Dynamic Courseware Generation: at the Cross Point of CAL, ITS and Authoring

Julita Vassileva

Institut für Technische Informatik, Universität der Bundeswehr München
85577 Neubiberg, Germany
E-mail: jiv@informatik.unibw-muenchen.de

Abstract
This paper presents a further development of our architecture for dynamic courseware generation which allows explicit representation of teaching expertise. The instructional course is generated automatically for a given teaching goal and can be dynamically changed according to specified teaching rules to suit better the student's individual progress and preferences. The architecture of the system is based on explicit representation of the structure of the domain concepts and relationships (what has to be taught) and the instructional tasks and methods (how to teach). The separate representation of the actual teaching materials allows better flexibility and individualisation, as well as easier updating and re-use of ready CAL materials. In this way our approach provides an alternative to traditional CAL-authoring. An implementation of this approach in a simple engineering domain has been done and an attempt for evaluation of the advantages for authoring is presented.

1 Introduction

Only recently the need of bringing together the fields of CAL and Intelligent Tutoring Systems (ITS) has been recognised (Larkin & Chabay, 1992) and there have been attempts for "intellectualizing" CAL. One possible approach is to start from a set of teaching primitives defined at different level of generality to manage the flow of instruction in a flexible way. Schemes for controlling the dialogue and presentation of teaching materials have been introduced borrowing from the formalisms for representing natural language dialogues, for example augmented transition networks, like (Woolf, 1987) and (Murray, 1992) or by taking a task-based perspective - defining instructional task hierarchies and modeling instruction as planning a sequence of tasks (Van Marcke, 1991). Several approaches take the other direction - of "de-intellectualizing" ITS - applying ITS-shell architectures (Elsom-Cook & O'Malley, 1989), (Wentland et. al, 1991), (Brussilovsky, 1992). Our approach for Dynamic Courseware Generation (DCG) (Vassileva, 1992) falls into this stream. It is based on an ITS-shell architecture (Vassileva, 1990), whose main idea is global planning of the content of instruction (Peachey & McCalla, 1986). Based on a separate explicit representation of the concept structure of the domain and a library of teaching materials, the system dynamically generates instructional courses. The course-plan is created individually for a given student with a given teaching goal; the plan is substantiated with teaching materials and can be changed dynamically according to the changing learning needs of the student. The main advantage of this approach is that it allows automatically building flexible CAL courses which is impossible within the traditional CAL concept of courseware. Because of the separation between the structure of knowledge from its presentation, it supports authoring by re-using existing CAL materials. By organising the concept structure so as to distinguish between different aspects, (e.g. structure and functioning) it is possible to use different discourse strategies for generating plans with meaningful contents. The partially generic pedagogical knowledge, which is explicitly represented by means of instructional task hierarchies, like proposed by Van Marcke (1991), and teaching rules provides for ensuring pedagogical consistency of the automatically generated courses. In this paper we describe an application of our architecture for dynamic courseware generation to an engineering domain. A

[1] Published in Proceedings ICCE'95 -- International Conference on Computers in Education, Singapore, 5-8 December 1995, pp. 290-297

prototype of this application has been developed for teaching with different goals the knowledge about a simple electrical device.

2 Architecture for Dynamic Courseware Generation

A Dynamic Courseware Generator (DCG) implements a combination of an ITS-shell architecture (Vassileva, 1990) which dynamically generates a course plan with a given goal and the GTE architecture (Van Marcke, 1991) which by means of a set instructional task hierarchies and methods decides how to carry out the plan in an optimal for the student way according to a set of teaching rules. The main feature of the DCG architecture is the definition of different levels in the domain knowledge representation. It allows separating the more constant concept-structure from the structure of teaching goals, that depends on the particular teaching session and student. This separation allows the teacher to define explicitly how she wants a goal-concept to be taught; what types of links to related concepts to be followed. The architecture of the system is shown in Figure 1. It will be discussed in the next sections.

