Course sequencing techniques for large-scale web-based education

Peter Brusilovsky

Department of Information Science and Telecommunications,
School of Information Sciences, University of Pittsburgh,
135 North Bellefield Ave., Pittsburgh, PA 15260, USA
E-mail: peterb@mail.sis.pitt.edu

Julita Vassileva

Computer Science Department, University of Saskatchewan,
1C101 Engineering Bldg.,57 Campus Drive, Saskatoon, SK,
S7N 5A9, Canada
E-mail: jiv@cs.usask.ca

Abstract: We argue that traditional sequencing technology developed in the field of intelligent tutoring systems could find an immediate place in large-scale web-based education. This paper discusses two models that have been explored by the authors – the dynamic course generation system DCG and the concept-based course maintenance system CoCoA. DCG includes components for domain authoring and for automatic generation of adaptive courses on the WWW. It allows automatic generation of individualised courses according to the learner's goal and previous knowledge, and can dynamically adapt the course according to the learner's success in acquiring knowledge. . CoCoA can check the consistency and quality of a course at any moment of its life and also assists course developers in some routine operations.

Keywords: Course sequencing; personalisation; adaptive courseware; learning object.

Reference to this paper should be made as follows: Brusilovsky, P. and Vassileva, J. (2003) 'Course sequencing techniques for large-scale web-based education', *Int. J. Continuing Engineering Education and Lifelong Learning*, Vol. 13, Nos. 1/2, pp.75-94.

Biographical notes: Peter Brusilovsky is an assistant professor of Information Science and Intelligent Systems at the University of Pittsburgh. He received his PhD in Computer Science from the Moscow 'Lomonosov' State University in 1987. His research interests lie in the areas of adaptive hypermedia, adaptive web-based systems, student and user modelling, intelligent tutoring systems, and web-based education. He has authored over 80 technical papers and has edited several books and special journal issues.

Julita Vassileva is an associate professor of Computer Science at the University of Saskatchewan. She received her PhD in Mathematics and Computer Science from the University of Sofia, Bulgaria in 1992. Dr. Vassileva's research interests are in the areas of artificial intelligence in education and user modelling. More recently, her research has focused on creating and motivating virtual learning communities using multi-agent systems, agent negotiation, coalition formation and peer-to-peer systems. She has authored over 60 technical papers and co-edited two volumes.

1 Introduction

Course sequencing is a well-established technology in the field of intelligent tutoring systems (ITSs). The idea of course sequencing is to generate an individualised course for each student by dynamically selecting the most optimal teaching operation (presentation, example, question, or problem) at any moment. By optimal teaching operation we mean an operation that in the context of other available operations brings the student closest to the ultimate learning goal. Most often, the goal is to learn the required set of knowledge up to a specified level in a minimal amount of time. However, it is easy to imagine other learning goals, such as minimising student error rates in problem solving.

An ITS with course sequencing represents knowledge about the subject as a network of concepts where each concept represents a small piece of subject knowledge. The learning material is stored in a database of teaching operations. Each teaching operation is indexed by the concepts it deals with. The driving force behind any sequencing mechanism is a student model that is a weighted overlay of the domain model – for every domain model concept it reflects the current level of student knowledge about it. Using this model and some teaching strategy a sequencing engine can decide which one of the many teaching operations stored in the database is the best for the student given his or her level of knowledge and educational goal.

Various approaches to sequencing were explored in numerous ITS projects. The majority of existing ITSs can sequence only one kind of teaching operation. For example, a number of sequencing systems including the oldest sequencing systems [1,2] and some others [3–5] can only manipulate the order of problems or questions, an approach usually called *task sequencing*. A number of systems can do sequencing of *lessons* – reasonably big chunks of educational material complete with presentation and assessment [6,7]. The most advanced systems are able to sequence several kinds of teaching operations such as presentation, examples, and assessments [8–10].

One could say that sequencing is an excellent technology for distance education. Indeed, sequencing is now the most popular technology in research-level web-based ITS [11]. However, there is a significant difference between the systems used in large-scale web-based education and research-level systems, even if we only consider research systems that were used to teach real classes such as ELM-ART [12] and 2L670 [13]. In a modern, large-scale web-based education context a single course provider operates tens to hundreds of courses that have to be delivered to thousands of students grouped into classes. The biggest concern of a provider is the problem of maintenance. To avoid problems with installation, support and training, all serious providers of multiple web-based education (WBE) systems tend to choose one single course management system (CMS).

Usually the providers choose modern commercial CMSs such as TopClass [14], BlackBoard [15] or webCT [16] that can support the main needs of a course provider from course material delivery to discussion forums to generation of various course reports. Unfortunately, current CMS systems leave no space for dynamic sequencing. The course model behind the majority of these systems is a static sequence (or a tree) of modules followed by static quizzes and assignments. Even the most advanced CMS systems such as TopClass never go beyond the classic computer-assisted instruction approach (CAI). This approach offers some limited ways to change the student's path through the material on the basis of his or her performance on a quiz. This static structure contradicts the traditional sequencing approach that accepts no predefined structure and

instead builds the sequence on the fly, presenting the student with one teaching activity at a time.

Could we find any use for the course sequencing ideas in this rigid context of large-scale web-based education? Our answer is 'yes'. The goal of this paper is to present three approaches that make it possible to use the benefits of adaptive sequencing in the context of practical web-based courses delivered through a standard CMS system.

2 Three approaches for the use of course sequencing in the context of large-scale web-based education

The first approach that we describe is to use a course sequencing mechanism as the core of a *course maintenance system* for traditional statically sequenced courses developed by a team of authors. The idea is simple. Since a sequencing mechanism can evaluate several possible options for the 'next steps' (i.e., presentation, example, assignment) and select the best one, it can also check whether the 'next step' predefined by the author in a traditionally developed course is a good one. If the next step is not really appropriate, the mechanism can report problems. For example, it can find a situation when an assessment requires knowledge that has not been presented yet or, vice versa, when the presented knowledge is never assessed. This kind of course consistency checking is necessary for any serious course development team. Large-scale modern courses include hundreds to thousands of learning items that are produced by a team of developers. Throughout the lifespan of a course it could be updated and restructured several times. A concept-based course maintenance system is as important for *courseware engineering* as a version tracking system is for software engineering. The strength of this first approach is that it can be used with any existing course and provide visible benefits. Its weakness is that the main benefit of dynamic sequencing – the ability to adapt the course to an individual student – is not applied here. This approach is implemented in the CoCoA system described in Section 4 of this paper.

