible transformations. Accept student's choice. Perform the selected transformation and display result. **Passing criteria: Check** if an inapplicable transformation was chosen (these cases are few and can be anticipated). If so, add 2EWT (bug-element) to SM_{L2}. If not, analyse the type of the resulting integrand and add the corresponding element to SM_{L2}. **Eval.criteria:** intelligence: high; concentration: low; motivation: low or medium; confidence: high.

Conditions: type (2I) **Effect:** type (2R)**s** **Action:** Display an integrand of type 2I and a menu of possible transformations. Accept student's choice. Invoke level 3 (L3) to check how the transformation is performed. Create a new SM_{L2}. **Passing criteria:** Depending on the state of SM_{L3} (success or failure), add the corresponding element to SM_{L2}. **Eval.criteria:** intelligence: low; concentration: high; motivation: high; confidence: low.

3.3 Representing knowledge for developing skills

On the third level of domain knowledge organisation there are several sets of TOs. Each of them is supposed to represent knowledge needed for carrying out a given transformation, e.g. integration by parts, decomposition into partial fractions, substitutions, trigonometric substitutions etc. The elements of knowledge organisation correspond to certain sub-skills. For example, in table 4 3U1 means that the student knows how to solve problems of the type "Decompose into partial fractions A/((ax+b)(cx+d)...)", i.e. with linear factors in the denominator. 3U2 means that the student knows how to solve problems containing a quadratic factor in the denominator. The TOs represent the order in which these sub-skills should be taught and the appropriate teaching actions (problems to solve). The type of pedagogical knowledge on the third level is like a small curriculum for teaching one skill. The example in table 4 shows a TO for teaching a certain sub-skill of

decomposing rational functions into partial fractions. All diagnostic procedures in the passing criteria add specific bug-elements to SM_{L3} , if the corresponding bug is recognised.

When the third level is invoked during work on the second level (solving a particular integral), the appropriate set of TOs is chosen according to the type of transformation that has been chosen by the student. During the performance of the transformation, the student can make an error or ask for help. Then an appropriate RO on the third level will be executed. It can advice the student or give him a new problem, more directly connected with some feature that caused the error in the previous problem. The student's work on the second level will be temporarily discontinued to stress on the third level. In this way, instruction on each level is self-contained.

Table 4: An example of a TO on the third level (L3).

Conditions: 3U1 **Effects:** 3U2 **Action:** Show problem of the type (Ax+B)/((ax^2+bx+c)(dx+e)) and accept the student's answer. **Passing criteria:** Check for syntactic errors. Check for factorisation errors. Check for errors in finding the type of the partial fractions. Check for errors in finding the constants in the numerators. **Eval.criteria:** intelligence: low; concentration: low; motivation: low; confidence: low.

4 Student Modelling

Our goal was to find an approach for student modelling that could be applied to various domains. Since we wanted to have a model of the student's individual features and of his domain knowledge, we had to find out a general way for representing information and a general diagnostic technique for any of these different models.

4.1 Modelling the Student's Domain Knowledge

In an ITS-shell, we need a domain-independent way of representation and updating the student model. We made a classification of existing student modelling techniques (Vassileva, 1990a) in order to find a general "kernel", applicable to different domains.

4.1.1 Representation

The model of the student's domain knowledge contains three parts: a dynamic model, a list of bugs and a list of ROs that have been executed (history). The "dynamic model" (in the previous chapters denoted with SM_{Li}) is the most important part of the model, on which the three type of operators (TOs, ROs and DOs) are defined. Teaching and Diagnostic Operators add elements to it and the Remedial Operators delete bug-elements. The "dynamic model" is represented by a list containing the names of knowledge elements, that the system considers as "known" by the student. Our way of student modelling follows the overlay paradigm. However, no actual domain-expertise elements are represented in it, because an element of the domain-knowledge organisation may correspond to one or several of the elements of actual domain expertise, which can be of different type (e.g. procedural, declarative).

In this way, the elements in the student model serve as a "buffer" representing domain expertise in an understandable way for the domain-independent Pedagogical Component. Therefore, it is represented in an unified and homogeneous way for different domains and ways of organisation of domain knowledge. "Homogeneous" means that the student's correct and incorrect knowledge is represented by the elements of domain knowledge organisation. A separate model of the student's domain knowledge is created when instruction is performed on a different level of domain knowledge organisation.