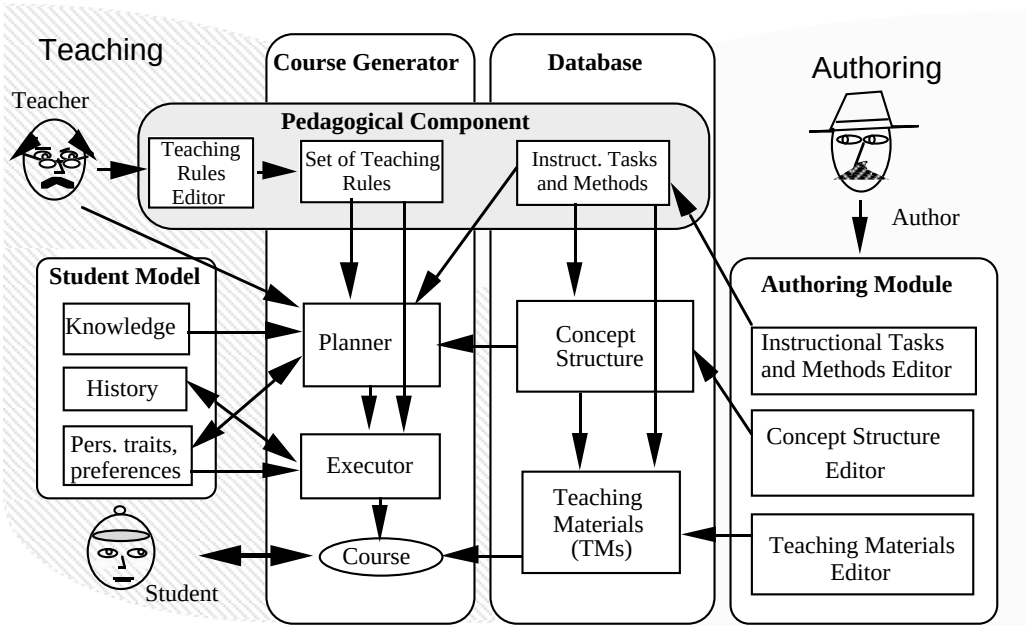

Figure 1: The DCG architecture

2.1 The Data Base

The subject knowledge is contained in the Database Component. It contains two parts:

• The **Teaching Materials** (TMs) contains presentation- and testing-units that carry out the communication with the student. They are focused on a given concept or relationship. Different types of TMs can have different pedagogical characteristics. For example, one can distinguish among an introduction to a concept, a motivating problem, an explanation, help, exercise, or test. In this sense TMs are equivalent to the "instructional primitives" in GTE (Van Marcke, 1991). TMs that carry out a dialogue with the student, like exercises and tests, are represented with a set of smaller units providing a pre-stored correct answer to the exercise/ test, a hint or help, explanation, eventually intermediate stages of solving the problem etc. The TMs are also classified with respect to the media they use, i.e. textual, graphical image, animation or video etc.

• The **Concept Structure** contains the structure of the subject knowledge that is going to be taught. It is represented as an AND-OR graph with nodes, corresponding to the elements of knowledge (concepts) and links, corresponding to the possible relationships between them. There can be different semantic relations between two concepts. For example, aggregation, generalisation, analogy, implication. Every node from the concept structure has an associated set of TMs with different pedagogical type and media. The Concept Structure is used for creating a plan of the course-contents (a subgraph of concept structure) to obtain a given teaching goal. During the course execution TMs are selected by different teaching tasks to teach the concepts of the plan.

2.2 The Student Model

The Student Model contains three parts: a model of student knowledge (an probabilistic overlay with the concepts and relations that have been taught (Diessel et. al., 1993) and (Villano, 1992)), a history (the instructional tasks / methods and the TMs that have been used) and a model of the student personal traits and preferences (contains two lists of variables with their values, the first one denoting psychological features like confidence, motivation, concentration, attention, intelligence, etc. and the other one denoting the preferred types of media).