A more progressive way of developing web-based courses, one that is growing in popularity, supports courseware reusability [17]. It assumes that courses are developed from reusable content objects that are stored in special pools and databases. The coming generation of CMSs provides stronger support for courseware reuse, thereby enabling authors to produce new courses from existing material faster. One of the major goals of courseware reuse is to support course customisation, i.e., by producing several versions of the same course, from the same rich set of learning objects, but targeted to different audiences. This context provides a fertile ground for another sequencing approach – adaptive courseware generation.

The idea of adaptive courseware generation is to generate a course suited to the needs of the students before they encounter it. Instead of generating a course incrementally, as in a traditional sequencing context, the whole course could be adaptively generated in one shot. This approach has several strong advantages. First, it can deliver an impressive level of adaptivity for small homogeneous groups of students by taking into account their learning goals and starting level of knowledge. Second, since all students will be following the same course, the shared context will allow them to communicate and learn from each other. Third, since a course generated in this way is static, it can be delivered by a regular CMS. The weakness of this approach is that the courses produced by one-

shot generation are not as adaptive as incrementally sequenced courses. This approach is supported by the DCG system [10] by 'switching off' the dynamic re-planning option. This system is presented in section 3 of this paper. Similar one-shot customised course generation approaches were suggested in [18–22].

The last of the approaches that we are suggesting is dynamic courseware generation. As in the previous approach, the goal of dynamic courseware generation is to generate an individualised course taking into account a specific learning goal and the initial level of the student's knowledge. The difference is that the system with dynamic generation observes and adapts to student progress with the generated course. If the student's performance does not meet expectations, the course is dynamically re-planned. The benefit of this approach is that it applies as much adaptivity to an individual student as possible in the context of CMSs. Through dynamic regeneration each student is able to get a highly personalised course for his/her needs. Between regenerations, the course stays static and can be delivered with any CMS. This approach is well suited for individual students taking a self-study distance-learning course. These students can be employees in an organisation who have different experience and background knowledge, or students in an online university with different ages, backgrounds and goals. An appropriate solution would be to generate a fully individualised course. Such a course would be specifically generated to take into account the students' existing knowledge, goals, and timeframe, adapting dynamically to their difficulties and rate of progress. DCG, proposed in 1992 [10], allows this type of dynamic courseware generation. Note that for a class- or group-based education, the use of the individualised generation approach requires some coordination of the planning mechanism on the group level. This is necessary to ensure reasonable group cohesion (for example, on the level of lectures or other large course fragments), while still allowing for individual variation in the paths followed by the students.

The goal of this paper is to promote the approaches listed above. In the next two sections we present the two systems -- CoCoA and DCG -- that can successfully apply course sequencing in the context of practical web-based education. In the following chapter we analyse common features of these systems to stress key steps towards using sequencing in practical web-based education. In our conclusion, we analyse differences between the systems and speculate about the future of course sequencing techniques in the context of large-scale web-based courses. We consider the approaches introduced to be an important evolution from the currently dominant static web-based courses to the more flexible and adaptive web courseware of the future.

3 DCG – dynamic generation of customised courses

The Dynamic Course Generation system (DCG) is a compromise between an ITS and a traditional CAI approach. It uses a concept structure as domain knowledge representation. A course generated by DCG looks like a traditional structured course. However, this course is generated individually for every student to achieve a certain learning goal (a concept or topic that has to be learned). The generation takes into account the already existing knowledge of the student and can accommodate differences in the individual's way and pace of acquiring the material.

The core of the DCG architecture is the explicit representation of the domain concept structure, separated from the teaching materials and pedagogical tasks [23,10]. The DCG

uses a structure of concepts (represented as a set of rules) as a roadmap to generate a plan of the course. Given a certain goal concept that the learner wants to acquire and a student model containing the concepts already known by the learner (initialised with a pre-test), a planner component searches for a route that connects the concepts known by the learner with the goal concept. The learner sees a sequence of teaching materials related to each concept from the plan. At every point the learner can be tested on his/her knowledge of the current concept by presenting a set of test items. A student model is created for every learner. This model is a numeric overlay on the concept structure, i.e. the student's knowledge of each concept is represented as a number within a certain interval. If the learner is not able to achieve the threshold score for a given concept that is needed to proceed further towards the goal, a new plan is constructed. The new plan avoids the difficult concept.

In the next sections, the architecture and functionality of the DCG are presented in more detail.

3.1 Content representation and dynamic planning

The Domain Structure contains the concept/topic structure of the subject knowledge 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, with an 'AND' arc, this means that both nodes A and B have to be taught when following the arcs from C. Otherwise, they would be considered as alternatives, i.e. there is a choice of nodes to be taught, either A or B. The arcs in the graph represent relationships between the concepts. These relationships can have various semantics. For example, if nodes A and B are connected with node C with an AND-relationship of type 'aggregation', this means that C contains sub-components A and B. If they are connected with an OR-relationship of type 'generalisation', this means that C is a general concept with possible instances A or B. There are many other possible semantic relationships, for example, causal, temporal, analogy, simple prerequisite, etc.

The simplest way to define a Domain Structure is to use only one possible semantic relationship, for example, to link domain concepts/topics with prerequisite links. In this way one obtains a curriculum-like structure that can be used to guide the sequencing of content. This structure was proposed first in [24] and can be seen appearing in the literature under different names: content model [25], pedagogical structure of the domain [26,27], or pedagogical content knowledge [28,29].

It is possible to organise the domain concepts/topics into a set of smaller, possibly interrelated AND/OR-graphs, representing relatively independent sub-areas of the domain, different 'views', or different levels of granularity.

Every node and every link from the Domain Structure is associated with a set of teaching materials (TMs), which instantiate different ways to teach the concept/topic (e.g. introduce, explain, give an example, exercise, or test). The Domain Structure is used for creating a plan of the course contents (a sub-graph of Domain Structure) to achieve a given teaching goal (concept). This plan is called '*Content Plan*' and the process – 'Content Planning'. During course execution TMs are selected by different instructional tasks to teach the concepts/topics to the student.

