A New Approach to Authoring of Adaptive Courseware for Engineering Domains

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

Summary

Our approach provides an alternative to traditional CAL-authoring. It is based on a different concept of a course, that is generated dynamically during instruction in a way to suit better the student's individual progress and preferences. The teacher is actively involved in the creation of a course without the need of knowledge in authoring. The actual authoring process is shifted to a higher level: the author has to represent explicitly the structure of the concepts and links that have to be taught. The system uses this knowledge to generate different course-plans for different goals and execute them by selecting presentation materials, depending on the teacher's instructions and the student's preferences. An application of this approach in an engineering domain is shown.

Keywords: authoring, adaptive courseware, "slightly-intelligent" tutoring systems, ITS authoring.

1. Introduction

In the last two decades the field of CAL has been developing mainly in the direction of including new representation media in the old concept of courseware. In the same period the field of ITS underwent an intensive growth, thriving and even some disappointment about its prospective for practical application (Bierman, 1991). However, it did not influence the traditional CAL-concept of a course. Only recently the need of bringing together the two fields has been recognized (Larkin & Chabay, 1992) and there have been attempts for "intellectualizing" CAL and CAL production (Wentland et. al, 1991), (Brussilovsky, 1992), (Murray, 1992). We see a first and most significant step in this direction the dynamic generation of adaptive courseware (Vassileva, 1992), (Diessel et.al., 1994). This is particularly important for teaching complicated domains, like science and engineering since the student's understanding at every step in following the course is vital and sometimes the course needs to be far more flexible than is possible within the traditional CAL concept of courseware. In this paper we describe an application of our architecture for dynamic courseware generation to an engineering domain characterized by a complicated and frequently changing device information. A prototype of this application has been developed for teaching the structure, functioning, maintenance, diagnostic, etc. of a simple electrical device (toaster in the examples).

2. Requirements

A list of requirements is given below to which our system conforms.

2.1. Requirements for adaptive courseware

In a dynamic courseware generator, the system doesn't lead individualized interaction at the smallest grain-level. The interaction has a pre-determined flow as in traditional CAL courseware. Adaptation to the individual student takes place in the following aspects:

• In the selection of the course to be followed. The course is generated to fit the teaching goals of the particular student.

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

• In the selection of the starting point in the course for the particular student depending on his prior knowledge.

• In the selection of presentation materials that are appropriate for the student.

• In the way of following the course. In CAL systems a course is defined *a priori* assuming a knowledge state of a hypothetical "average" student. The actual knowledge state of the particular student is not taken into account, neither is its development during the course. In a dynamic courseware generator a course is generated in the beginning according to a particular teaching goal, but the system is able to infer the initial status and the development of the student's knowledge state and to modify the course according to this development.

• In the way of "branching". While in CAL courseware branching of a course is usually an immediate response to a student's error, in our understanding the system should be able to judge whether and when a branching or changing the course is needed by evaluating the student's knowledge state, not necessarily in response to a trigger. It is a general opinion that the system must by reactive, but to our mind this is a pedagogical decision of the competence of the Teacher who is going to use the system in his classes, or of the student himself and should be taken dynamically during the course. This decision shouldn't be taken once forever at the stage of creating the course.

In this sense, we consider another aspect of adaptivity: the possibility to adapt courseware to the Teacher 's wishes, goals and preferences. The traditional CAL concept of courseware excludes the Teacher, unless he decides to take the role of the Author and write his own courseware. In this we see one of the main obstacles for a wider application of CAL in traditional educational settings. We believe that the Teacher should be provided with means to generate a course according to his wishes without the need of authoring.

2.2. Requirements for teaching complicated technical systems

Besides the above mentioned differences between the traditional CAL concept of courseware and our concept of adaptive courseware, there are some other differences, implied by the specifics of teaching advanced and complicated technical systems.

• It is important to be able to teach the same contents with different teaching goals by taking different viewpoints. The current technical systems are so complicated that it is impossible to describe them comprehensively without considering different aspects and levels of abstraction. For example, a device can be described from a structural, functional or geometrical viewpoint, or from different combinations of these main ones, when teaching, say, how to install the device, how to maintain it, or how to diagnose it. Within the traditional CAL courseware-concept this can be done only by creating a different course for every possible viewpoint or combination of viewpoints.