3 Pedagogical Component

This component contains two parts each of which has a generic kernel and can be expanded with subject specific knowledge (tasks, methods and rules).

3.1 Instructional Tasks and Methods

This is a part of the Pedagogical Component which contains a representation of instructional tasks and their decomposition into sub-tasks by means of different instructional methods, like in GTE (Van Marcke, 1991). The so defined task-structures can be represented with AND-OR graphs similar to the Concept Structure. For example, figure 2 represents the generic task "Give exercise" (adapted from Van Marcke, 1991). The sub-task "Remedy" can be decomposed in different ways according to different methods (shown in figure 2 with different types of lines). The purpose of the instructional task-hierarchies is to allow local planning of the sequence of TMs focused on one given concept from the plan. The tasks and methods can be generic, but as described in (Van Marcke, 1991), the deeper the sub-tasks are in the task-decomposition hierarchy, the more subject-dependent they become. A special editor is provided in the Authoring Module so that the pre-defined set of generic instructional tasks and methods in the system could be extended with subject-specific ones.

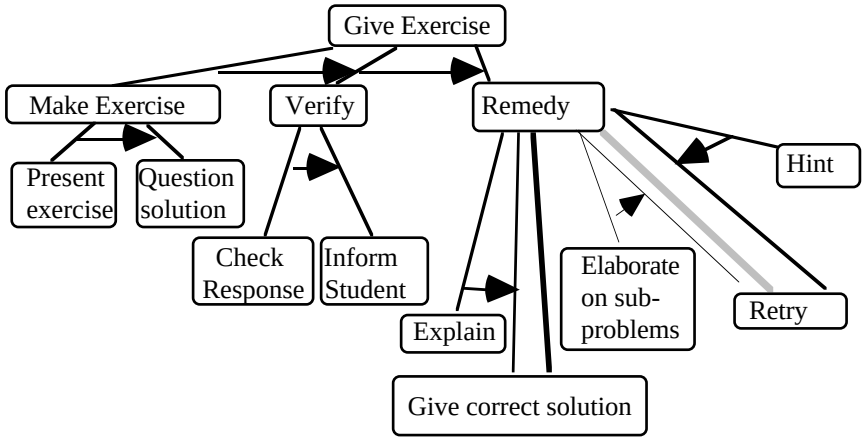

Figure 2: An example of a task hierarchy

3.2 The Set of Teaching Rules

The teaching rules manage the selection and describe different discourse and teaching strategies, manage the selection of instructional (task-decomposition) methods and the selection of TMs. They determine the goal of planning and the plan-selection criteria. Most of the strategy-, method- and task-selection teaching rules are generic. The rules defining a specific discourse strategy, however, are usually subject-specific. The teaching rules take into account data from the student model (student's knowledge and personal traits) as well as external factors like time. Our set of teaching rules has been developed after an analysis of didactic literature (Bohnert, 1995).

- **Discourse rules**

These rules manage the plan selection at the concept level by establishing criteria, how to select the plan when several alternatives are possible (for example, according to what type of semantic links to plan, when to allow switching to follow a different type of link etc.) and how to follow the plan. They also define which

aspects to cover, if the concept structure is organized according to different aspects, and in which way to combine them. One discourse rule, for example, states that in case that the student is intelligent, top-down following of the plan is appropriate with respect to the abstraction links (from general to specific), while for not very intelligent students - bottom-up. If re-planning on the concept level is needed, the discourse rules select whether it will be local replanning or a global change of the plan. If the concepts about a technical device are organized according to, say, functional and structural aspects, one discourse rule states that is better for not-knowledgeable students to use the functional aspect for the main plan allowing jumps to explain concepts from the structure, while for knowledgeable students it is more appropriate to teach from the structural aspect (Paris, 1993).