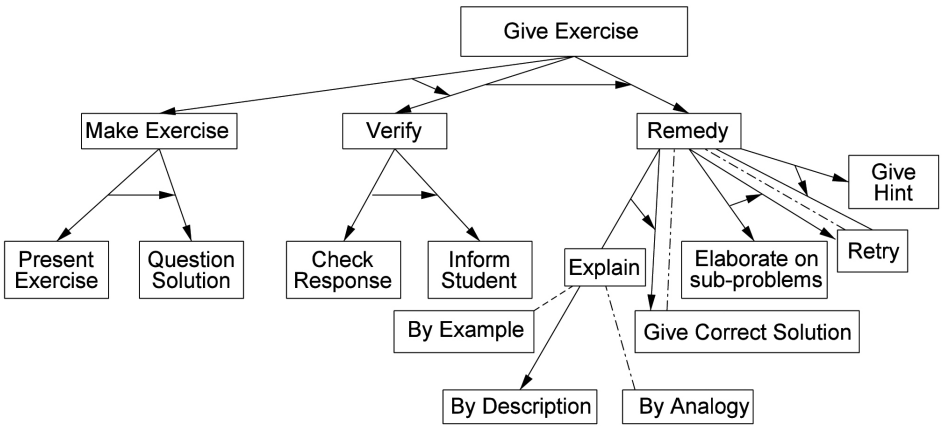
The teaching materials contain presentation and testing-units that carry out the communication with the student, i.e. they are in fact what the student sees on the screen.

Each TM is focused on a given concept or relationship. The TMs are classified according to their pedagogical function. Examples of types of TMs that can be used to teach a concept include an introduction, a motivating problem, an explanation, a help item, an exercise, or a test. In this sense TMs are equivalent to the 'instructional primitives' in Van Marcke's GTE system [30]. TMs 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, hint or help, explanation, eventually intermediate stages of solving the problem, etc. TMs of type 'test' have in addition two associated weights denoting to what extent the 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 Planning the presentation of a given concept

The DCG content planner by itself cannot decide *how* to present the selected contents (the current concept or relation) to the learner, i.e. what pedagogical type of TMs to select, or how to sequence several TMs to teach a given concept. . For this purpose another planning process in the DCG creates a *presentation plan* for each concept. This planning process uses a graph-representation of *Teaching Tasks* (see Figure 1) similar to the one proposed by Van Marcke [30], which expresses pedagogical knowledge, of how to teach a concept. Like the Domain Structure, the instructional task decompositions can be represented with AND/OR graphs, however, here the nodes represent teaching tasks and the links – task-decomposition methods.

Figure 1 An example of a teaching task hierarchy for the generic task 'give exercise'



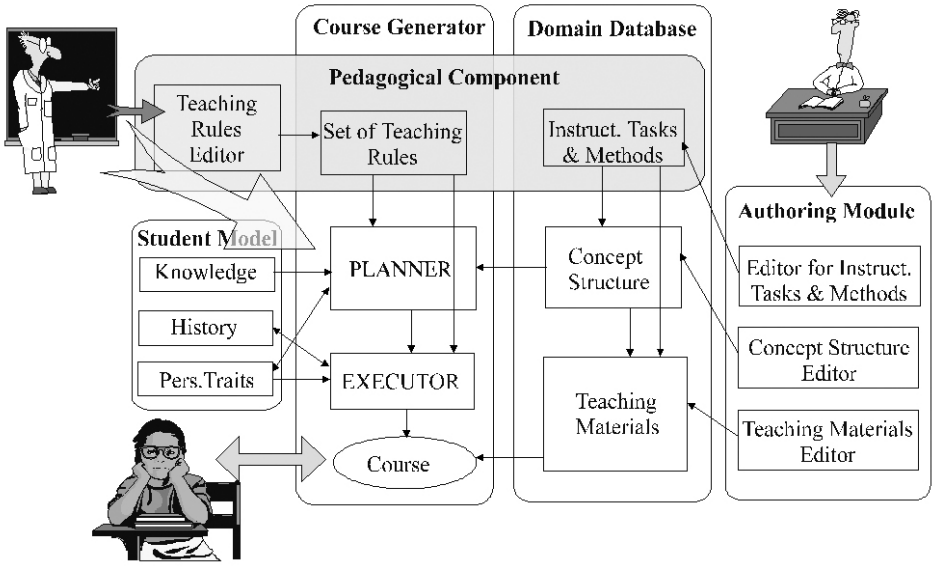
The AND-links in the teaching task structure represent links to sub-tasks of a certain task according to a certain task-decomposition method. The OR-links correspond to alternative task-decomposition methods. For example, Fig. 1 represents the generic task 'Give exercise' which can be decomposed into a sequence of the following sub-tasks:

'Make exercise', 'Verify', 'Remedy' (adapted from Van Marcke [30]). The sub-task 'Remedy' can be decomposed in different ways according to different methods (OR-types of links shown in Figure 1 with dotted lines).

A set of *Teaching Rules* manages the selection of content and presentation plans according to the cognitive style or learning preferences of the student [31,32]. Most of the rules are generic (i.e. domain independent).

DCG first decides which concepts will be taught, i.e. dynamically creates a *content* plan of the course. The representation of instructional tasks and methods allows the system to plan dynamically *how to present* the contents related to the current concept in a way suited to the learner, i.e. what types of TMs to select and how to sequence them. The full DCG architecture including pedagogical planning of the presentation is shown in Figure 2.

Figure 2 The DCG architecture: content and presentation planning



3.3 Dynamic course re-planning

During the presentation of the course to the student, if the student answers the test items correctly, i.e. demonstrates that he/she has acquired the concepts, he/she progresses along the course and no changes to the course are necessary. However, if the student fails to demonstrate knowledge of a concept, a re-planning of the course follows. Re-planning takes place first at the presentation level, i.e. an alternative sequence of teaching materials or pedagogical method for presentation of the concept is shown to the student. If the student fails again, the content planner generates a new sequence of concepts leading to the goal concept, starting from the current state of student knowledge as recorded in the student model.

