Dynamic Course Generation

Julita Vassileva
Universität der Bundeswehr München, Neubiberg, Germany

This paper presents an approach and architecture for Dynamic Course Generation, based on applying AI planning techniques to a structured representation of the domain knowledge and allowing explicit representation of teaching expertise. An individual course is generated automatically for a given teaching goal and is dynamically adapted at run-time to a student's individual progress and preferences according to the teaching expertise. The separate representation of the teaching materials from the domain structure allows an easier updating and re-use of ready CAL materials. In this way our approach provides an alternative to traditional CAL-authoring. An implementation in a simple engineering domain is described. An evaluation of the benefits of this approach in terms of cost-effectiveness for authoring is shown.

Keywords: authoring, adaptive CAL, discourse strategies, domain ontology, instructional tasks and methods, Intelligent Tutoring Systems, ITS, ITS authoring, tutoring systems, teaching strategies.

1. Introduction

The need of bringing together the fields of CAL and Intelligent Tutoring Systems (ITS) has been recognized (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 levels of generality and to manage their sequencing. Schemes for controlling the dialogue and presentation of teaching materials have been introduced by borrowing from the formalisms for representing natural language dialogues, for example augmented transition networks, like in (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 opposite direction — of "de-intellectualizing" ITS — applying ITS-shell architectures (Elsom Cook & O'Malley, 1989), (Leonhardt et al, 1991), (Linard & Zeiliger, 1995), (Brusilovsky, 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 applying AI planning techniques to determine the content of instruction, which was originally proposed in (Peachey & McCalla, 1986). Based on an explicit representation of the structure of the concepts / topics in the domain and of the 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 at run-time according to the changing learning needs of a particular student. The main advantage of this approach is that it allows automatically building goal-directed adaptive CAL courses, which is impossible within the traditional CAL concept of courseware. Because of the separation between the structure of knowledge and its presentation, it supports an easier authoring: extending and modification of the teaching materials as well as re-using existing CAL materials. By organizing the Domain Structure so as to distinguish between various domain aspects, it is possible to generate content plans representing different viewpoints of the material. The partially generic pedagogical knowledge, which is explicitly represented by means of instructional task hierarchies and teaching rules, as proposed by Van Marcke (1991), provides for ensuring pedagogical consistency of the automatically generated courses. In this paper we describe the architecture and functioning of the DCG, the DCG as an authoring tool and an application of the tool to an engineering domain.

Between: **Differences in:**	**CAL**	**Dynamic Courseware Generator**
Course Definition / Selection	Pre-defined course with a fixed teaching goal	A goal-oriented course is generated at run time
Starting Point	Fixed starting point or a choice of several starting points	Selected with respect to the student's prior knowledge
Presentation Materials	Fixed sequence of Presentation Materials	Dynamically selected according to *teaching strategies* and / or the student's preferences
Way of Following the Course	Pre-defined sequence according to the hypothetical progress of an average student	Automatic course re-planning at run time to match individual differences in knowledge and teaching strategies
Teacher's Role	Excluded (unless she herself is the Author of the course)	Assigns the teaching goal and can manage the teaching strategy by editing the set of teaching rules

Table 1. Differences between CAL and DCG.

2. Characteristics of Dynamic Courseware Generation

A course generated by a DCG looks for the student like a CAL course. However, this course is generated individually for every student to achieve a certain teaching content goal (a topic or concept that has to be learned), and it takes into account the already existing knowledge and preferences of a individual student. In addition, the course is dynamic, i.e. it changes at run-time and adapts to a student's progress, his/her learning style and preferences. The main differences between the two approaches are summarized in Table 1.

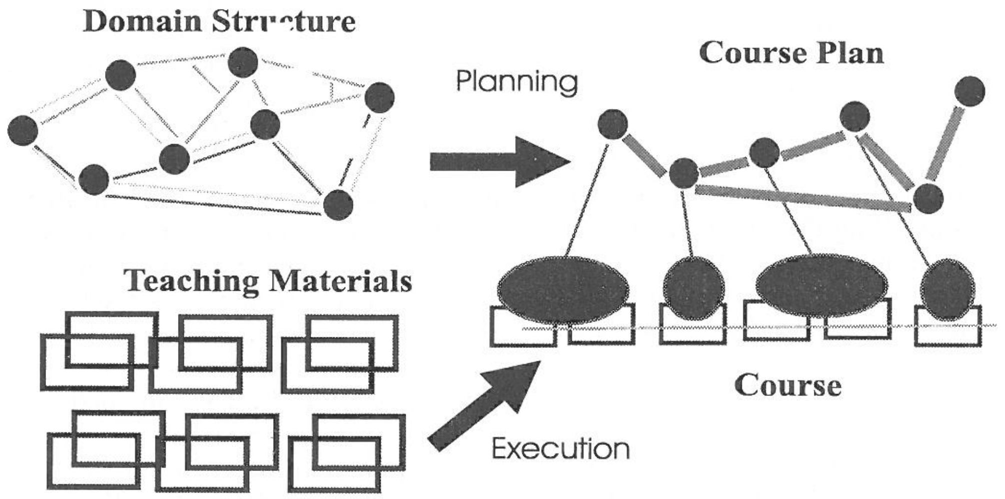


Fig. 1. Content planning in the Domain Structure.

3. Architecture and Functioning

The system implements a combination of the DCG architecture (Vassileva, 1992) which dynamically generates a content plan of the course 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 a way optimal for the student according to a set of teaching rules. The main feature of the proposed architecture is separation between the domain knowledge structure and the presentation materials. The idea of the DCG is to use a classical AI mechanism for planning in an AND/OR-graph representation of the domain knowledge for automatic generation of a content plan of a course (see Figure 1).

The architecture of the system is shown in Figure 2. It will be discussed in the next sections.

3.1. The Domain Data Base

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

• The **Domain Structure** contains the concept/topic structure of the subject knowledge that is going to be taught. It is represented as an AND/OR graph. The nodes represent the elements of knowledge (concepts, topics, rules etc.). If two nodes A and B are connected with a third one, C, by an "AND" link, this means that both nodes A and B have to be developed when following this link from C. Otherwise, they would be considered as alternatives, i.e. there is a choice of nodes to be developed, either A or B. The links represent the possible relationships between the nodes. These relationships can have various semantics. For example, if nodes A and B are connected with node C by an AND-relationship of "aggregation" type, this means that C contains sub-components A and B. If they are connected by an OR-relationship of type "generalization" , this means that C is a general concept with possible instances A or B. There are many other possible semantic relationships, for example, causal relationship, temporal, analogy, simple prerequisite etc.