- **Strategy-selection rules**

These rules define how to select the teaching strategy before the execution of the plan starts. The teaching strategy defines the general principles of teaching, for example, who has the initiative in deciding what to do next - the student or the system. We distinguish between two main types of teaching strategies: **"structured"** and **"unstructured"**. A "structured" strategy means that the initiative is in the hands of the system: it selects which concept will be taught next and how (i.e. with which instructional task). An "unstructured" strategy leaves the choice of a next concept to the student. A highlighting of the "ready to be learned concepts" those whose prerequisites in the plan are considered as known by the student, as in (Beaumont & Brussilovsky, 1995) can help her navigate in the concept structure. The student can also choose an instructional task and method for the current concept from the graphical representation of the task hierarchies. Following the spirit of the general principles for choosing teaching strategies according to different student's aptitudes (Siegler, 1988), we defined several strategy-selection rules. One for example, is that if the student is motivated and success-driven, an "unstructured" strategy would be appropriate, while if she is unsure and not confident, the "structured" strategy should be preferred.

- **Method-selection rules**

There are usually three to eight alternative task-decomposition methods for each instructional task (Van Marcke, 1991). The method-selection rules take into account the history of the used instructional tasks and TMs and the personal traits and preferences from the student model in order to decide what main instructional task(s) will be selected for the current concept and which task-decomposition method will be chosen. This is done right before planning at the instructional task-level. The method-selection rules solve the problem in GTE with the definition of relative applicability conditions of the task decomposition methods, i.e. how to select among alternative possible methods for a given task. Instead of generating weights in a random way, the use of every method is recommended according to specific rules which take into account factors describing the situation.

Three alternative methods for teaching a concept can be found in pedagogical literature. The **"hierarchical"** method teaches by a sequence of the sub-tasks "introduce", "explain" and "give example", "give exercises" and finally "give a test". The **"advanced organiser"** method performs the same task-decomposition with an additional first sub-task which presents explicitly to the student the current teaching goal and the plan of sub-tasks which are going to be executed, what is expected from achieving the current goal, i.e. what is the importance of learning the current concept for the global goal. The **"basic concept"** method's first sub-task is to present a problem (exercise) whose solution requires knowledge of the goal concept. In case the student can't solve the problem, an introduction to the concept is given, an explanation of its main features, an example, the solution of the initial problem, an exercise, and finally a test.

Following Siegler (1988) we defined a rule asserting that the "basic concept" method has to be preferred for motivated students, the "advanced organiser" method should be selected for not very concentrated students, the hierarchical method - for concentrated ones.

For lower-level subtasks the method-selection rules state that an appropriate method should be selected which decomposes the sub-task further into subtasks involving TMs of a pedagogical type which is preferred by the student. For example, for the task "clarify concept" we can have three alternative methods: "explain by description", "explain by example" and "explain by analogy".

Another method-selection rule states that if in the Database contains TMs for the current concept from all needed types, i.e. if all of the methods can be used for the current goal concept, the student model should be checked to see whether any of these pedagogical types is in the list of the personal preferences and if so, the corresponding method is selected.

- **Teaching Material Selection Rules**

For the current instructional sub-task the teaching rules decide how to select a TM on an appropriate type of media (i.e. text, graphics, animation or video etc.). They take into account the model of the student's preferences in order to select among alternative TMs which have the same pedagogical characteristics. They, however, might give a priority to a certain type of media which is preferred by the Teacher.

3.3 Teaching Rules Editor

The Teaching Rules Editor allows the teacher to define her own teaching strategy-, method- and TM-selection rules. This is done by assigning conditions for the application of the rule (variables from the student model) and effects (the choice). It is clear that the set of teaching rules has an extremely important role for the functioning of the DCG. However, the Teaching Rule Editor itself doesn't solve the problem of creating the rules. How can we get such rules? Three approaches are possible:

• to define them ad-hoc, following some guidelines from existing didactic theories - the current solution. This is comparatively easy, but the disadvantage is that most didactic theories are very general and don't formulate precise rules; interpretation is needed in order to arrive at concrete rules guiding action in a specific situation and this interpretation might be not correct.