3.4 Implementation and deployment of DCG

The modularity of the architecture allows the use of the system in various modes:

- 'one-shot' planning – creating a content plan for a student with particular existing knowledge and a teaching goal and no further adaptation to his/her individual progress during the course
- dynamic planning – creating a content plan as in the one-shot planning, but with the possibility to re-plan the contents of the course (i.e. the concepts/topics taught) during execution, if the student is not able to acquire a certain concept.

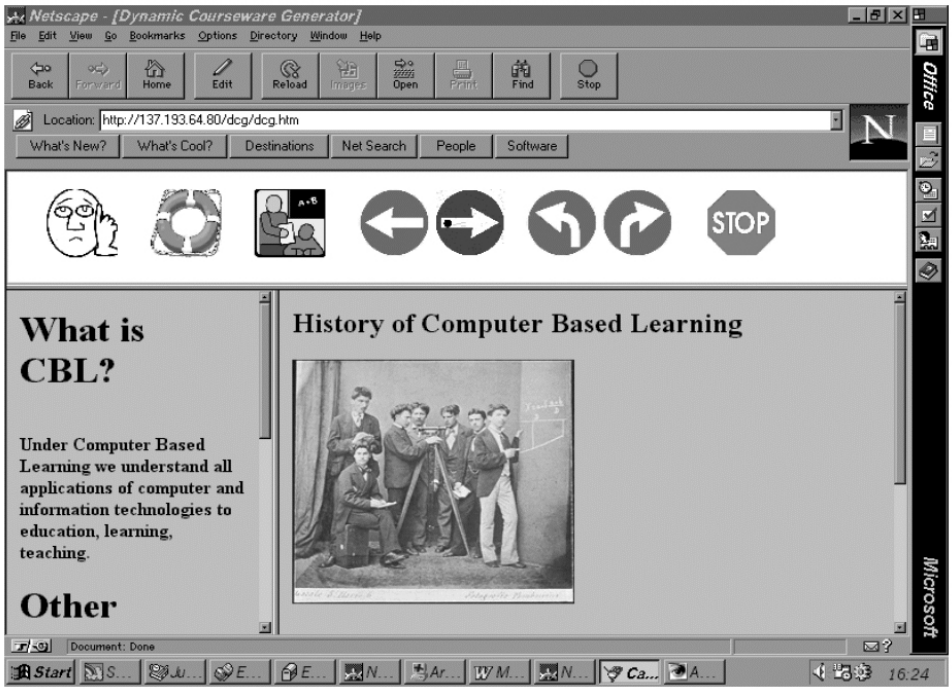
The DCG was implemented in several domains in the period 1995-97, including teaching about the structure and functioning of a simple electric toaster, teaching the theory of jazz, case-based medical diagnosis and training mechanical skills (typewriting). The platform was IBM PC 486 in a MS-Windows environment. The planner was implemented in C++ and the interfaces (authoring, student) – in OpenScript. *Asymetrix* ToolBook© was used as an authoring tool for creating the TMs since it allowed a fairly easy creation of TMs with advanced graphics and multimedia.

In order to evaluate the effort spent for creating an hour of instruction, the time spent for authoring was divided by the sum of the durations of all possible courses that can be generated by the system (with all possible teaching goals and several typical initial states of knowledge of students). Different results were obtained for the different domains, but all of them were less than 20 hours of authoring for one hour of instruction. This is quite a favourable result even in comparison with traditional CAL courseware, and especially in comparison with authoring for ITS, 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 for the generation of numerous alternative courses for different goals, the extra effort required to design and edit the Domain Structure seems well justified.

The one-shot and the dynamic modes of DCG were implemented and used successfully in a web-based educational system [33,34]. In this system, the planner, the domain knowledge structure and the permanent student model reside on the server. The student downloads an executor on the client together with the course plan and the TMs, which are web pages, spread throughout the WWW. The executor presents the TMs according to the plan and creates a temporary student model on the client, which tracks the progress of the student through the session.

The DCG on the web worked in the domain of 'computer based training' (Figure 3) and the authoring effort was approximately the same as in the full version – 18 hours of authoring for one hour of instruction.

Figure 3 The DCG+WWW interface



4 CoCoA: analysis and consistency checking of static web-based courses

Concept-based Courseware Analysis (CoCoA) is a course maintenance system developed at Carnegie Technology Education (http://www.carnegietech.org/). CoCoA can check the consistency and quality of a course at any moment of the course life and also assist course developers in some routine operations. The core of this system is a course-sequencing engine that works in an 'inverted way' to analyse the quality of sequencing in a static human-authored course. As in many other sequencing systems, the key to the sequencing power of CoCoA is a structured domain model and a refined approach to indexing the course material.

4.1 Domain modelling and content indexing

The core of the CoCoA framework is a domain model made of concepts – the elementary pieces of domain knowledge. The size of a concept is not fixed and may depend on the course. The course concepts are connected to form a heterarchy (it is not a hierarchy since it has a number of root nodes, each forming an overlapping hierarchy). To simplify the domain model authoring the concepts are connected using one non-typed parent-child link. This link has to express the value usually expressed by 'part-of' and 'attribute-of' links. The meaning of this link is simple – the knowledge of a parent concept is the

sum of knowledge of child concepts plus some 'integration' knowledge. Creating a parent-child network without the need to type links is relatively easy.

The domain concepts in CoCoA are used to index the course content, i.e. to connect elements of learning material called *learning items* (somewhat similar to the TMs in DCG) with the domain knowledge. There are several possible ways to index the content varying from very advanced and powerful to very simple.

The CoCoA approach is an extension of plain prerequisite-outcome concept indexing that was used in systems like Piano-Tutor [7] or InterBook [35]. Plain prerequisite-outcome indexing associates a teaching operation with two sets of concepts – prerequisite and outcome concepts. This approach does not distinguish among different types of teaching operations and allows only two roles in which a concept can be involved in a teaching operation: prerequisite and outcome. It also does not take into account relationships between concepts. Plain indexing has shown to be useful in simple domains or with coarse-grain level of domain modelling (all systems with plain indexing known to us to date use about 50 concepts).