• It is important to maintain a centralized representation of the device-dependent knowledge, so that it can be easily updated without affecting the rest of the system knowledge. This is vital for advanced technical systems that change every, say, three months. Within the traditional CAL courseware concept, the course has to profoundly analysed and to make the changes considering their potential effects and the corresponding changes throughout the course. This is very difficult, because there are no tools helping to keep consistency of the course when some change is done.

Therefore, we need a centralized representation of the device-dependent knowledge, which can be updated easily. From the other side, we need a representation of the teaching materials that will display this knowledge to the student. The base of teaching materials contains "poly-valent" atoms that can be combined to form "molecules", that in their turn can be combined to form a course of instruction for obtaining a given teaching goal. In addition the course generator contains:

• a course-planner that creates a course-plan adjusted to the student's prior knowledge and targeted to obtain the teaching goal given by the Teacher ,

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

• a course-executor that evaluates the student knowledge (maintains a student model), carries out the course-plan according to the student's presentation preferences and if needed, invokes the planner to modify the course-plan, and

• a student model to register the student's history of teaching and the current state of his knowledge.

3. Architecture

The architecture of the system consists of the following components (figure 1):

3.1. The Data Base

The subject dependent knowledge is contained in the Data Base Component. It contains two parts:

• The **Teaching Materials** (TMs) contains presentation- and testing-"atoms" that carry out the communication with the student. They can be any means of visualisation of the knowledge that has to be taught: text, graphics, animation, video, sound, etc. Their most important feature is that they are "atomic" in the sense that they are focused on a given element of knowledge (concept, part of device, function) or explaining a particular relationship (link) between two elements of knowledge. They are self-explainable and a combination of them creates a course. The TMs do not contain those facts about the device that are expected to change. This information is stored in the Concept Structure level, where it can be easily updated. The TMs contain only parameters that are substituted with the real values during the execution of the course.

• The **Concept Structure** contains the structure of the knowledge that is going to be taught. It is represented with two data-types: nodes, corresponding to the elements of knowledge (concepts) and links, corresponding to the possible relationships between them. These main elements of knowledge representation are very appropriate for describing technical systems (Tyrväinen, Saarinen, Hätönen, 1993). The Concept Structure is the skeleton of the knowledge that is going to be taught. During the course execution, "flesh", i.e. teaching materials is added to represent this knowledge. More about the the Concept Structure is given in section 4.

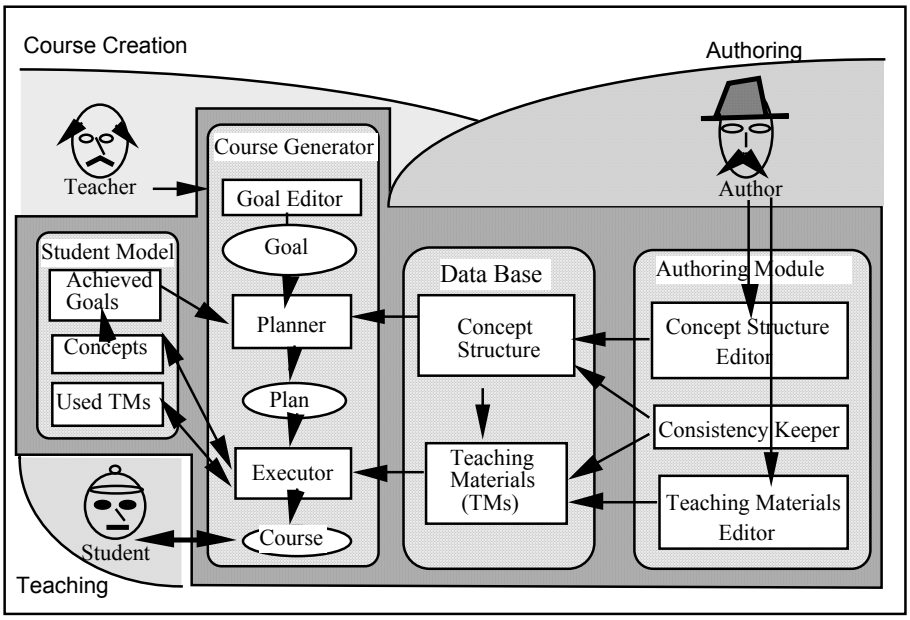


Figure 1

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

3.2. 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:

• the Course **Planner**. It generates a plan of the course according to the teaching goal and viewpoint assigned by the Teacher. The plan consists of a set or a sequence (depending on the teaching strategy) of nodes and links from the concept structure that have to be taught during the course. This plan is passed to the following component,

• the Plan **Executor.** It generates a course based on the plan by selecting and presenting appropriate TMs. In the same time the Plan Executor updates the Student Model taking into account the results of the student on the test-atoms. The Plan Executor invokes remedial TMs (explanation of the reason of the error, associated with the test-atom) for providing immediate feedback to the student's errors. It also keeps track of the changes in the student model and is able to re-invoke the Course Planner to create a new plan, if the level of knowledge is not satisfactory. More about the mechanism of re-planning is given in section 5.

3.3. The Student Model

The Student Model contains three levels:

• At the first level, the identifiers of the TMs that were already used are stored with annotation of their type and success with the particular student.

• At the second level, the student model contains an overlay with the concept structure in the data base, containing probabilistic evaluations of the beliefs that the student knows and does not know a given concept or link. This is not an original student modelling technique and we will not discuss here in detail its mechanism. The interested reader can refer to (Villano, 1992) and (Diessel et. al., 1993).

• At the third level, the student model contains a list of the accomplished sub-goals (learned concepts and links are considered those, whose belief probabilities exceed a threshold value assigned by the Teacher). If during course-following the belief probabilities of certain concepts drop under the threshold (this may well happen since one test-atom can affect the belief probabilities of several concepts at the same time), this concept will be deleted from the accomplished goals list in the student model and will be re-taught again. The first level Student Model will be used in order to avoid using the same TMs.

3.4. The Authoring Component

The Authoring Component consists of a TMs-Editor and a Concept Structure Editor.

The TMs-Editor can be any authoring tool for producing multimedia materials. Its purpose is to extend the presentation formats for the domain information. All factual data that is expected to change often should be parametrized and the parameters - referring to the corresponding objects in the Concept Structure. Authoring with this editor is in fact an initialisation of the system and it normally takes place once in adjusting the system for a given domain.

The second editor allows changing factual data about the concepts as well as extending and modifying the structure of concepts and links. A consistency keeper takes care of the coverage of the concept structure with TMs. If concepts/links are introduced which have no corresponding TMs in the database, the Author is warned to create them by using the second editor.

4. The Concept Structure

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

The Concept Structure as described in the previous section suggests a homogeneous network of concepts and relationships of different types among them. The relationships can have different semantic (e.g. aggregation, abstraction, analogy, or other). In order to generate a meaningful course for teaching a given concept, the Course Planner needs knowledge about the semantic of links to related concepts that has to be used. This has to be assigned by the Teacher, since there could be many different ways to teach one concept and deciding in which way this should be done is a part of the teaching goal. This, however, would pose difficulties to the Teacher, since he is then supposed to know not only what concepts are there, but also all different types of links between them. A way of systematizing the possible links between the concepts is needed.