The simplest way to define a Domain Structure it to use only one possible semantic of links, for example, to link domain concepts /

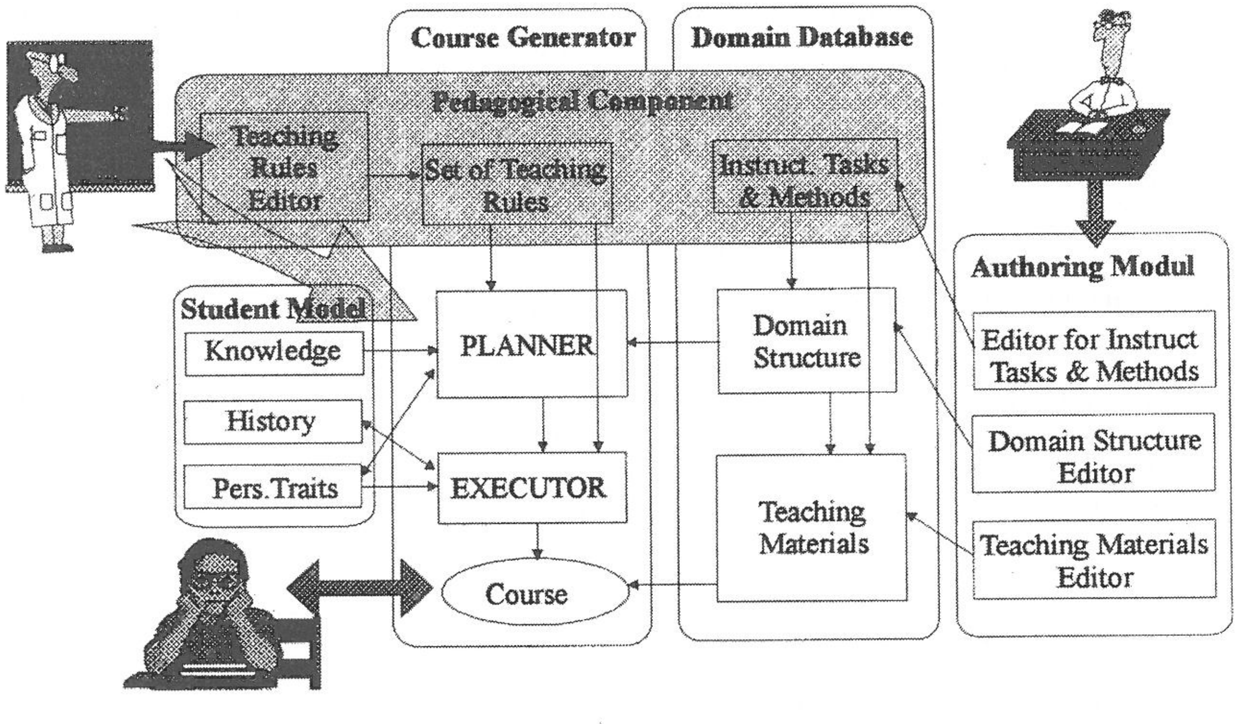


Fig. 2. Architecture of the system

topics with prerequisite links encoding pedagogical following. In this way one obtains a curriculum-like structure which can be used to guide the sequencing of content. This approach has been originally proposed in (Peachey & McCalla, 1986) and can be seen appearing in literature under different names: content model (Van Marcke, 1991), pedagogical structure of the domain (Vassileva, 1990), (Mitrovic, 1996), or pedagogical content knowledge (Leinhardt, 1988), (Calderhead, 1991).

AND/OR graphs have been selected as a representation formalism, since their expressive power is very high — it is equivalent to decomposable production rule systems (Nilsson, 1980) or to STRIPS-like operators. In addition, AND/OR-graphs can be visualized on the screen, which has some psychological advantages for authoring and teaching.

It is possible to organize the concepts / topics in a domain into a set of smaller, possibly interrelated AND/OR-graphs, representing relatively independent sub-areas of the domain, or different "Views". We call such sub-graphs "Aspects".

Every node and every link from the Domain Structure is associated to a set of TMs which represent (teach, explain, exercise and test) this concept. The Domain Structure is used for creating a plan of the course-contents (a subgraph of Domain Structure) to achieve a given teaching goal (a concept). This plan is called a "Content Plan" and the process — "Content Planning" . During the course execution TMs are selected by different instructional tasks to teach the concepts / topics to the student.

• The **Teaching Materials** (TMs) contains presentation-and-testing-units that carry out the communication with the student, i.e. they are in fact what the student sees on the screen. Each TM is focused on a given topic, concept or relationship. The TMs are classified according to their pedagogical function. For example, an introduction, 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 – for example, 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, possibly intermediate stages of solving the problem etc. TMs of "test" type have two additional associated probabilities denoting to what extent a student's correct / incorrect answer means that the student knows/doesn't know the concept(s) which they are supposed to test. The TMs are also classified with respect to the media they use, i.e. textual, graphical image, animation, video etc.

3.2. The Student Model

The Student Model contains three parts: a model of student knowledge (the concepts / topics and relations that have been taught), a history (the instructional tasks / methods and the TMs that have been used) and a model of a student's personal traits and preferences.

The model of **Student Knowledge** is an overlaying over the Domain Structure in the Domain Data Base, containing probabilistic evaluations of the beliefs that a student knows a given concept. Updating of the knowledge probabilities is carried out dynamically as a result of a student's correct/erroneous answer to a test-TMs. This is not an original student modeling technique and we will not discuss its mechanism in detail here. More details can be found in (Diessel et. al., 1993); the original technique is described in (Villano, 1992).

The **History** contains a list of all instructional tasks, methods and subtasks that have been used for every concept while following the plan. It also stores statistics about the success of different instructional task-decomposition methods (see section 3.3.) and statistics of the success of the various media-types of TMs used (text, sound, graphics, animation, video).

The model of a student's **Personal Traits and Preferences** contains two lists of variables with their values. The first contains psychological features like confidence, motivation, concentration and the second one contains the student's preferences for different types of media. The values can take three discrete values (Low, Medium, High) and are assigned by the student at the beginning of the teaching session.

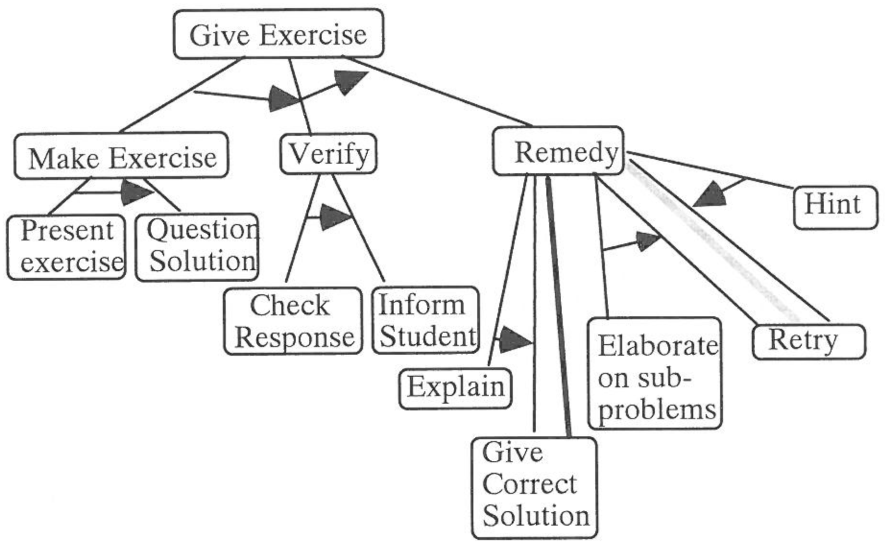


Fig. 3. An example of a task hierarchy.