The CoCoA approach uses relationships between concepts and two extensions of plain indexing: typed items and advanced concept roles. Typed items let the system distinguish among several types of teaching operations. Advanced concept roles can specify more roles of the learning items in regard to concepts. CoCoA is able to distinguish among several kinds of learning items – presentations, examples, assignments, and multiple-choice questions. The type of an item is a part of the index for the item. Concept-role pairs form the rest of the index. Four kinds of roles are used in CoCoA (in comparison with only two in InterBook and Piano-Tutor): light prerequisite, strong prerequisite, light outcome and strong outcome. Strong prerequisite or strong outcome means that 'deep' knowledge of a concept is demanded or produced by a learning item, while a light prerequisite or outcome deals with surface knowledge of a concept. These four roles were introduced to accommodate the needs of real courses.

Typed items and advanced concept roles let the course developer specify more knowledge about the content and support more powerful algorithms. The negative side of these extensions is increased authoring time. The increased authoring time could be a problem for a 'traditional' (single teacher) context of course development but it is justified in a context of large-scale web-based education. Here indexing expenses constitute a small fraction of overall course development expenses and are repaid by the possibility of helping course designers with developing and modifying courses. The rest of this section describes several kinds of courseware checking that were developed in the CoCoA system. All these kinds are powered by the rich knowledge representation described above and course sequencing algorithms.

4.2 Inverted sequencing for courseware engineering

Prerequisite checking. Prerequisite checking is one of the key benefits of concept indexing. It is important for original course design as well as for a redesign when learning items are moved or changed. With multiple-level indexing we are able to check prerequisites for all learning items. In CoCoA, prerequisite checking for linear courses is performed by a sequencing engine that simulates the process of teaching with an overlay student model. It starts with an empty overlay model, scans learning items in the order specified by the author, updates the student model, and checks the match between the

current state of the model and each subsequent item. The following prerequisite consistency rules can be checked:

- *Presentation prerequisites*: a presentation item can be understood because all prerequisite concepts are already presented up to the required level.
- *Question prerequisites*: all concepts involved in all questions designed for a presentation page are learned at least up to the advanced level when the page is completed.
- *Example prerequisites*: all concepts involved in an example are learned to the required level within the section where the example is presented or before; strong prerequisite concepts are learned at least up to the advanced level, weak prerequisite concepts are learned at least up to the surface level.
- *Exercise prerequisites*: at the point where an exercise is presented, all strong prerequisite concepts are learned and demonstrated with examples, all weak prerequisite concepts are either learned or demonstrated with examples.

The prerequisite checking on the level of course items is especially important for programming courses that usually have very few direct prerequisite relationships between concepts. Since most programming concepts could be introduced independently from other concepts, there are many conceptually possible ways to teach the same subject. However, adopting a particular approach to teaching the subject usually results in invisible indirect prerequisites 'hardwired' into educational material. One example of indirect prerequisites is presentation-level prerequisites: a concept *A* does not depend on concept *B*, but the method of presentation of *A* chosen by the author required understanding of *B*. For example, an author may decide to present '*for* loop' in comparison with '*while* loop', thus creating an indirect prerequisite link from '*while* loop' to '*for* loop'. Another case is example-level or problem-level prerequisites. *A* concept *A* does not depend on concept *B* and could be learned either before or after *B*. However, in the current course material all available examples or exercises that use *B* also include *A*. As a result, the material requires *A* to be learned *before B*. All these kinds of prerequisites are very hard for developers to keep in mind. The only way to ensure that the course is built or redesigned with no prerequisite conflicts is careful prerequisite checking.

Finding content 'holes'. A failure to meet the prerequisites could mean either a problem with structure (the item that could meet the prerequisite does exist in the courses but is placed after the item being checked) or a problem with content (no item to cover the prerequisite). The system can distinguish these two cases and provide a helpful report of a problem. While the former problem could often be resolved by restructuring the material, the latter indicates a need to expand the course material.

Consolidation of presentations. In a well-designed course each concept has to be fully presented in a single place (subsection or section). It is the place where the student will be returning to refill the gaps in his/her knowledge of a concept. This place is called the concept *host section*. A concept could be introduced before its host section (to enable the student to learn or practice other concepts), but never more than twice and not after the full presentation. The system can check these rules using indexing. (Note: The same is not true about examples. It is desirable to have several examples for each concept.)

Question placement and repositioning. Well-designed questions have one or two outcome concepts (the question's goal). The system can automatically place new questions into the proper place in the course by finding the host section of the question goal. With automatic placement, course and question design can be delegated to several authors without the loss of consistency. If the course is restructured the questions can be automatically repositioned.

Guidelines for question design. By matching concepts presented in a section and concepts assessed by the section question pool it is easy to identify a set of concepts that cannot be assessed. The identified deficit could drive the question design process. The same procedure can also ensure that the questions in the pool are reasonably evenly distributed among the section concepts (to avoid the situation where, for example, 80% of the questions are testing 20% of the concepts).

Matching presentations with examples and exercises. It is possible to check to what extent examples and exercises match their place in the course and to what extent they cover the presented content. This checking can be done by matching the set of concepts presented in the section with the joint sets of goal concepts of exercises and examples located in this section. In an ideal situation each section should present, demonstrate (by examples) and assess (by exercises) the same sets of concepts. If there are too many concepts that are presented but not covered by examples or exercises, the coverage is low. If there are too many concepts that are covered by exercises or examples but not presented in the section (if there is no prerequisite conflict they could be simply presented in previous sections) then the relevance is low. A minor mismatch between presentations, examples, and concepts is not a problem, but a major mismatch in either direction is a sign of a poorly designed section and an indication that something has to be redesigned.

Checking course design goals against the real course. An author could start the course design with a design document that lists all essential concepts to be introduced in each section. The design document could be stored separately from the course. The system can check how the real course matches the original design by comparing where the author planned to introduce the key concepts and where they were really introduced; how the set of target concepts is supported by questions, examples and exercises.

Presentation density and sectioning. While different concepts may require different amounts of presentation, the conceptual complexity of a content fragment could be measured by the number of concepts presented in it. By controlling the number of concepts presented in each section we can identify two types of problems: presentation density, where too many concepts are presented in a relatively short section, and uneven distribution of content where the number of concepts presented in subsections of the same level significantly differs.