• to interview teachers or to ask them to implement the rules directly themselves. This, however, requires that the teachers are able to articulate the factors influencing their decisions which is not often the case. This method could also lead to invalid rules, since there is a lot of evidence that people reflect on their decisions in a different way than they actually take them.

• to analyse protocols of teaching sessions, to identifying cases, generalise them if possible to scripts or apply machine learning techniques to generate decision trees and rules. This approach would probably give most reliable results. However, it is difficult to realise, since it requires advanced machine learning tools and a lot of empirical data.

4 The Course Generator

The Course Generator is the component that creates the course, carries out the interaction with the student and maintains the Student Model. The Course Generator contains the following components:

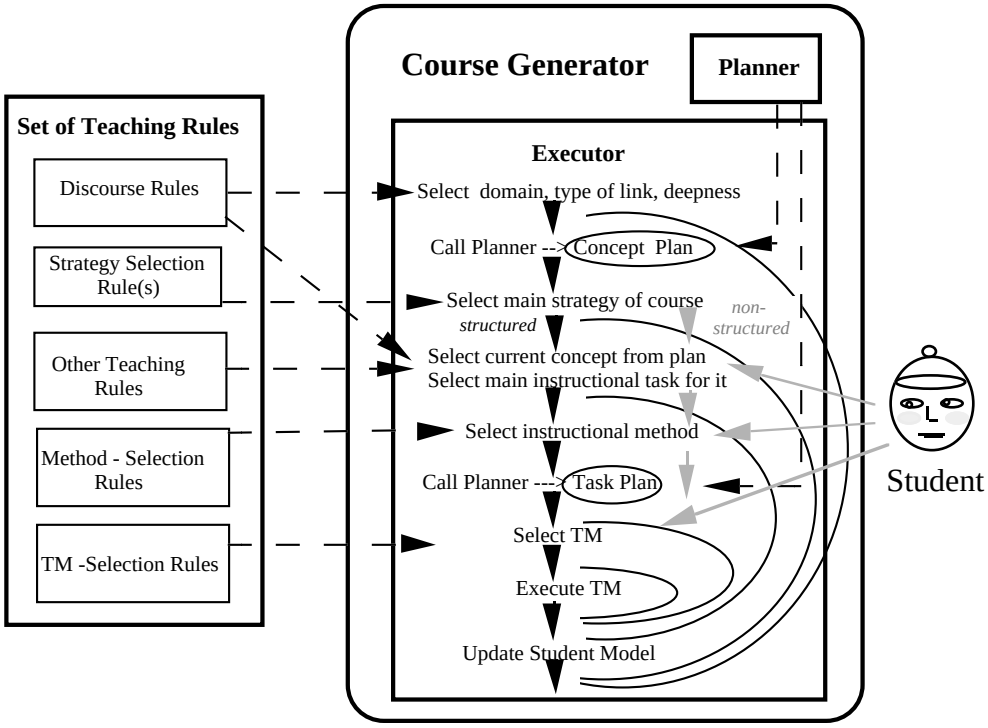

Figure 3: Course generation and execution

4.1 The Course Planner

The course planner is an AND-OR graph planning program which can be invoked with two purposes:

- to generate a plan of the concepts to be covered by the course to achieve a given teaching goal;
- to plan a task-sequence for teaching the current goal-concept.

The teacher invokes the planner and assigns a teaching goal for the course, a given set of domains to be covered by the plan, link types with respect to which to plan and maximal depth of traversing the graph. If there are discourse rules which assign these parameters for a certain teaching goal, the task of the Teacher is only to assign a teaching goal-concept. The Planner is activated to create a course plan. The planning algorithm is a modification of the AO* (Nilsson, 1980). The optimisation function h can be selected so as to achieve different criteria for optimality (i.e. for plan-selection), e.g. the shortest, the plan avoiding certain concept, plan with a certain topology-type etc.. The selection of h is managed by the discourse rules. The solution graph of an AND-OR graph (i.e. the course plan) imposes only a partial ordering on the solution steps. The final ordering of sub-goals is done by certain domain-specific discourse rules or takes place at run time according to the selected teaching strategy.