3.3. Pedagogical Component

This component contains two main parts, a set of Instructional Tasks and Methods and a Set of Teaching Rules, each of which has a generic kernel and can be expanded with subject specific knowledge (see Figure 2). In addition, the Pedagogical Component contains a Teaching Rules Editor which allows the teacher to modify and assign new teaching rules.

Instructional Tasks and Methods

This part of the Pedagogical Component contains a representation of instructional tasks and their decomposition into sub-tasks by means of different instructional methods, similarly to the GTE (Van Marcke, 1991). Like the Domain Structure, instructional task decompositions can be represented with AND/OR graphs, however, here the nodes represent tasks and the links — task-decomposition methods. The AND-links represent links to sub-tasks of a certain task according to a certain task-decomposition method. The OR-links correspond to alternative task-decomposition methods. For example, Figure 3 represents the generic task "Give exercise" which can be decomposed to a sequence of the following sub-tasks: "Make exercise" , "Verify" , "Remedy" (adapted from Van Marcke, 1991). The sub-task "Remedy" can be decomposed in different ways according to different methods (OR-types of links shown in Figure 3 with different types of lines). The purpose of instructional task-hierarchies is to allow planning of the sequence of TMs focused on one given topic / concept from the content plan, i.e. to plan in a pedagogically meaningful way the presentation of a certain topic / concept. Therefore, this plan will be called "Presentation plan" and the process — "Presentation planning". The tasks and methods can be generic, but as suggested 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.

The Set of Teaching Rules

Teaching Rules manage the selection of content and presentation plans. Most of them are generic. The Teaching Rules can be classified into the following categories:

- **Discourse rules**

These rules manage the plan selection when planning or re-planning at the content — (Domain Structure) — level. They establish criteria of how to select the plan when several alternative plans 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. One discourse rule, inspired by (Flammer, 1975), states that the appropriate way of following the plan, in case that the student is intelligent, is deductive (top-down) with respect to abstraction links (from general to specific) or with respect to aggregation links (from whole to parts). For not well doing and not confident students inductive (bottom-up) following the plan is better. If re-planning on the content level is needed, the teaching rules select whether it will be local re-planning or a global change of the plan (see section 3.4 and Figure 6).

- **Strategy-selection rules**

These rules define how to select the teaching strategy before starting the execution of the plan. The teaching strategy defines general principles of teaching, for example, who has the initiative in deciding what to do next — the student or the system. We distinguish 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. Presenting the Domain Structure on the screen and highlighting of the "ready to be learned concepts / topics", i.e. those whose prerequisites in the plan are considered as known by the student, as in (Beaumont & Brusilovsky, 1995), can help navigate in the Domain Structure. The student can also choose an instructional task and method for the current concept from the graphical representation of the task hierarchies. One strategy-selection rule, for example, states that if a student is motivated, an unstructured strategy would be appropriate. If he/she is unsure and not confident, the structured methods should be preferred (Siegler, 1988).

Both the discourse and strategy-selection rules take into account data from the student model (student's knowledge and personal traits) as well as external factors like time.

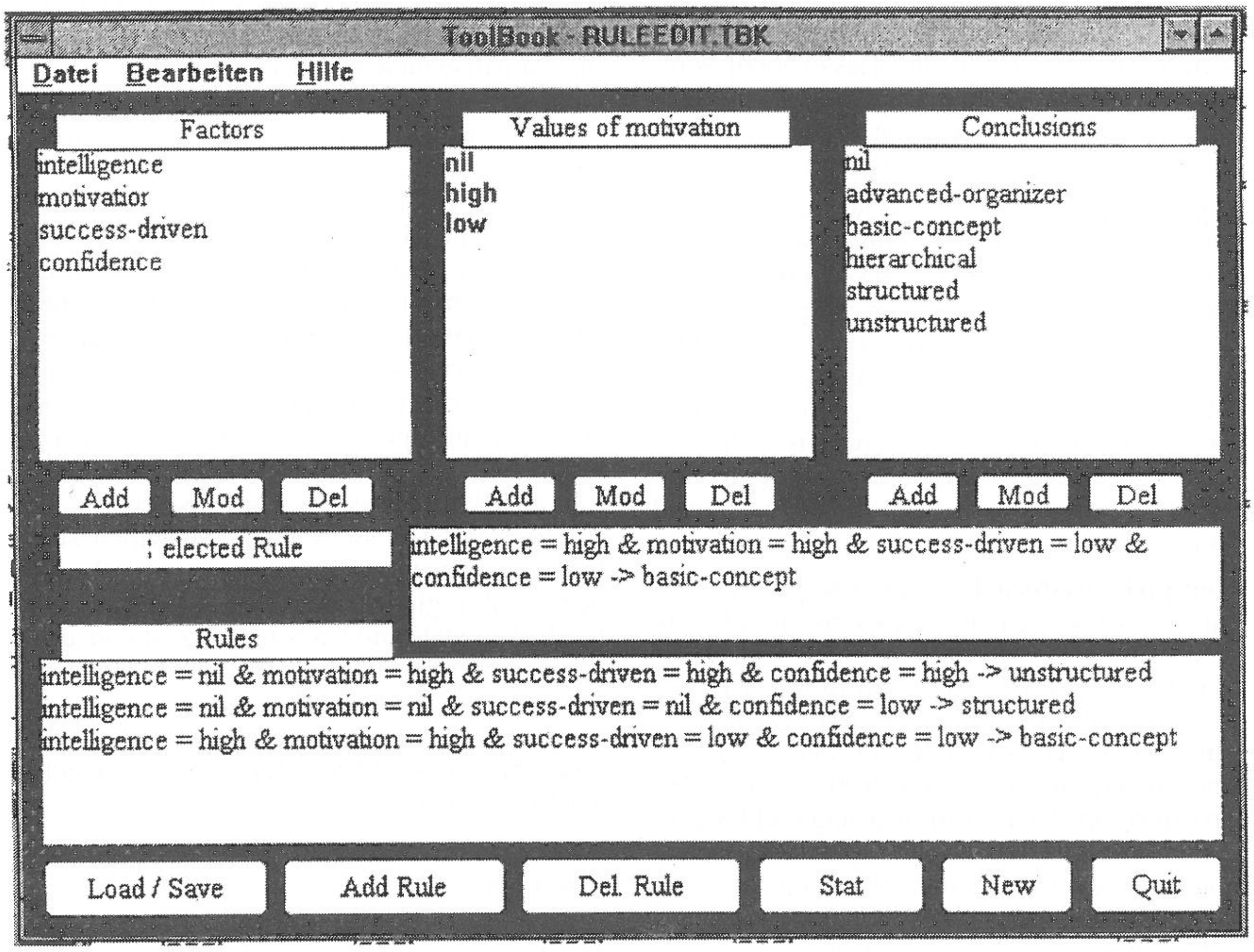
- **Method-selection rules**

There are usually three to eight (Van Marcke, 1991) alternative task-decomposition methods for each instructional task. The method-selection rules take into account the student's History and his / her Personal Traits and Preferences from the Student Model in order to decide what main instructional task will be selected for the current concept and which task-decomposition method will be chosen. This is done at the stage of presentation planning. The method-selection rules solve this problem in GTE with the definition of relative applicability conditions of the task-decomposition methods, i.e. how to select among possible alternative methods for a given task.