Controlling the difficulty of examples and exercises. Prerequisite indexing of exercises and examples specifies minimal preparatory requirements for the concept level. It is normal, however that when starting an exercise or an example some concepts have a higher knowledge level than is demanded by prerequisites. For example, a strong prerequisite concept of an example has to be learned up to the advanced level. In real life, a student can encounter this exercise when he or she has already seen several examples with this concept or even solved an exercise involving this concept. In this situation, the

exercise is easier for that student. Generally, we can estimate the difficulty of a learning item by measuring the difficulty difference between the target state of the goal concepts and the starting state. If all goal concepts of an exercise have already been used in earlier solved exercises, the exercise is quite simple. If none of them has been used in examples, the exercise is very difficult. Thus, the difficulty of an exercise is not a constant – it depends on the placement of the exercise in the course. It makes sense to control the difficulty of examples and exercises to make sure that no example or exercise is too simple or too difficult.

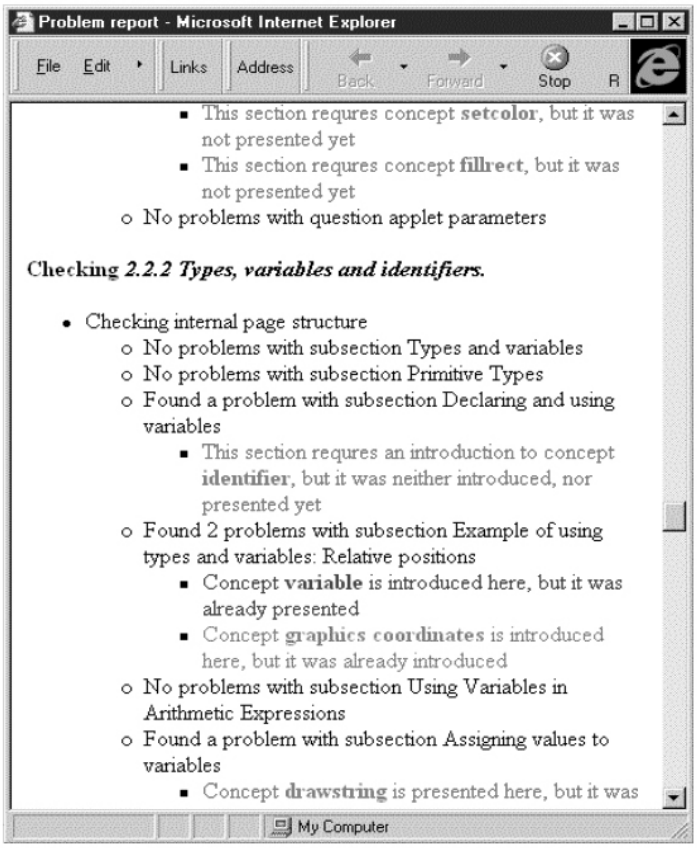
There is research evidence that there exists an optimal difficulty of a learning item for each individual student (i.e., that the student learns best when he or she is presented with learning items of near optimal difficulty). It is quite likely that different groups of users can handle different difficulties. CoCoA's tools for controlling the difficulty of examples and exercises could be used for making courses with levels of difficulty targeted at different categories of users.

4.3 Implementation and first experience

The first version of the system was developed in Java and evaluated on real courses developed by Carnegie Technology Education (CTE) in collaboration with Carnegie Mellon University faculty. With the help of the system it was possible to find and fix a number of problems. The first version supported prerequisite checking, finding content 'holes', consolidation of presentations and question placement and repositioning. Since the original learning contents in CTE courses were not indexed with metadata and the CMS used by CTE had no place for them, CoCoA required the author to specify the course structure along with concept tags in a separate file. The situation with question indexing was different – here concept tags were stored as parts of the questions. Various checking procedures could be called using a Java command line interface.

The first version of the system was used to check two real courses. While the system turned out to be very useful, we encountered a problem. In addition to revealing a substantial number of real large and small hidden problems the system has also reported a number of problems that no real teacher would consider a problem. It turned out that the course consistency rules behind the system are too rigid. In real life teachers and students can tolerate a number of small inconsistencies in the course. Moreover, in some cases the course may be designed formally 'inconsistent' for a reason. A teacher may want to provoke student thinking by presenting an example that is based on material that has not yet been presented, but could be understood by analogy with the learned material. To respond to this problem, the second version of the system applied colour coding in the course problem report (Figure 4). In particular, messages that report real problems in the course are coloured red – not to be missed. On the other hand, messages reporting problems that often may be tolerable are coloured green. We used three to four colours in our reports.

Figure 4 A fragment of a problem report for a Java course



5 Main steps in implementing sequencing mechanisms in large-scale web-based education courses

While DCG and CoCoA systems are quite different they share many common features. Studying the common parts of these two systems is important in understanding which components are required of any sequencing system to be used in large-scale web-based education. The most important similarity can be seen in knowledge structuring used by DCG and CoCoA to represent knowledge about the domain to be taught and the learning material.

The heart of both DCG and CoCoA knowledge representation is a structured *domain model* that is composed of a set of small domain knowledge elements (DKEs). Each DKE represents an elementary fragment of knowledge for the given domain. This model is used in all known sequencing engines. In different systems DKEs are named differently – concepts, knowledge items, topics, knowledge elements, learning objectives, learning outcomes, but in all cases they denote elementary fragments of domain knowledge. For simplicity, we will be calling these DKE concepts. Depending on the domain, the application area, and the choice of the designer, concepts can represent bigger or smaller

pieces of domain knowledge (see Figure 5). Domain concepts form *a domain model*. The simplest form of the domain model is a model without links between concepts. We call it a *set model* or *vector model* since the set of concepts has no internal structure. In the more advanced kind of domain model, *concepts* are related to each other, thus forming a kind of semantic network. This network represents the structure of the domain covered by a system or a course. We call this kind of model a *network model* (Figure 6). Note that both DCG and CoCoA use a network domain model (AND-OR graph in the case of DCG and heterarchy in the case of CoCoA). While a vector model can also be applied for sequencing needs [7,35], a network model is more powerful and has a wider applicability.