For example, we may have the following dynamic models of the student's knowledge at the same time:

(1L1 1L2 1L3) on the first level means that the student knows the standard integrals, the elementary methods for integration and how to solve expressions of the type $1/(x^2+bx+c)^n$.

(2L4 2L3 (2L1) (2L5) (2L6)) on the second level means that during solving a given problem the student had to integrate first a rational function and he managed to transform it afterwards into a sum of partial fractions of three different types.

(3U2 3E3) on the third level means that the student knows how to decompose into partial fractions $(Ax+B) / (ax^2+bx+c)(dx+e)$, but doesn't know how to cope with expressions with a higher degree in the denominator, like

$$(Ax+B) / (ax^2+bx+c)^n(dx+e) \text{ or } (Ax+B) / (ax^2+bx+c)(dx+e)^n.$$

The list of bug-elements that have ever entered the dynamic model is kept in the history. Its purpose is to provide additional information to help in resolving conflicts during diagnosis. For example, if there are two different elements corresponding to the error the student has made and one of them has already been once in the student's model, this one will be chosen, because students usually tend to repeat the same errors. The ROs applied during work at a given level are kept in the history too, in order to avoid repetition.

4.1.2 Diagnosis

The techniques used in known ITSs for diagnosis of the student's knowledge from his answers show a great diversity. However, all of them are based on comparison of the student's answer with an internal answer either generated by the domain-expert program, or obtained by the application of ad-hoc patterns or mechanisms. The process of selecting the pattern and of matching the results of this comparison with the elements representing the student's knowledge is usually complex and domain-specific and we don't believe that a general matching scheme exists. However, all known techniques have the same kernel:

comparison (student's answer, pattern) --> change(student model)

That is why the diagnosis in TOBIE is organised using this simple kernel.

We believe that in a given context (specific stage of solving a problem or performing a specific transformation) there is a limited number of possible matching schemes (the basic assumption underlying the "model-tracing" paradigm (Anderson & Reiser, 1985) and the patterns for comparison can be defined in advance. They are the results obtained by the domain expert-program, solving either the same problem or a modified problem depending on the type of the error that must be diagnosed. Depending on the context of comparison two types of diagnosis are carried out in TOBIE:

Diagnostic Procedures (Passing criteria)

Diagnosis of errors within the context of one TO (one answer) is undertaken by diagnostic procedures. Each one has the "comparison -> change" kernel and therefore has a standard structure. Each procedure is intended to diagnose a specific type of error and to add one element, corresponding to this error, to the student model. In order to ensure the appropriate context, a set of diagnostic procedures are associated with every TO: the "Passing criteria"-component containing a list of pointers to diagnostic procedures. These procedures are executed in a linear sequence until one of the patterns is triggered by the student's answer. See for example, the passing criteria of the TO on the third level in section 3.3.

A diagnostic procedure may contain also a call to a different level of knowledge organisation (i.e. the passing criteria of the TO on the first level in section 3.1.). Another level is invoked to trace the way the student solves a particular problem in order to find the reason of his erroneous result.

Diagnostic Operators

Diagnosis, however, should be carried out in a wider context than provided by a single TO. This type of diagnosis is realised by DOs. Their purpose and structure have been described in section 2.4. The DOs provide a way of implementing the same kernel (comparison --> change) as any diagnostic procedure. However, they compare a pattern with the structure of the dynamic student model on a given level. The positive result of comparison may indicate a bug on the same or on a different level of knowledge organisation, so it may lead to adding a new element to the student model on any level.

DOs that analyse the dynamic student model on a given level are collected in sets; a set of DOs is provided for every level. The last DO in every set "fires" when the sequence of knowledge elements in the dynamic model becomes too long (longer than a given threshold) and no other DO can match a pattern in it. The first DO "fires" when the time of instruction has exceeded a threshold value. All DOs from the set provided for the given level of knowledge organisation are executed consecutively in two cases:

1) any time when a bug-element enters the student model;
2) repeating at a given period of time during instruction.