In the pedagogical literature (Einsiedler, 1976, pp. 290–293) one can find four methods for teaching a concept, which decompose the main task "teach" into sub-tasks. The "**hierarchical**" method teaches by a sequence of the sub-tasks "introduce" , "explain", "give example" , "give exercises" , and finally "give a test". The "advanced organizer" method performs the same task-decomposition with an additional first sub-task which explicitly presents to the student the current teaching goal and the plan of sub-tasks which are going to be executed, and an objective stating what is expected from achieving the current goal, i.e. what is the importance of learning the current concept for the global goal of the course. The "basic concept" method's first sub-task is to present a problem (exercise) whose solution requires knowledge of the goal concept. A student is not expected to solve the problem, since he / she lacks knowledge related exactly to the concept which is going to be taught. However, the attempt to solve the problem prepares him / her to understand the need for the new concept. Then the new concept is introduced, along with the examples and explanations of how this type of problems is solved, exercises, and finally a test. The "discovery" method involves presenting a motivation problem, analyzing the problem and letting a student solve the problem alone, hoping that he/she will discover the concept in the problem solving.

Following Einsiedler (1976) we defined a set of rules asserting that:

- the "basic concept" method has to be preferred for successful, but not highly motivated and concentrated students,



```
intelligence
motivatior
success-driven
confidence
```

```
nil
high
low
```

```
nil
advanced-organizer
basic-concept
hierarchical
structured
unstructured
```

```
intelligence = high & motivation = high & success-driven = low &
confidence = low -> basic-concept
```

```
intelligence = nil & motivation = high & success-driven = high & confidence = high -> unstructured
intelligence = nil & motivation = nil & success-driven = nil & confidence = low -> structured
intelligence = high & motivation = high & success-driven = low & confidence = low -> basic-concept
```

Fig. 4. The Screen of the Teaching Rules Editor

- the "advanced organizer" method – for successful, but not very concentrated and confident students,
- the "hierarchical" method – for concentrated students, with not very good success so far,
- the "discovery" method – for confident, concentrated students.

• Teaching Material Selection Rules

When the current instructional sub-task is selected and decomposed down to primitive sub-tasks, the teaching material selection rules decide how to select a TM on an appropriate type of media (i.e. text, graphics, animation or video etc.). In order to choose among numerous TMs for a selected primitive sub-task, they take into account the model of a student's preferences which use a type of media, preferred by the student.

Teaching Rules Editor

The Teaching Rules Editor (Figure 4) allows a teacher to define his/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 (decisions of choices).

It is clear that the Set of Teaching Rules plays an important role in functioning of the DCG. However, the Teaching Rule Editor itself doesn't solve the problem of creating the rules. How do we get such rules? Three approaches are possible:

• **theory-based**: to compile them from existing didactic theories — the current solution (Bohnert, 1995). A disadvantage of this approach is that most didactic theories are too general and do not formulate precise rules. A designer / author / teacher needs to interpret general directions described in the theories to obtain some concrete rules that can guide action in a specific situation. This interpretation is always subjective and therefore can be criticized.

• **person-based**: 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 way, different from the way they actually make them (because they are aiming at a "post-mortem" logical justification of their actions).

• **empirically-based**: to analyze protocols of real individual teaching sessions, to identify cases, and apply machine learning techniques to generate decision trees and rules. This approach would probably give more reliable results. However, it is more difficult to realize, since it requires a lot of empirical data. We have developed a machine learning tool for generation of decision trees and rules from descriptions of cases (Horstmann, 1995) and we intend to apply it for generating rules from protocols of teaching sessions once such data is available.

3.4. The Course Generator

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

The Course Planner

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

• to generate a content plan (the concepts / topics to be presented in the course) according to the teaching goal assigned by the teacher;

• to create a presentation plan (a task-sequence) for teaching the current goal-concept.

A teacher calls the Planner for a particular student and assigns a teaching goal for the course, a given set of aspects to be covered by the plan, link types with respect to which to plan and maximal deepness in traversing the graph. If there are discourse rules assigning these parameters for a certain teaching goal (e.g. as in our prototype for the goals "acquaintance", "installation", "maintenance" and "diagnosis"), 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* graph search algorithm (Nilsson, 1980). The optimization function h can be selected so as to achieve different optimality criteria (i.e. for plan-selection), e.g. the shortest plan, a plan avoiding a certain concept, a plan with a certain topology-type etc. Selection of h is managed by discourse rules. The solution graph is an AND-graph which starts from the concepts / topics known by the student (with high knowledge probabilities in the Student Model) and leads to the goal concept / topic. The plan imposes only partial ordering on the solution steps. Final ordering of subgoals is done at run time, by domain-specific discourse rules and according to the selected teaching strategy.

The Executor

The Executor receives the plan from the Planner. A main teaching strategy (structured or unstructured) is selected by checking the strategy-selection rules. If an unstructured strategy has been selected, a student has to choose from a graphical representation of the plan the next concept which he/she wants to be taught and then to select an instructional method which will be used. 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 and selects an instructional method. Then it invokes the Planner to create a plan of the instructional sub-tasks which are needed to implement the chosen method. Finally, the TM-selection rules are consulted to select an appropriate TM (see Figure 5).

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. The Model of the Student's Knowledge