Figure 5 An example of concept structure showing two levels of detail. The concept 'Repetition' is expanded on a lower level of detail. The concepts are connected with prerequisite links

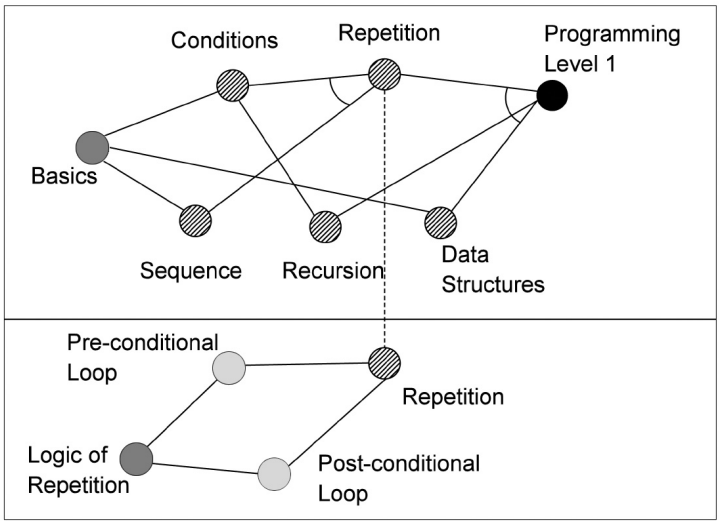
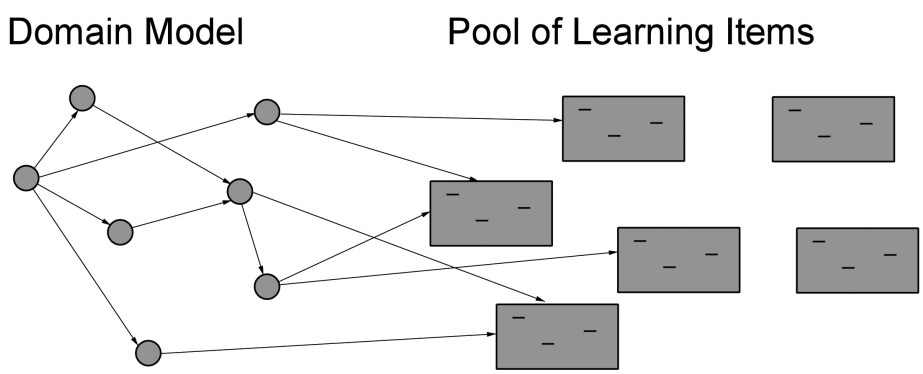


Figure 6 Bridging the gap between the world of knowledge and the world of learning materials. A network domain model is connected with learning items by multi-concept indexing

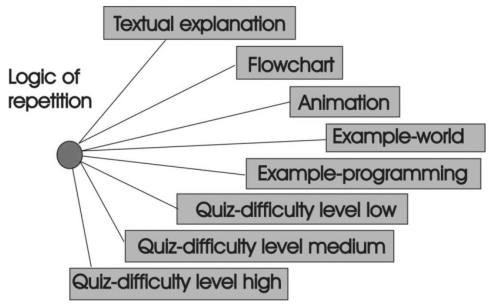


The second key to sequencing is a connection between domain model concepts and fragments of learning material, called *teaching materials* in DCG and *learning items* in CoCoA. For a sequencing engine, the fragments of learning material are not black boxes as they are in traditional courses, but teaching tools that the engine can skilfully handle. The sequencing engine sees the knowledge behind the teaching material. This is possible since every fragment is connected (or indexed) with elements of domain knowledge, i.e., with concepts.

There are many known ways of indexing – a review can be found in [36]. For the purpose of this paper it is enough to distinguish two major approaches: single-concept indexing in which each fragment of educational material is related to one and only one domain model concept (Figure 7), and multi-concept indexing in which each fragment can be related to many concepts (Figure 6). While DCG uses single-concept indexing, CoCoA uses multi-concept indexing. This choice does not stem from differences between the applications of the systems, but rather reflects two different indexing approaches explored by the authors in the past [8,10]. Both approaches are popular in various sequencing systems. Single-concept indexing is simpler and more intuitive for the authors. Multi-concept indexing is more powerful, but it makes the system more complex and requires more highly skilled authoring teams. It is probably a good idea to choose single-concept indexing whenever it is meaningful from the educational point of view (i.e., in smaller systems and simpler domains). At the same time, in many cases, using multi-concept indexing is imposed by the nature of the domain. For example, in programming and mathematics, elementary constructs and operators are often selected as domain model concepts. In that case a hypermedia system that needs to have reasonably precise indexing must use a multi-concept indexing approach, since most examples and problems involve several constructs and operators.

The final touch in knowledge representation is to represent more knowledge about fragments of learning material in addition to their connection to concepts. Some advanced sequencing systems represent and use different additional information such as duration, complexity, or type of a learning item (Figure 7). Of all these, both CoCoA and DCG choose to represent one aspect – the (pedagogical) type of the item, i.e., a presentation, an introduction, a question, an example, a test item, etc. While the use of this information in CoCoA and DCG is quite different, both authors believe that knowing the type of every learning item is very beneficial for both dynamic sequencing and consistency checking. Besides, this kind of knowledge is very easy for an author to provide.

Figure 7 Example of single-concept indexing. The concept 'Logic of repetition' is related with several learning materials of different type



The three types of knowledge about the domain and the learning material listed above are used in many (though not all) known sequencing engines and the authors believe that they are essential for making sequencing work in the context of large-scale web-based education.

6 The future of course sequencing in practical web-based education

This paper presents three meaningful approaches for using sequencing technologies in the context of practical web-based education: two of them based on dynamic planning of the course content and presentation as in DCG, and one based on the sequencing technique to verify the consistency of traditionally authored courses, as in CoCoA. In this paper we demonstrate that despite the sharp difference between the modern CMSs currently used to create and host the majority of web-based courses and the 'intelligent' systems with course sequencing, there are several ways to use adaptive course sequencing in modern web-based education. Another issue, however, is whether and when this will happen. In this context, it is important to consider the differences between the three approaches presented in the paper.

There is no doubt that dynamic courseware generation in the form supported by the fully functional DCG system can provide most benefits by building a personalised course for every learner. At the same time, while the DCG approach fits technically to large-scale WBE context, it has two problems. The first problem is that in most places web-based education is still class-based. A virtual class is still a class. The students from the same class have to learn the same material in about the same time and even take exams on the same date. As we noted in section 2, some advanced group-level planning coordination is required to make individually generated courses work in a class-based WBE. This kinds of 'group-coordinated individual generation is possible, but has not yet been developed and investigated. In this context a one-shot generation of a course adapted to a class of users provides a cheaper alternative. While the product of generation should be called 'customised course' rather than 'adaptive course', this approach allows a fair level of individualisation, especially in the case of reasonably homogeneous classes. We believe that systems that can produce courses on demand from the same body of teaching material would be very popular in the future, since they will enable a course provider to accommodate the needs of different customers at a fairly small cost.