For example, one DO on the second level of knowledge organisation finds bad plans called "cycles" in the student's plan. A cycle means that the student has arrived at an integrand of a type, from which he has started several steps before. There are several exceptions, for example:

• the type which closes the circle is 2L1 (standard);

• the student has arrived to the integrand while solving independently integrands in a sum;

• the integrand is of a type that can be solved by integration by parts, leading to a recurrent or other formula for solving the integral, i.e. $1/(x^2+a^2)^n$ and some trigonometric functions.

The DO for finding cycles in the student's plan has a passing criterion, containing a procedure that analyses the structure of the student model. Its conditions-part includes the name of the second level dynamic student model. The effect is adding an element corresponding to "cycle in the student's plan". The action of this DO is empty. The Pedagogical Component will decide, if a corresponding RO or a re-planning of instruction will be executed (this will be explained in section 5).

It is not always possible to decide in advance how to organise diagnosis. Sometimes a diagnosis conflict situation may occur and the Pedagogical Component will have to make a decision. Two diagnosis-conflict situations and the possible pedagogical decisions, that can be taken by the system are shown below:

• What to do when there are differences in the student models at the same level of organisation and for the same type of problems? Usually such differences appear as a result of learning or forgetting, that has happened in the meantime. The trend in the sequence of student models can be significant in this case. A simplified "learning frontier"- interpretation (Goldstein, 1982) is possible via including an appropriate DO on the given level. For example, on the first level we can have the following DO: The type of problems is considered as "mastered" and the corresponding element is added to the student model, when either more than 65% from the given problems were solved correctly or more than 90% from the second half of the sequence of problems (the second level models for these problems show success). It is assumed that learning has occurred although a lot of problems were not solved correctly.

• If there are two or more diagnoses that account for the student's behaviour, which one will be chosen? There are several possibilities:

A. to choose the first diagnosis, explaining the student's behaviour;

B. to choose one of the candidates that has already been in the student model (check the history), because students tend to repeat the same errors;

C. to call another level of organisation in attempt to find additional evidence for one of the hypotheses, e.g. "vertical extension" (Self & Dillenbourg, 1990);

D. to give the student another problem of the same type, e.g. "horizontal extension", (Self & Dillenbourg, 1990);

E. to give the student a special problem that will discriminate between the different hypotheses.

Every specific conflict situation can be treated in many different ways and the decision which one to choose relies more on general pedagogical than on domain-dependent considerations. The teacher can decide which of them will be executed by choosing an appropriate "character" of the Pedagogical Component. This shall be discussed in more details in section 5.2.

4.2 Modelling the Student's Individual Characteristics

Wenger (Wenger, 1987) explains the lack of attempts to model the student's individual characteristics within ITSs with the lack of representational language and diagnostic techniques. The point is to define a set of parameters (domain-independent) that will allow qualitative evaluation of the student's individual characteristics, and methods (domain-specific) to update the values of these parameters according to the student's performance. The only known system, which models the user's individual characteristics is GRUNDY (Rich, 1983), which uses the method of vocabulary analysis. It is questioned (Carroll & McKendree, 1987) whether the user's vocabulary can provide adequate information about his individual features, especially in domains where there is a fixed terminology.

However, we can view instruction as a field of the student's performance apart from the specific subject he is been taught. Then we can consider the psycho-pedagogical type (PP-type) of the teaching actions that are used successfully with the particular student as a "user's vocabulary". If we define the user's vocabulary in this way, we can use vocabulary analysis. The representation of the model is based on evaluation of parameters. Diagnosis makes use of statistics of the different types of "words" used by the student.

How is a "word" defined? Teaching actions can be classified into different clusters with respect to the individual characteristics the student must have, in order to be able to learn from the teaching action. For

example, self-dependence and confidence, preference of abstract or a presentation style, inductive or deductive style of thinking etc. Since diagnosis has to differentiate between teaching actions, the PP-type of a teaching action is indicated in the list of "evaluation criteria" in every TO.

The evaluation criteria in fact are the domain-independent characteristics of the specific teaching action. They allow the system to make conclusions about the domain-independent individual characteristics of the student from his domain-specific actions. Some examples of individual features and the types of actions, by which they can be inferred are listed below:

• Inductive type of learning - TOs and ROs whose action shows textual explanations or solved example problems.