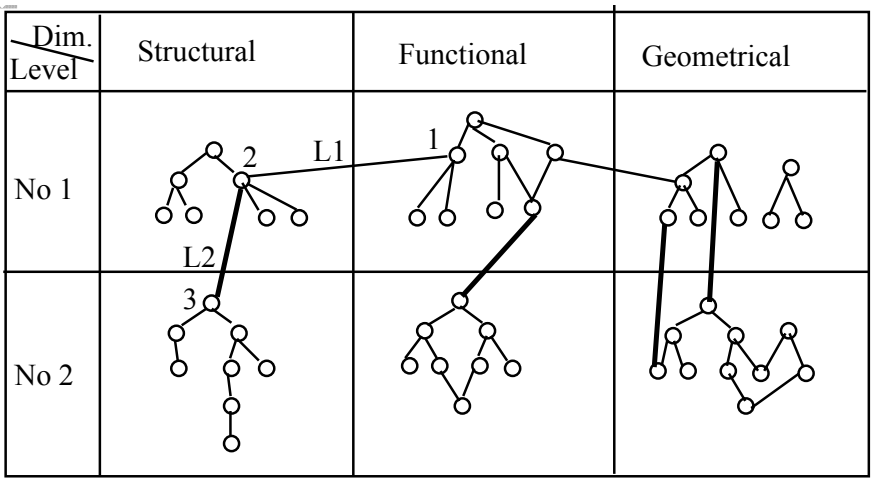
As already mentioned before, it is impossible to describe a complex engineering system without considering a structured representation allowing different aspects and levels in the description. In the highly complex domain of chip-design, for example (Gajski, Kuhn, 1983), (Walker, Thomas, 1985), a standard technique is distinguishing between a structural, functional and geometrical aspect (called "dimensions") of the description of a chip and each of these has many different levels of abstraction (generality or detail). Each dimension allows a self-contained description of the chip according to this dimension.

For a given teaching goal often it is enough to consider only one of these dimensions at one single level. However, sometimes a device has to be described referring to several different dimensions. That is why the representation should allow switching from one to another dimension, possibly at a different level of detail. For example, when explaining the structure of a toaster's lifting mechanism, one could wish to explain also the function of a thermostatic device.

To allow distinguishing between different aspects in the description of a device or system, we introduce the notion of "dimension" at the concept-level. For our purposes the three dimensions mentioned above (functional, geometrical and structural) are enough, however they could be more, if this is considered important by the Author (for example, costs, behavior etc.). A device is described by means of a structure of concepts (nodes) and links. The links can be one-dimensional and one-level (reflecting the structure in the description). For example, links between nodes corresponding to the component and its sub-components in the structural dimension, or between a function and its sub-functions in the functional dimension. They could be between two different levels, reflecting the abstraction- or aggregation- links between the descriptions (concepts). The one-dimensional links are normally hierarchical. There are also cross-dimensional links, which have more specific semantic. For example, a link with the semantic "is implemented by means of" can connect a node corresponding to a given function with a node corresponding components in the structural dimension that implements the function.

The structure of the concept level is shown in Figure 2.

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.



Explanation:

- concept (node)
- 1-dimensional, 1-level link
- 1-dimensional, 2-level link
- 2-dimensional, 1-level link
- 2-dimensional, 2-level link

Figure 2.

The main object types are nodes and links. The representation of nodes and links is given below:

node name: node-name;
  parameters: list of parameters;
  dimension: name-of-dimension;
  level: level-ident;
  presentation: list of (present-atom |type |time);
  test: list of (test-atom | difficulty | time).

link name: link-name;
  parameters: list of parameters;
  dimension: {name-of-dimension,
    name-of-dimension: name-of-dimension};
  level: {level-ident, leve- ident : level-ident};
  source: node-name;
  destination: node-name;
  presentation: list of (present-atom | type | time);
  test: list of (test-atom | difficulty | time).

Examples of a node and a link in a toaster-description are given below.

node name: *toast ejecting*;
  parameters: list of parameters;
  dimension: *functional*;
  level: *No 1*;
  presentation: *(te1 | text | 4'), (te2 | video | 2'),*
    (te3 | animated schema | 3');
  test: *(qte1 | 2 | 4')*.

link name: *parts of toast ejector*;
  parameters: list of parameters;
  dimension: *structural*;
  level: *No 1 : No 2*;
  source: *toast ejector*;
  destination: *toast ejector set*;
  presentation: *(pte1 | schema | 3'),*
  test: *(qpte1 | 2 | 4'),*

The "parameters"-part of the structure contains device-information that is subject of changes during time. The "dimension", "level", "source" and "destination" (in case of a link) parts are used by the Planner to decide whether to present the node or the link to the student (depending on whether they can be related to the teaching goal). The "presentation" and "test" parts are used

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

by the Executor to select presentation- (test-) atom of a type (difficulty) and a duration appropriate for the student.