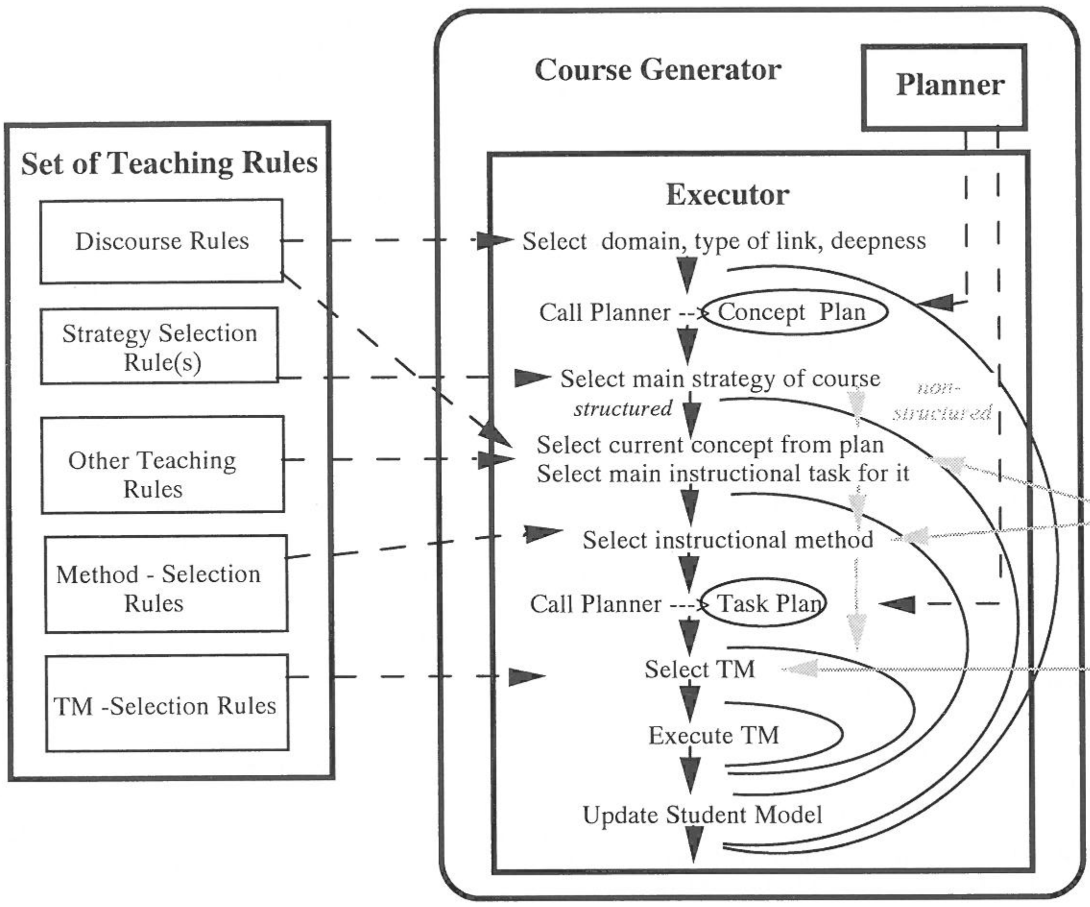

Course Generator
Planner
Set of Teaching Rules
Discourse Rules
Strategy Selection Rule(s)
Other Teaching Rules
Method - Selection Rules
TM -Selection Rules
Executor
Select domain, type of link, deepness
Call Planner --> Concept Plan
Select main strategy of course
structured
non-structured
Select current concept from plan
Select main instructional task for it
Select instructional method
Call Planner ---> Task Plan
Select TM
Execute TM
Update Student Model

Fig. 5. Course generation and execution

is updated according to the TM's conditional probabilities. With both main strategies, the unstructured and the structured one, it may happen that a 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. During teaching there is a button available to the student which allows him/her to change the values of his/her Personal Traits and Preferences in the Student Model. At every selection of instructional method or teaching material for the next concept, the updated state of the Student Model is taken into account.

There are two principal types of re-planning, local plan repair and global re-planning. Local plan repair means that only that part of the plan which is related to the current goal will be changed (see Figure 6). In this way the system tries to find an alternative way to teach a difficult concept without changing the overall plan. A global re-planning means finding an alternative plan for the main teaching goal. The discourse rules define which type of re-planning will be chosen.

3.5. The Authoring Module

The Authoring Module consists of a TMs-Editor, a Domain 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 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

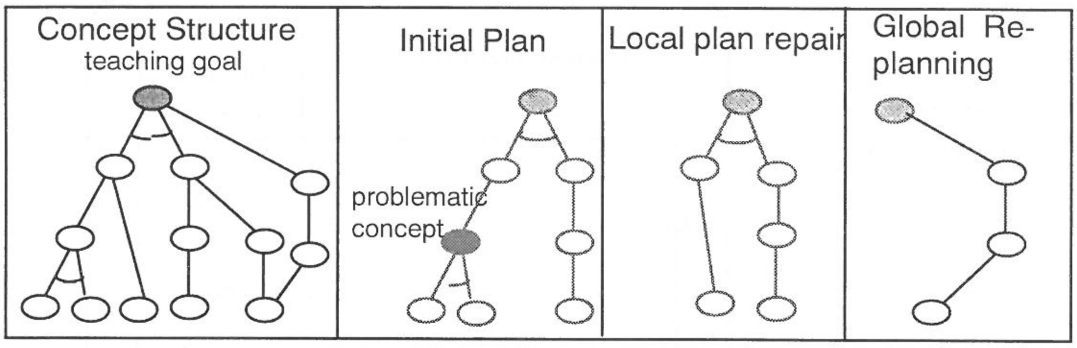


Fig. 6. Two ways of re-planning

from the Domain Structure. In order to be included in the Database, it is needed to classify the TM according to its pedagogical type (what instructional task it implements) and media and to assign a time allotment for it.

TMs interacting with the student have to be included in the database as a set of "particles" : this is a list of pointers respectively 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 instructional sub-tasks. If not all above-mentioned "particles" are present, corresponding TMs can still be used with the main task, but some of the task-decomposition methods will not be applicable. For example, let's assume that in the Data Base for a certain concept A there are two TMs with the pedagogical characteristic "exercise" : the first one is represented as a list of pointers to the following particles:

$Ex_1 \rightarrow$ (*problem_statement, *correct_answer, *hint),

$Ex_2 \rightarrow$ (*problem_statement, *correct_answer, *hint, *recorded_steps_to_solution, *explanation-_of_solution).

If Ex_1 is selected for implementing the task "Exercise" for concept A, only three from the five alternative task decomposition methods of the sub-task "Remedy" are available (see Figure 7). The two other methods would be applicable with Ex_2 because it has the needed particles to implement all of their sub-tasks.

So, in order to enable the system to execute instructional sub-tasks giving a feedback to the student's errors, for every exercise the Author has to create an associated remedial material. This could be a hint, an explanation, a stepwise solution, a leading question etc. They will be presented to the student by instructional tasks in the order prescribed by the task-plan.

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, whether he/she knows the associated concept. To provide means for the system to evaluate the degree of knowledge for each of the concepts / topics addressed by a given test-atom, the Author has to define a likelihood vector containing the probabilities of the student knowing each of the involved concepts / topics, if he/she answers the test-atom correctly.

Even though there is no guarantee that the probabilities given are adequate, we suppose that it is not hard for the Author to give approximate estimations of the probabilities, for example: "If a student answers test-atom A correctly, in 85% of the cases this means that he/she knows concept X and in 90 % — that he/she knows concept Y".

The **Domain Structure Editor** is a graphical editor which allows developing, extending and modifying the Domain Structure. It supports creating, deleting and switching between aspects; for a selected aspect it allows to insert, delete and move, name and re-name nodes on the screen; to insert, delete and connect links; to represent different semantics of the links in different colors; to view the existing teaching materials in the data-base and to associate them with the nodes and links from the Domain Structure.