4.2 The Executor

The Executor consults the discourse Rules and invokes the Planner to create a course plan for achieving the teaching goal. A main teaching strategy (structured or unstructured) for teaching is selected, by checking the strategy-selection rules. If an unstructured strategy is selected, the system lets the student choose the current concept from the plan and an instructional method from the instructional task-hierarchies representation. If a structured strategy is selected, the executor consults the discourse rules again and chooses the current concept or link to be taught, then consults the method-selection rules, selects a method and invokes the Planner again to create a plan of the instructional sub-tasks which are needed to implement the chosen method. Then the TM-selection rules are consulted to select an appropriate TM (see figure 3).

The selected TM is presented according to the teaching sub-task, then the next sub-task from the task plan is executed etc., until a testing TM is executed which checks whether the concept is learned, then the Model of the Student's Knowledge is updated according to the TM's conditional probabilities. With both main strategies, the unstructured and the structured one, it may happen that the student is not able to acquire some concept within the time provided for it. A sign for this is the insufficient knowledge probability of the concept in the student model ("sufficient" is a probability threshold defined by the Author). In this case the executor invokes the Planner to find a new concept plan, bypassing the difficult concept. Our system provides two principal types of re-planning, local plan repair and global re-planning (Vassileva, 1995). Local plan repair means that only the part of the plan related to the current goal will be changed. In this way the system tries to find an alternative way to teach a difficult concept without changing the overall plan. A global replanning means finding an alternative plan for the main teaching goal.

5 The Authoring Component

The Authoring Component consists of a TMs-Editor, a Concept Structure Editor and an Editor for Instructional Tasks and Methods.

The **TMs-Editor** is a tool that allows "wrapping", i.e. presenting in the way that the system can use TMs created by any authoring tool for producing multimedia materials. Ready made CAL materials, courses, videos and graphics can be reused. A unique name is given to every TM and it is associated with one concept or link from the Concept Structure. In order to be included in the Database, it is needed to classify the TM according to its pedagogical type and media and to assign a time allotment for it.

TMs which interact with the student have to be included in the database as a set of "particles": this is a list of pointers resp. to the "body" of the TM (the question, problem etc.), to the correct answer, to a hint, to an explanation, a decomposition of the solution into steps. All these are individually accessible by the instructional sub-tasks. If not all above-mentioned "particles" are present, the corresponding TMs can still be used with the main task, however, some of the task-decomposition methods will not be applicable. At least one test-atom should be created for every node and link, so that the system can judge from the student's success or failure on it whether she knows the associated concept. To provide means for the system to evaluate the degree of knowledge of each of the concepts involved in a given test-atom, the Author has to define the likelihood vectors, i.e. two vectors containing the conditional probabilities of the student's knowledge each of the involved concepts, if she answers correctly and if she answers incorrectly to the given test-atom. Even though there is no guarantee that the probabilities given are adequate, our experience shows that it is not hard for the Author to give approximate estimations of the probabilities.

The **Concept Structure Editor** is a graphical editor which allows developing, extending and modifying the concept structure. It supports creating, deleting and switching between domains; for a selected domain it allows to insert, delete and move, name and re-name nodes on the screen; to insert, delete and connect links; to represent the different semantics of the links with different colours; to view the existing teaching materials in the data-base and to associate them with the nodes and links from the concept structure. This editor provides the possibility to assign conditional probabilities as a special type of links between the concepts, independent of the links denoting inter-concept relations. They have a weight which shows how the knowledge probabilities of the concepts will be internally propagated.

The **Editor for Instructional Tasks and Methods** is similar to the Concept Structure Editor. It allows creating, deleting, modifying of instructional task-structures. Alternative task-decomposition methods are represented by linking the task-nodes with arcs which have different colours, thickness and pattern. Every leaf-node (not decomposable) sub-task is provided with a list of the appropriate pedagogical types of TMs which can be presented. The majority of task hierarchies and decomposition methods are generic and defined in advance. The editor allows the author to define subject-specific instructional tasks.