5. Dynamic Course-Generating and Authoring

5.1 Plan generation

A course plan based on the Concept Structure is generated automatically by the Planner. The main input for plan-creation is the teaching goal of the course, assigned by the Teacher .

The Teacher is provided with a Goal Editor that allows him to assign a teaching goal (either a concept or a link in the Concept Structure) and to give his requirements for the course-plan. These can be:

• Dimension(s) and level(s) that have to be explored by the course. For example "Teach the component *Ejector* up to level *Smallest-grain-level* of sub-components".

• Nodes or links that must be included. For example, if the Teacher wants that the course contains not only explanations of the function Eject at its sub-functions are presented, but also wants to include presentation of how these functions are implemented with components, he will include *cross-dimensional-link-Functional-Structural.*

• Type of presentation-atoms and difficulties of test-atoms that he prefers to be used. If he doesn't assign any, the system will select ones dynamically according to the first-level of the student model.

• Time allotment for the course. If the Teacher assigns a type of presentation atoms, the sum of their time-durations is calculated and the system tries to include as many concepts and links as possible in the time allotment.

The plan generator creates a plan by including the concepts (links) that are not known by the student, but always trying to start with links to known concepts (links) or at least with ones with high enough knowledge probabilities. The mechanism for the plan generation is described elsewhere (Diessel et. al, 1994).

5.2. Course execution

There are two main teaching strategies employed by the system: a discovery-oriented one, with the initiative belonging to the student and a traditional tutoring one, with the system having the initiative of presenting teaching and testing materials to the student.

In the first strategy the system provides a map of the course, with all the concepts and links between them and the student can select the one he wants to learn at any time. After this selection, the system presents the corresponding teaching materials (presentation- and test-atoms), with the actual device data, evaluates the student's answers and updates the student model. If the student makes an error when answering a test-atom, the provided for it remedial teaching material is shown.

With the second strategy, the system uses the student model to guide instruction by consecutively presenting to the student the links and concepts from the plan starting with presenting the links connecting new concepts with concepts that are already familiar to the student (with higher belief probabilities).

Within both strategies, it may happen that the student is not able to acquire some concept within the time provided for it. A sign for this may be not only a single incorrect answer to a test-atom, but an insufficient knowledge probability in the student model after several presentation- and test-atoms. In this case the plan executor invokes the course planner to change the plan of the course, bypassing the difficult concept.

Now a flexible teaching strategy, with mixed initiative defined by rules for pedagogical decision making is being experimented (Vassileva, 1993). These rules are acquired through supervised machine learning from teachers.

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

5.3. Authoring

The Authoring tool allows an easy creation and updating of the data base. We distinguish between an Author and a Teacher, who will use the ready system in his domain and adjust it to his requirements (goal, planning requirements, presentation preferences, etc.). The Author is expected to create first the concept structure for a given domain with the help of the Concept Structure Editor which visualizes the nodes and links between them and prompts about the information the Author has to input. He has also to create the base of TMs, by using a commercial authoring tool and adding to each of them parameters showing the TM's type and time allotment. Every data that is expected to be modified has to be parametrized. The real values are kept in the Concept Structure; they will substitute the parameters in the corresponding TMs at the stage of execution of the course. When updating these values, the Author is prompted that there might be a need to change the values of the parameters in the related concepts/links. In this way, the system helps the Author to propagate changes of system-dependent parameters in the concept structure.

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 has demonstrated knowledge about it. To provide means for the system to evaluate the degree of knowledge of each of the nodes/links 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 nodes/links, if he answers correctly and if he 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 nor hard for the Author to give a quantitative estimation of the type "If the student answers incorrectly to this question, in 75% of the cases it means he doesn't know concept X and in 10% that he doesn't know link Y". If the Author wants that the system is reactive, i.e. to give an immediate feedback to the student's errors, he has to create for every test-atom an associated remedial material. This could be, for example, just a text-window annotating the error and possibly giving the right answer. No diagnosis of the reason for the error is necessary, since we are not aiming to have a really intelligent system. Instead of teaching by addressing the reasons of the student's misconceptions, the system is trying to teach by alternative representations.