The **Editor for Instructional Tasks and Methods** is similar to the Domain Structure Editor. It allows creating, deleting, and modifying instructional task-structures. Alternative task-decomposition methods are represented by linking the task-nodes with arcs which have different colors, thickness and pattern. Every leaf-node (not decomposable) sub-task is provided with a list of 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.

4. Implementation of the System in an Engineering Domain

In engineering domains it is important to distinguish between the basic physical concepts (e.g. power, force, mass, velocity, electricity, voltage, current etc.) and the concepts corresponding to certain devices, parts of devices, functions etc. It is also important to distinguish between different types of relations, corresponding to their different semantics (e.g. aggregation, abstraction, analogy, causal, temporal and spatial). A plan of a course is a sub-graph with given properties, e.g. starting with a goal-node, terminating with certain leaf-nodes or known by the student nodes, following links with a certain semantics and meeting certain optimality criteria. Different teaching goals can be achieved by taking different views of the same knowledge. In order to generate a meaningful course, the Planner needs knowledge about the semantics of the links that have to be followed, so it has to be assigned explicitly by the Teacher. This, however, would be difficult, since the Teacher will need to know not only what concepts there are, but also all different types of links between them and how they can be traversed for a specific teaching goal. A way of organizing the Domain Structure is needed to separate different semantic views over the subject. This is where a decomposition of the Domain Structure into smaller relatively independent sub-graphs or "aspects" (see section 3.1.) is particularly useful.

4.1 "Aspects" within the Domain Structure

It is impossible to describe a complex engineering system without considering a structured representation allowing different aspects and levels in the description, (Hewett & Hayes-Roth, 1994). We will use the word "aspect" as a synonym for a "view-point" , since it seems to us a more general notion. Usually technical systems can be well described in three aspects: functional, geometrical and structural.

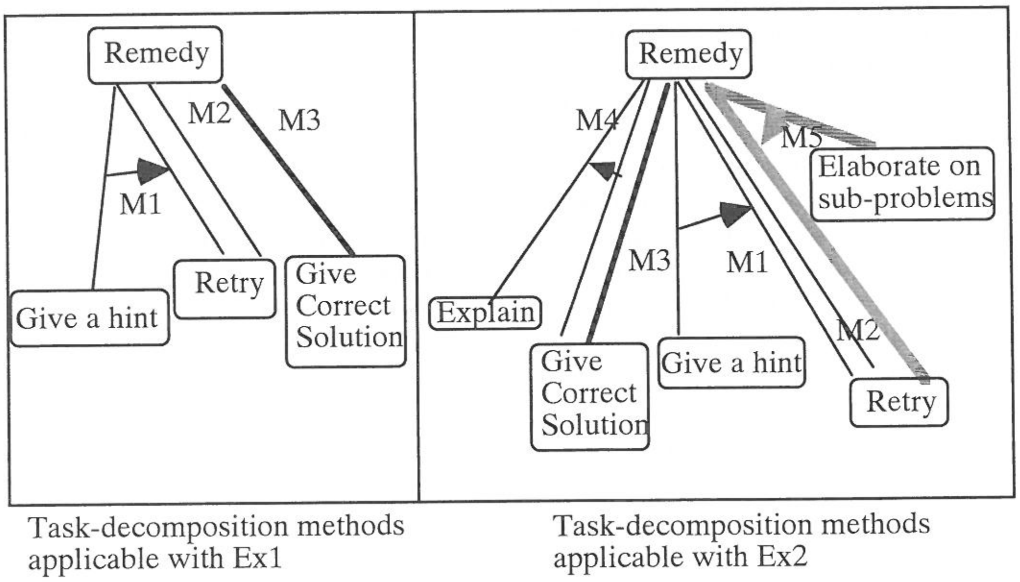

Remedy
M2
M3
M1
Give a hint
Retry
Give Correct Solution
Task-decomposition methods applicable with Ex1
Remedy
M4
M5
Elaborate on sub-problems
M3
M1
Explain
M2
Give Correct Solution
Give a hint
Retry
Task-decomposition methods applicable with Ex2

Fig. 7. Applicability of Different Task Decomposition Methods.

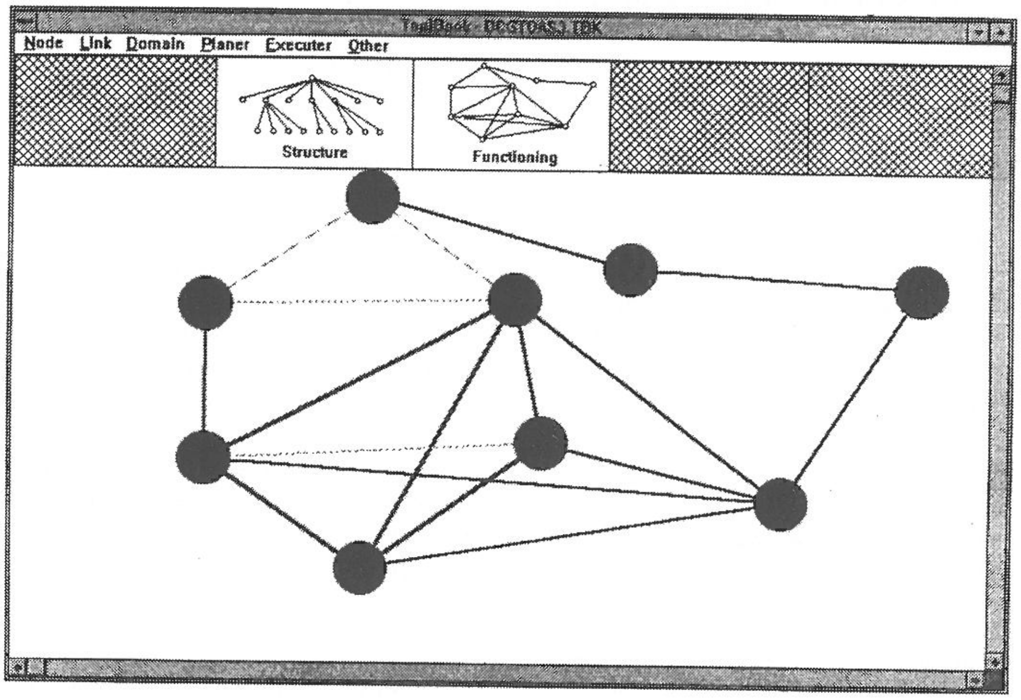


Fig. 8. The Screen of the Domain Structure Editor

In other domains different aspects may be more relevant, for example, costs, behavior etc. A relatively self-contained description of the domain according to any of these aspects should be possible.

Links among the concepts in one aspect may have different semantics (aggregation, abstraction or causal). For example, an aggregation type of links may be used to describe a component hierarchy in the structural aspect, or generalization links — to describe functional hierarchies, or causal links — to show operation flows in the functional aspect. There are also cross-aspect links, which have a subject-specific semantics. For example, a link with the semantics "`is performed by`" ↔ "`performs`" can connect a given function-node with the corresponding structure node(s) that implement(s) the function. As an example, the Domain Structure with two aspects — structural and functional — of a toaster is shown in Figure 8.