6 Implementation

The platform chosen for the implementation is IBM PC 486 in a MS-Windows environment. The system is implemented in C++ and OpenScript. ToolBook © Asymetrix is used as an authoring tool for creating the TMs. It allows a very easy creation of TMs with advanced graphics and animation and permits linking photos, and sound- records. At this stage a prototype of the system has been implemented and tested for teaching about electric toasters. Even for such a simple device the concept structure is quite complicated. It was structured according to three aspects — structure, geometry and functions. In the functions-aspect 12 functions (nodes) are connected with time-relations of 3 types: "`before`", "`after`" and "`in parallel`". In the structure-aspect there are 18 nodes organized in a hierarchy connected with links of the type "`is a part of`". There are 5 cross-domain links between the structure- and functional-domains.

The main part of time (one week) an Author spent for "paper and pencil" development of the concept structure. Editing the instructional task-structures took one afternoon and the concept structure - two days. More time - three days - was spent in editing TMs with ToolBook. Having the Data Base ready, the time for generating a course-plan when a Teacher assigns a teaching goal is less than two minutes. The Teacher found reasonable 14 different teaching goals divided in 3 groups: initial acquaintance, montage, maintenance, diagnosis and repair. The length of the generated plans (in terms of nodes and links to be presented) varied in wide interval, depending on the position of the teaching goal in the concept structure and the initial knowledge of the student. The time-duration of a course varied between 10 and 30 minutes, depending on the length of the plan and the duration of the selected tasks with the selected TMs.

In order to evaluate the effort spent for creating an hour of instruction, the time spent for authoring has to be divided by the sum of the durations of all possible courses that can be generated by the system (with all possible teaching goals). If we take 8 hours as an average duration of a working day, 6 hours - for one afternoon and 20 minutes as an average duration of a course, we obtain an approximate ratio of 86 hours of authoring for 5 hours of instruction. This is a favourable result in comparison with other authoring approaches for ITS, since the average time of design and authoring for one hour of intelligent instruction is considered 100 hours. If the Data base allows the generation of many courses with many different goals, the extra-efforts for design and editing the concept structure, and updating and propagation probabilities etc. will be justified. We believe that authoring with our system is far more effective than authoring in the traditional sense. However, we will be able to claim that the system is more effective only after experimenting in a more complicated technical domain. Now we are encoding the Database for a telephone switching device.

7 Conclusions

Dynamic Courseware Generation provides an alternative to the traditional approach for authoring in CAL. Its main advantages are:

- **flexibility in the goals of courses**. By means of a multi-aspect organization of the subject concepts and the possibility to define and use different types of semantic links among them the system can decide how to plan a course for any given goal in an optimal way according to various discourse rules.
- **individualisation of instruction**. Teaching of complex technical systems requires to take into account that the students have different backgrounds, education, level of knowledge, motivation,

intelligence, confidence, independence. By continuous monitoring of the student's behavior and maintaining a simple student model of her knowledge, personal traits and preferences, the DCG is able to find alternative course-plans, teaching methods, and TMs which are dynamically tailored to the student's benefit.

• **possibility to assign and change the teaching strategies** of the system. A human teacher must have a clear metaphor of the mechanism of the system's functioning. She can think of the system as an agent teaching according to a plan like a curriculum by performing instructional tasks according to specific methods. This agent can be "instructed" by means of adding teaching rules in which cases to select which plans, methods and TMs.

• **possibility for easy authoring**, re-use of already developed courseware and of using technical documentation as a basis for developing Teaching Materials. This is obtained by separation of the concept structure from the actual TMs which allows them to be updated and extended directly without changing the structure of the domain.