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 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, video- and sound- records.

At this stage a prototype of the system is implemented and it has been tested in the domain of electric toasters. Even in such a simple domain, it turns out that the concept structure is quite complicated. In the geometry-dimension there are 22 nodes corresponding to physical parts of the toaster organized at 2 levels of detail. The links have semantic: "is connected to". There are 12 links at the first level and 29 at the second level. In the functional dimension 12 functions (nodes) are organized at one level connected with time-relations of 3 types: "must be done before", "must be done after" and "must be done in parallel". At the structure-dimension there are 18 nodes in 3 levels connected with hierarchical links of the type "is a part of". There are 5 cross-dimensional links between the structure- and functional-dimensions with the semantic "is implemented with" and 5 between the structure- and geometry- dimensions with the same semantic.

The main part of time (one week) an Author spent for "paper and pencil" development of the concept structure. The coding of the structure with the Concept Structure Editor took half a day. 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 10 seconds. The Teacher found reasonable 13 different teaching goals divided in 3 groups: montage, maintenance and repair. The length of the generated plans (in terms of nodes and links

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

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 TMs.

In order to evaluate the effort spent for creating an hour of instruction, the time spent for developing + encoding the Concept Structure and the time for development of all TMs that cover the concept structure 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). It is obvious that, if the Concept Structure allows generation of many courses with many different goals, the extra-efforts for creating it will be justified. We believe that authoring with our system is far more effective than authoring in the traditional sense. However, it is very hard to give a precise evaluation since the more possible courses, the more complicated the concept structure, the more effort spend for authoring the concept structure and TMs. We will be able to claim that the system is more effective only after experimenting in a more complicated technical domain. Now we are doing this for the domain of telecommunications.

7. Conclusions

Our approach provides an alternative to the traditional approach for authoring in CAL. It is based on a different concept of a courseware, the so-called "adaptive" course that is generated dynamically during instruction and in a way to suit better the student's individual teaching goal and way of acquiring the material. We believe this approach is particularly good for teaching complicated, rapidly changing engineering devices and systems, since it provides the possibility to modify the factual data that is taught independently, without taking care of changing correspondingly the teaching materials. Another important feature is the possibility of taking different viewpoints when creating the course according to different teaching goals.

Though having certain "intelligent" features, our system is not an ITS since there is no diagnosis of the student's answers, dynamic domain and teaching expertise (including some elements of teaching expertise is a topic of our current research). We believe this is a reasonable compromise between ITS and CAL that results mainly in an advanced authoring concept, because the course is created automatically. 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 links that have to be taught. 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.

Acknowledgements
This work has been partially supported under project I-24 by the Bulgarian Ministry of Science and Higher Education. I am very grateful to Thomas Diessel and Axel Lehmann for useful discussions of some of the ideas presented in this paper.

References

Bierman, D. To Be Intelligent or Not To Be: Is That the Question? Proc. Internat. Conf. on Computer Assisted Learning and Instruction in Science and Engineering, CALISCE´91, pp.25-34, (1991).

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, pp.499-506.

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

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.

Gajski D., Kuhn R. (1983) Guest Editor's Introduction: New VLSI Tools, Computer, 12.

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

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

Tyrväinen P., Saarinen P., Hätönen K. (1993) Domain Modelling for Technical Documentation Retrieval. Information Modelling and Knowledge Bases IV, H. Kangassalo et al (eds.) IOS Press: Amsterdam.

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

Vassileva J. (1993) A Framework for Taking Local Pedagogical Decisions in Tutoring and Coaching, Proceedings of AI-ED 93, World Conference on AI and Education, pp. 594.

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, pp. 491-498.

Walker R., Thomas D. (1985) A Model of Design Representation and Synthesis, Proc. of 22nd Design Automation Conference.

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

Published in the Proceedings of the International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Paris, August 31 - September 1-2 1994, pp. 241-248, Telecom - Paris 1994.