• Deductive type of learning - the student learns better and recovers from errors easier if he is given another appropriate problem to solve.

• Self-confidence - the student prefers to solve problems with a minimum of help or guidance from the system.

This "kernel" set of individual characteristics together with standard psychological characteristics like level of concentration, intelligence, motivation can be used in an arbitrary domain. It could be extended to include additional characteristics important for any specific domain. The model of the individual student characteristics is represented by a set of parameters and their values. Two different methods for diagnosis are used. The method of vocabulary analysis is based on statistics of the values of the evaluation criteria of successfully used TOs and ROs. A psychological pre-test and methods for quantitative evaluation of certain personal characteristics (Wittig, 1986) are used for initialising the values. The programs that make the statistics and analyse the results to update the student model are included in the domain-independent Pedagogical Component (more specifically, in the Executor program).

The individual student model in TOBIE is still rudimentary. It contains only five characteristics which can take discrete values: intelligence, self-confidence, motivation, concentration, preferred type of teaching actions.

5 Pedagogical Decisions

The domain-independent Pedagogical Component contains two sub-components, adjustable by the teacher : a Planner and Executor. Depending on the context of interaction with the student the Pedagogical Component can take global and local pedagogical decisions.

5.1 Global level

The global-level pedagogical decisions are connected with instructional planning. The Planner creates dynamically a plan of instruction for any level of domain-knowledge organisation. It can generate sequences of knowledge elements that lead from the current state of the dynamic student model to the goal state. During the teaching session the Planner can be invoked again to modify or exchange the plan according to the changing situation (goals of instruction, student's knowledge and resources). The planning process is controlled by parameters, adjustable by the teacher. Through these parameters, for example, the way of instructional planning can be chosen - automatic (the plan is created by the system) or not automatic (the plan is created by the teacher). Another parameter allows the teacher to mark certain elements in the knowledge decomposition on the level of planning, which will be paid more attention during instruction. The teacher can participate in the planning of the time-schedule by assigning a given value of another parameter.

After the Planner has found all possible paths to the goal, an Optimizer-program is invoked to choose one of them as a current plan. An optimum path can be, for example, the shortest path, the path that does not contain certain knowledge element etc. The Optimizer uses criteria that can be combined. Once chosen, the plan is followed until an obstacle in the student model or in the environment appears. In this case we say that a local event occurs and a local-level decision has to be taken.

5.2 Local level

The local-level pedagogical decisions are taken during the execution of a plan. The Executor is a program that chooses TOs that, when applied to the dynamic student model will carry it to the goal-state according to the plan. If there are several TOs that can do a desired transformation of the student model, the one whose evaluation criteria match best the individual student model is chosen.

The Executor has to take decisions in case of unexpected situations that arise during the following of the plan. Three types of these situations can be defined:

1) The student:

a. asks for help, or

b. an error or is diagnosed by a diagnostic procedure, or

c. a DO finds a misconception, explaining a sequence of bugs.

In all these cases the current TO cannot achieve its effects. Therefore, the conditions for some of the next TOs in the plan will not be present in the student model and at a given stage it will be impossible to continue following the plan. Since a bug-element enters the student model, conditions for execution of a RO appear.

2) The current TO or a DO makes a call of the system on another level of organisation of domain knowledge which is more appropriate for teaching the student or for diagnosing a specific error.

3) No applicable TO can be found.

The decisions that the Executor has to make in situations 1) and 2) can be generalised. We define two types of reactions: **opportunistic** and **plan-based**. Within an opportunistic reaction the system acts so, as to serve the needs of the moment and within a plan-based reaction the system reconsiders the plan. Whether or not it is necessary to change the plan depends on the situation; the Planner will be instructed by pointing out which of the optimising criteria will be given the highest weight. An opportunistic reaction for situations of type 1) is to execute an appropriate RO for the bug, misconception or call for help immediately. In case of an situation of type 2) the system executes the TO's call for teaching on another level of domain knowledge organisation. In case 3) no opportunistic reaction is possible.