4.2. Discourse Rules for Course Planning

It is often enough to consider one isolated aspect for generating a course. For example, if the teacher wants a course on how to assemble a toaster, she should consider the "Structure" aspect and assign the concept "Toaster" as a teaching goal. The Planner generates a set of possible plans for the course in this aspect, according to the main type of link in this aspect with the semantic "aggregation" semantics ("`a part of`" ↔ "`contains`"). The plan includes all components needed to build a toaster. The Discourse Rules determine which one out of all possible plans should be selected for execution and how it should be followed (bottom up or top-down).

Results from the field of automatic text generation indicate that in technical domains, devices are most often described with referrence to two different aspects: structural and functional. Paris (1993) analyzed encyclopaedia texts describing technical devices. She found out that the explanations that were targeted at novices (junior encyclopaedias) usually describe the way of functioning — a "process trace" of the device and include only the necessary minimum of information about the structure components carrying out these functions. On the contrary, texts intended for experts (manuals for technicians) focus mainly on the structure and component details — they use the "constituency schema" (McKeown, 1985). For an individual user whose level of knowledge is somewhere between these two extremes, the description of a device should be composed as a combination of the process trace and the constituency schema.

Our course a planner is able to generate plans which cover more than one aspect because it can plan with respect to different types of links, including cross-dimensional, at the same time. However, in order to keep the description focused, it is needed to select one aspect as a main "back-bone" of the course. Depending on the changes in the student's knowledge and in the environment restrictions, the executor should be able to switch to the other aspect according to certain rules. Paris (1993) shows a way to use an explicit model of the user's knowledge of the domain concepts (which is similar to our model of the student's knowledge) for generating individually tailored explanations.

Our student model, similarly to the one proposed by Paris, contains information regarding the student's knowledge about basic concepts and about other concepts. The decision how to choose an overall schema of the course, in analogy with Paris (1993), is made by using the following discourse rule:

A process trace (i.e. the functional aspect) is selected as a main aspect for planning only in case that all of the following conditions are fulfilled:

1) there is a functional aspect and a corresponding function to the structural concept (the teaching goal), and

2) the student has no knowledge about the goal concept, and

3) no knowledge about any super-ordinate concept with respect to a "generalization" — type of link,

4) the plan in the functional aspect does not involve unknown basic concepts, and

5) the student doesn't know most of the functionally important structural components (related by cross-aspect links to the most interconnected nodes in the function aspect).

If any of these conditions are not fulfilled, a constituency schema (the structural aspect) is chosen as the main one. The rule for choosing an overall schema is encoded in the set of Discourse Rules. It is an example of a domain-dependent rule, in contrast with generic discourse rules examples of which were given in section 3.3. Once an overall schema for the course is selected, a plan is generated for teaching the goal concept in the selected aspect. This can be the structural aspect (in case that the constituency schema is chosen) or the functional aspect (in case that the process trace is chosen).

Combining the two aspects can be done at a specific decision point in the algorithm of the Executor (see Figure 5). This is the point when a new concept is introduced and needs to be described, i.e. at the point "Select current concept from the plan". At this point the system has to decide whether to provide structural or functional information. The decision whether to switch to the other aspect is made by the same discourse rule and one additional discourse rule that checks whether there is still the time needed to perform the switch. During the execution of the initial plan it may become clear that the student model has changed, the selected plan is no longer appropriate and has to be changed locally or even for the whole course. In case of re-planning the course, the Discourse Rules will select a different plan, taking into account the new state of the Student Model.

The described discourse rule is activated for the teaching goal "initial acquaintance". There are other discourse rules for other teaching goals, like "installation" , "maintenance" , " diagnosis".

5. Evaluation

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, videos, and sound-tracks. 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 Domain 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-aspect links between the structure and functional-aspects.

Most of the time (one week) an Author spent for "paper and pencil" development of the Domain Structure. Editing the Instructional Tasks and Methods took one afternoon (there were only 12 rules) and the Domain Structure — two days. More time — three days — was spent in editing TMs with ToolBook. Once the data base with the TMs is ready, the time for automatic generation of a course-plan when a specific teaching goal is assigned is less than a minute. The Teacher found reasonable 14 different teaching goals divided in 3 groups: initial acquaintance, montage, maintenance, diagnosis and repair. The length of generated plans (in terms of nodes and links to be presented) varied in a wide interval, depending on the position of the teaching goal in the Domain Structure and on the initial knowledge of the student. Duration of a course varied between 10 and 30 minutes, depending on the length of the plan and on the duration of selected tasks with selected TMs.

In order to evaluate the effort made for creating an hour's 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 available 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, i.e. the ratio is 17.2 to 1. This is a quite favorable result in comparison with other authoring approaches for IST and even for traditional CAL courseware, since the lowest average time of design and authoring for one hour of intelligent instruction quoted by different authors is around 100 hours. If the Domain Structure allows generation of numerous alternative courses for different goals, extra-efforts for design and editing the Domain Structure are 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. Recently, the DCG has been re-implemented in a more modest version on the WWW (Vassileva & Deters, 1997), and it has been applied as an authoring tool for creating a WWW-based course of lectures on Computer-Based Learning, given by the author of this paper at the Federal Armed Forces University. The authoring effort was approximately the same — 18 hours of authoring for one hour of instruction. Results of the initial tests showed that the DCG on WWW will not make a ,breakthrough" in the paradigm of university teaching, but it can be easily and usefully integrated in the existing organization. The following main applications were outlined:

- *Lecture Support, Distance and Continuous Education.* The courses generated with the DCG can be used as additional learning materials (as an interactive script) supporting lectures given regularly or occasionally at the University. What is specific about our university is that all students are officers, obliged to serve in the Army five years after graduating. Interactive courses on the WWW accompanying the lectures offered at the university would provide an "umbilical cord" between our students and their Alma Mater. It this way they can deepen and refresh their knowledge permanently.

- *Re-use and Sharing of Domains.* The distributed architecture of the DCG allows for authors to collaborate and cooperate in editing domain structures and relating TMs to the concepts / topics. It also allows a reuse of TMs and domain structures. Libraries of often used concepts / topics and corresponding URLs can be developed. An electronic domain represented in the DCG, linked to actual documents on the WWW, such as "hot" scientific papers, or just textbook explanations can be shared by lecturers teaching the same subject at different universities, where everyone can make extensions and modifications according to his/her personal view.

- *Learners as Authors.* Modern learning theories point out positive effects of letting the learner create his/her own understanding and knowledge structures feeling: motivation because of feeling "ownership" over the problem, development of searching — and metacognitive — skills. For this reason students are often left to plan a lesson themselves, for example, by organizing lectures as seminars. This enables them to create an individual view of the domain, and to search for new information. As an authoring tool the DCG can be used for carrying out this type of projects. For example, a student or a team can be assigned the task of authoring a certain theme (subdomain). The student / team has to review the literature, discover important concepts / topics and relationships and create a domain structure, to produce or find related materials on the WWW and link them to the concepts / topics in the domain structure. The structure and materials will be discussed and criticized by the lecturer and the class, and the domain produced in this way could be used later by the DCG for automatic generation of instructional courses on this theme.