The course-authoring is shared between the Author (designer of the data-base for a particular domain) and the Teacher. That makes teachers actively involved in the creation of a course without too much efforts and special knowledge of authoring and helps in the overall acceptance of the system. The actual authoring process is shifted to a higher level: to represent explicitly the structure of the concepts and teaching tasks. This is a non-trivial task. However, we believe this is a justified effort in order to obtain a system able to teach in a variety of ways for a variety of goals.

References

Bohnert, A. (1995) Analyse und Entwicklung pädagogischer Lehrstrategien für ein intelligentes tutorielles System, Magisterarbeit, Philosophischen Fakultät der Rheinischen Friedrich-Wilhelms-Universität zu Bonn.

Brussilovsky, P. (1992) A Framework for Intelligent Knowledge Sequencing and Task Sequencing, *Proceedings ITS'92, Lecture Notes in Computer Science No. No 608*, Springer: Berlin-Heidelberg, 499-506.

Beaumont, I. & Brussilovsky, P. (1995) Adaptive Educational Multimedia: from Ideas to Real Systems, *Proceedings of ED-MEDIA'95*, Graz, AACE.

Diessel Th., Lehmann A., Vassileva J. (1994) Individualized Course Generation: A Marriage Between CAL and ICAL. *Computers Educ.*, Vol. 22, No.1/2, 57-64.

Elsom-Cook, M. & O'Malley, C. (1989) Bridging the gap between CAL and ITS. CITE Report 67, I.E.T., The Open University, Great Britain, 1989.

Larkin J. & Chabay R.(eds.) (1992) *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*, Introduction chapter, Lawrence Erlbaum Assoc: Hillsdale, NJ, 1-10.

Murray, T. (1992) Tools for Teacher Participation in ITS Design, *Proceedings ITS'92, Lecture Notes in Computer Science No 608*, Springer: Berlin-Heidelberg, 593-600.

Nilsson, N. (1980) *Principles of Artificial Intelligence*, Morgan Kaufmann Publ., Los Altos, CA.

Paris C., (1993) *User Modeling in Text Generation*, Pinter Publishers: London.

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

Siegler, R. (1988) How Content Knowledge, Strategies and Individual Differences Interact to Produce Strategy Choices, in Schneider & Weinert (eds.) *Interaction among Aptitudes, Strategies and Knowledge in Cognitive Performance*. Springer: New York, 74-88.

Van Marcke, K. (1991) A Generic Task Model for Instruction, in *Proceedings of NATO Advanced Research Workshop on Instructional Design Models for Computer Based Learning Environments*, Twente.

Vassileva J. (1990) An Architecture and Methodology for Creating a Domain-Independent Plan-based Intelligent Tutoring System, *Educational & Training Technologies International*, 27,4, 386-397.

Vassileva J. (1992) Dynamic Courseware Generation within an ITS-shell Architecture. *Proc. ICCAL'92, Lecture Notes in Computer Science No 602*, Springer: Berlin-Heidelberg, 581-591.

Vassileva J. (1995) Reactive Instructional Planning To Support Interacting Teaching Strategies, *Proceedings of AI-ED 95, World Conference on AI and Education*, Washington, AACE.

Villano M. (1992) Probabilistic Student Models: a Bayesian Belief Networks and Knowledge Space Theory, *Proceedings of ITS-92, Lecture Notes in Computer Sciences No 608*, Springer: Berlin-Heidelberg, 491-498.

Wentland, M., Ingold R., Vaniorbeek C., Forte E. (1991) HIPOCAMPE: Towards Learner-Sensitive, Content-Optimized Interactive CAI, *Proceedings CALISCE'91*, Lausanne, EFPL, 233- 240.

Woolf, B. (1987) Theoretical Frontiers in Building a Machine Tutor, in G. Kearsley (ed.) *Artificial Intelligence and Instruction: Applications and Methods*, Addison-Wesley: Reading, 229-267.

Acknowledgements: This work has been partially supported by Project I-408 with the Bulgarian Ministry of Science and Higher Education.