A plan-based reaction will cause invoking the Planner to re-plan instruction with a given optimisation criterion, specific for the situation and the individual student. Re-planning does not necessarily mean that the current plan will be abandoned. The resources, goals and the student model will be reconsidered and the optimal way to continue will be chosen. For example, in case of situation 1)-b, a possible plan-based reaction is to keep on following the same plan, without letting the student know he has made an error, if only situations of this kind are occurring. This "keeping silent" will end when either a 1)-c happens (DO finds the misconception underlying the sequence of bugs, or the time-limit is exhausted) or when the student asks for help (situation 1)-a) or situations of type 2) or 3) occur.

The plan-based approach gives rise to situations where a number of alternative Pedagogical decisions are possible. Consider the following case: The current local situation of type 1) is due to "firing" the last DO on the given level (see section 4.1.2.). Let us suppose now that the opportunistic-type of reaction is chosen. In this case the bugs in the student model will be treated independently with ROs. So, there are several errors (bugs, bad-plans, misconceptions) in the same time in the student model on one level or in models on different levels. In what order should the ROs be executed? For example, let us suppose that the student has chosen several times inappropriate transformations during solving an integration problem. Which of them has to be corrected first?

If the student is decisive, self- confident, it might be better to start with the one that first entered the model. If he is indecisive and shy, may be it will be wrong to discourage him by showing that his way was wrong from the beginning.

The same question arises when two or more bugs are diagnosed on different levels of organisation simultaneously. Is it good, for example, to start with executing a RO for a bug on the second level when there are also "performance-bugs" on the third level? Isn't it better to execute ROs for the performance bugs, to show the student that even if he had performed correctly the transformations, he wouldn't have obtained the solution?

To decide in this case, a human-teacher should know more about the student's individual features, motivation, need for criticism etc. We don't believe that there is a single correct answer to these questions. That is why we decided to create different "characters" of the Executor and ask the teacher to choose one of them before the beginning of the teaching session. We define "character" of the Executor as follows: The character is a set of rules for choosing a type of reaction: opportunistic or plan-based in case of unexpected situation. These rules operate on information coming from:

A. the model of the student's domain knowledge (including the record of his bugs and of the ROs used);

B. the individual student model;
C. the type of the situation;
D. the resources available (time, equipment, memory).

The rules in case of identical data coming from these sources of information, can be radically different in the different characters. For example, *Character 1* can contain the following rule:

Rule 1: If the student is concentrated, consistent and self-confident and there is already a bug-element in the student's model on this level about which the system has chosen not to tell the student and the student is asking for help now, then choose an opportunistic reaction (a RO for the bug will be executed).

Since the student is self-confident, concentrated and consistent, the Executor assumes that the reason of his getting stuck and asking for help is the error he has made before and helps him by pointing it out to the student.

The rule for the same situation in *Character 2* has a different effect; a plan-based reaction is chosen. The system will keep quite and discard the student's call for help. The Executor supposes that the student is following a logical plan of solving the problem and wishes to make the student follow it as long as possible. The fact that the student is self-confident shows that it will be more useful to criticise the whole plan when it turns out to be fruitless (when a DO finds a bad plan).

It is possible to define explicit domain-independent rules that interpret different combinations of these data-sources into either opportunistic or plan-based reactions. At present, six characters are defined. We intend to observe the decisions taken by the system with different characters and different individual student models. We believe this will help to find the significant individual student's features for taking pedagogical decisions.

6 Implementation issues

The integration tutor was implemented on an IBM PC/AT under the DOS 3.30 operation system. The languages used were PASCAL and muSIMP. The system is equipped with an authoring component which allows the teacher to create a problem base and to choose a character of the pedagogical component. The authoring component invokes the domain expert to analyse every new problem in the problem base and to create a set of TOs, ROs and DOs for it. The automatic creating of the sets of operators was not our purpose initially; it can not be done in an arbitrary domain. However, it turned out to be possible for the narrow domain of integration and for one type of teaching action (problem solving). The response time is satisfactory.

Implementing the three different levels of organisation of domain knowledge was, in fact, equivalent to implementing three ITSs in different domains, since the nature of the tasks, the teaching strategies (incl. managing the initiative in the dialogue) and the criteria for success are fundamentally different. However, the unified way of knowledge representation, of student modelling and taking pedagogical decisions provided by the architecture, allows a rapid prototyping. The problem of building a whole system is reduced to the problem of finding an appropriate way of defining the elements of knowledge organisation for the particular domain and level. In general, this is a non-trivial problem of knowledge engineering, however, there have been already methods proposed for knowledge structuring (Grazotto et. al, 1990).