We haven't been able to carry out empirical evaluation of the learning effect of the system with enough students and to compare their results with a control group. However, our first experiences show a very positive attitude and we expect good results, especially with introductory courses in basic disciplines read at the University, like in mathematics, where students come with very different background knowledge and adaptive composition of the course for every individual student would have obvious advantages.

6. 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 / topics 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.

• **individualization of instruction**. Teaching of complex technical systems requires considering the fact that students have different levels of knowledge, motivation, confidence. By continuous monitoring of students' behavior and requesting their feedback, the system maintains a model of student knowledge, personal traits and preferences and therefore is able to find alternative course-plans, instructional methods, and TMs which are dynamically tailored to the students' benefit.

• **possibility to assign and change the teaching rules** of the system. A human teacher must have a clear metaphor of the system's functioning mechanism. He/she can think of the system as a teaching agent. This agent can be "instructed" by means of modifying or creating new teaching rules.

• **possibility for easy authoring**, re-use of already developed courseware and using technical documentation as a basis for developing Teaching Materials. This is obtained by separating the Domain Structure from 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. The actual authoring process is shifted to a higher level: to represent explicitly the structure of the concepts / topics and instructional tasks. This is a non-trivial task. However, we believe this is a justified effort in order to create a system able to teach in a variety of ways for a variety of goals.

References

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

I. BEAUMONT AND P. BRUSILOVSKY, (1995) Adaptive Educational Multimedia: from Ideas to Real Systems, In Proceedings of ED-MEDIA'95, Graz, AACE: Charlottesville, VA.

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

CALDERHEAD J., (1991) Representations of Teachers' Knowledge, in Teaching Knowledge and Intelligent Tutoring, P. Goodyear (Ed.), Ablex: Norwood, N.J., pp. 269–278.

T. DIESSEL, A. LEHMANN AND J. VASSILEVA (1994) Individualized Course Generation: A Marriage Between CAL and ICAL. Computers and Education, Vol. 22, No. 1/2, pp. 57–64.

W. EINSIEDLER, (1976) Lehrstrategies und Lernerfolg: eine Untersuchung zur lehrziel und schülerorientierten Unterrichtsforshcung. Weilheim: Beltz.

M. ELSOM–COOK AND C. O'MALLEY, (1989) Bridging the gap between CAL and ITS. CITE Report 67, I.E.T., The Open University, Great Britain, 1989.

A. FLAMMER, (1975) Wechselwirkungen zwischen Schülermerkmalen und Unterrichtsmethoden (WSU). In. Schwarzer R., Steingaden K. (Hrsg.): Adaptiver Unterricht: Zur Wechselwirkung von Schülermerkmalen und Unterrichtsmethoden. Munchen: Kösel, pp. 27–41.

R. HEWETT AND B. HAYES–ROTH, (1994) Physic Systems Modeling for Integrated Reasoning, Modularity and Reuse, International Journal of Expert Systems, vol. 7, no. 4, pp. 359–386.

HORSTMANN, E., (1995) Entwicklung und Implementierung eines Moduls zur Generierung von pädagogischen Entscheidungsregeln, Diplomarbeit ID 10/95, Fakultät für Informatik, Universität der Bundeswehr München.

J. LARKIN AND R. CHABAY, (Eds.) (1992) Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches, Introduction chapter, pp. 1–10, Lawrence Erlbaum Assoc: Hillsdale, NJ.

G. LEINHARDT, (1988) Situated Knowledge and Expertise in Teaching. In J. Calderhead (Ed.) Teachers' Professional Training, London: Falmer, pp. 146–168.

J.–L. LEONHARDT, M. BAKER AND C. BESSIERE (1991) Towards Integration of Multimedia and Artificial Intelligence in Courseware Production SHIVA: A Step in European Context. In Proceedings of CALICSE'91, Lausanne, pp. 35–42.

M. LINARD AND R. ZEILIGER, (1995) Designing Navigational Support for Educational Software, in Blumentahal B., Gornostaev, J. and Unger C. (Eds.) Proceedings EWHCI'95, Moscow, Springer Lecture Notes in Computer Science No 1015, pp. 63–78.

A. MITROVIC, S. DJORDJEVIC AND L. STOIMENOV, (1996) INSTRUCT: Modeling Students by Asking Questions, User Modeling and User Adapted Interaction, 6 (4), pp. 273–302.

K. MCKEOWN, (1985) Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text, Cambridge University Press: Cambridge.

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

N. NILSSON, (1980) Principles of Artificial Intelligence, Tioga Press: Palo Alto, CA.

C. PARIS, (1993) User Modelling in Text Generation, Communication in AI Series, Pinter: London.

D. PEACHEY AND G. MCCALLA, (1986) Using Planning Techniques in Intelligent Tutoring Systems, International Journal of Man-Machine Studies, 24, pp. 77–98.

R. SIEGLER, (1988) How Content Knowledge, Strategies and Individual Differences Interact to Produce Strategy Choices. In Interaction among Aptitudes, Strategies and Knowledge in Cognitive Performance. (Schneider & Weinert, Eds.) Springer: New York, pp. 74–88.

K. VAN MARCKE, (1991) A Generic Task Model for Instruction, in Proceedings of NATO Advanced Research Workshop on Instructional Design Models for Computer Based Learning Environments, Twente, July, 1–4, 1991.

J. VASSILEVA, (1990) An Architecture and Methodology for Creating a Domain-Independent Plan-Based Intelligent Tutoring System. Education and Training Technology International, 27, 4, pp. 386–397.

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

J. VASSILEVA, (1995) Reactive Instructional Planning To Support Interacting Teaching Strategies, Proceedings of AI-ED 95, World Conference on AI and Education, pp. 334–342, AACE: Charlottesville, VA.

J. VASSILEVA AND R. DETERS, (1997) Dynamic Courseware Generation on the WWW. Proceedings of PEG'97, Sozopol, Bulgaria, 30.05.–1.06.1997, pp. 111–117.

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

B. WOOLF, (1987) Theoretical Frontiers in Building a Machine Tutor. In Artificial Intelligence and Instruction: Applications and Methods (G. Kearsley, Ed.), pp. 229–267, Addison–Wesley: Reading.

Received: February, 1997
Accepted: July, 1997

Contact address:
Julita Vassileva
Universität der Bundeswehr München
85577 Neubiberg
Germany
E-mail: jiv@informatik.unibw-muenchen.de

JULITA VASSILEVA received her Ph.D. in Computer Science from the University of Sofia, Bulgaria, in the field of Intelligent Tutoring Systems. Since 1992 she has been working as a research assistant at theFederal Armed Forces in Munich, Germany. Dr. Vassileva has worked mainly on the development of user-adaptive systems: intelligent tutoring systems and user modeling.