7 Directions of future work

There are several different directions in which this project may be further developed. Some of them are:

1) An object- oriented representation of TOs. This will allow representing explicitly different types of links between the elements of knowledge organisation, e.g. precedence, analogy, generality. At present a given set of TOs encodes only one type of link.

2) Elaborating pedagogical expertise; defining new rules and characters, new situations in which pedagogical decisions are needed and carrying out experiments with real students;

3) Applying the existing architecture to different domains, e.g. chemistry, language learning and development of commercial systems and comparing the results of teaching students with the results obtained by applying other systems. Evaluating the effectiveness of TOBIE's architecture, i.e. how much human effort is saved by applying it in comparison with other approaches for building intelligent tutors is needed in order to justify the label "ITS-shell".

References

Anderson J.R.and Reiser, B.J. (1985) The LISP tutor. *Byte*. 10(4): 159-175.

Burton R., Brown J.S. (1982) An Investigation of Computer Coaching for Informal Learning Activities, In Sleeman, D. and Brown, J.S. (Eds.) *Intelligent Tutoring Systems*, New York: Academic Press: 79-98.

Carroll J.M., McKendree J. (1987) Interface design issues for advice-giving expert systems. *Communications of the ACM*. 30 (1).

Garzotto F., Paolini P., Schwabe D. (1990) HDM - A Model-Based Approach to Hypertext Application Design, Report 90-075, Departimento di Electronica, Politecnico di Milano.

Goldstein I.P. (1982) The Genetic Graph: a Representation for the Evolution of Procedural Knowledge. In Sleeman, D. and Brown, J.S. (Eds.) *Intelligent Tutoring Systems*, New York: Academic Press: 51-78.

Kimball R. (1982). A Self-Improving Tutor for Symbolic Integration. In Sleeman, D. and Brown, J.S. (Eds.) *Intelligent Tutoring Systems*, New York: Academic Press: 283- 308.

Major N., Reichgelt H. (1992) COCA: A Shell for Intelligent Tutoring Systems. *Proceedings ITS'92, LNCS*, Springer: Berlin - Heidelberg: 523 - 506.

Murray T., Woolf B. (1992) Tools for Teacher Participation in ITS Design. *Proceedings ITS'92, LNCS*, Springer: Berlin - Heidelberg: 593 - 600.

Nicaud J.-F. (1992) Reference network: A Generic Model for Intelligent Tutoring Systems. *Proceedings ITS'92, LNCS*, Springer: Berlin - Heidelberg: 351 - 359.

Peachey D & McCalla G. (1986) Using Planning Techniques in Intelligent Tutoring Systems. *Int.J.Man-Machine Studies*. 24: 77-98.

Rich E. (1983). Users are Individuals: Individualizing User Models. *Int. J. Man-Machine Studies*.18 : 199-214.

Self J.and Dillenbourg P. (1990). A Framework for Student Modelling. Lancaster University AI Group Technical Report No 48.

Slagle, J.R. (1963). A Heuristic Program That Solves Symbolic Integration Problems in Freshman' Calculus. *Journal of the ACM*. 10: 507-520.

Vassileva J. (1990). A Classification and Synthesis of Student Modelling Techniques in Intelligent Computer Assisted Instruction. *Proceedings of the 1990 International Conference on CAL* , *LNCS No.458*. Berlin: Springer :202-213.

Vassileva J. (1990). An Architecture and Methodology for Creating a Domain-Independent Plan-Based Intelligent Tutoring System. *Educational and TrainingTechnology International.* 27 (4): 386-397.

Wenger E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos: Morgan Kaufmann Publishers, Inc.

Wittig A. (1986) *Psychology of learning*. Shaum's Outline Series. McGraw Hill.

Acknowledgements

I am very grateful to Boiko Dimchev, Dr.Jana Madjarova and Dr.Roumen Radev from the Institute of Mathematics, Bulgarian Academy of Sciences, who participated in TOBIE's design and implementation. This work has been supported by Project I-406 with the Bulgarian Ministry of Science and Higher Education.