Still, the major obstacle for both DCG and CoCoA is the initial knowledge representation (the concepts structure and indexing of teaching materials with respect to concepts, roles, etc.). Bootstrapping a system like DCG is quite expensive. To produce the first customised course, a provider needs to have a reasonably large database of well-indexed learning material (at least, two to three times larger than the size of a typical course being produced). The start-up price of developing and indexing a pool of rich course material could be an obstacle to using a DCG-like approach for a small company and it may not be a worthwhile investment for a small number of students or for a student population that is relatively homogeneous.

In this situation another difference between the three approaches has to be taken into account. The CoCoA approach requires minimal investment into preparation of an indexed material – only the items already included in the course have to be indexed. Moreover, the CoCoA approach can be used incrementally with partially indexed

material. The larger the proportion of indexed material, the more problems could be discovered by CoCoA. Therefore it is likely that courseware-checking approaches similar to CoCoA will be the first to be used in the context of large-scale courses. They will provide an immediate benefit to the authoring teams. The outcome of this process is not only consistent courses of higher quality, but also a large volume of carefully indexed learning material.

The authors are very optimistic about the future of course sequencing approaches in large-scale web-based educational systems, since indexing learning materials with metadata is becoming an important trend in practical web-based education. This trend has been fuelled by recent works on courseware reuse, learning pools, learning object libraries and metadata standards [17,37,38]. We expect that large volumes of consistently indexed learning materials will decrease the bootstrapping cost for more flexible sequencing technologies. The course developers in large courseware publishing teams will realise the fact that when the variety of courses that can be generated can meet the various needs of the learners, the cost compares favourably to traditional authoring. We hope that this process will eventually lead to the acceptance of more flexible approaches in large-scale web-based education such as DCG-like adaptive course generation and full-scale courseware sequencing [8,9].

Acknowledgements

The authors thank Helen Bretzke, Christopher Cox, Darina Dicheva and the anonymous reviewers for their comments on the earlier versions of this paper.

References

1 Barr, A., Beard, M. and Atkinson, R.C. (1976) 'The computer as tutorial laboratory: the Stanford BIP project', *International Journal on the Man-Machine Studies,* Vol. 8, No. 5, pp.567–596.

2 McArthur, D., Stasz, C., Hotta, J., Peter, O. and Burdorf, C. (1988) 'Skill-oriented task sequencing in an intelligent tutor for basic algebra', *Instructional Science,* Vol. 17, No. 4, pp.281–307.

3 Brusilovsky, V. (1993) 'Task sequencing in an intelligent learning environment for calculus', *in Proc. of Seventh International PEG Conference*, Edinburgh, pp.57–62.

4 Eliot, C., Neiman, D. and Lamar, M. (1997) 'Medtec: a web-based intelligent tutor for basic anatomy', in S. Lobodzinski and I. Tomek (Eds.) *Proc. of WebNet'97, World Conference of the WWW, Internet and Intranet*, Toronto, Canada, AACE pp.161–165.

5 Rios, A., Millán, E., Trella, M., Pérez, J.L. and Conejo, R. (1999) 'Internet based evaluation system', in S.P. Lajoie and M. Vivet (Eds.) *Artificial Intelligence in Education: Open Learning Environments*, IOS Press, Amsterdam, pp.387–394.

6 Brusilovsky, P. (1994) 'ILEARN: an intelligent system for teaching and learning about UNIX', in *Proc. of SUUG International Open Systems Conference*, Moscow, Russia, ICSTI pp.35–41.

7 Capell, P. and Dannenberg, R.B. (1993) 'Instructional design and intelligent tutoring: theory and the precision of design', *Journal of Artificial Intelligence in Education,* Vol. 4, No. 1, pp.95–121.

8 Brusilovsky, P. L. (1992) 'A framework for intelligent knowledge sequencing and task sequencing', in C. Frasson, G. Gauthier and G.I. McCalla (Eds.) *Intelligent Tutoring Systems*, Springer-Verlag, Berlin, pp.499–506.

9 Khuwaja, R., Desmarais, M. and Cheng, R. (1996) 'Intelligent guide: combining user knowledge assessment with pedagogical guidance', in C. Frasson, G. Gauthier and A. Lesgold (Eds.) *Intelligent Tutoring Systems*, *Lecture Notes in Computer Science*, Springer Verlag, Berlin, Vol. 1086, pp.225–233.

10 Vassileva, J. (1992) 'Dynamic CAL-courseware generation within an ITS-shell architecture', in I. Tomek. (Ed.) *Computer Assisted Learning. Lecture Notes in Computer Science*, Vol. 602, Springer-Verlag, Berlin, pp.581–591.

11 Brusilovsky, P. (1999) 'Adaptive and intelligent technologies for web-based education', *Künstliche Intelligenz*, Vol. 4 pp.19-25,
available online at http://www2.sis.pitt.edu/~peterb/papers/KI-review.html.

12 Brusilovsky, P., Schwarz, E. and Weber, G. (1996) 'ELM-ART: an intelligent tutoring system on World Wide Web', in C. Frasson, G. Gauthier, and A. Lesgold (Eds.) *Intelligent Tutoring Systems. Lecture Notes in Computer Science*, Springer Verlag, Berlin, Vol. 1086, pp.261–269.

13 De Bra, P. and Calvi, L. (1998) '2L670: a flexible adaptive hypertext courseware system', in K. Grønbæk, E. Mylonas and F.M. Shipman III (Eds.) *Proc. of Ninth ACM International Hypertext Conference (Hypertext'98*), Pittsburgh, USA, ACM Press, pp.283–284.

14 WBT Systems (1999) *TopClass*, Dublin, Ireland, WBT Systems, available online at http://www.wbtsystems.com/

15 Blackboard Inc. (2002) *Blackboard Course Management System*, Blackboard Inc., available online at http://www.blackboard.com/

16 WebCT (2002) *WebCT Course Management System*, Lynnfield, MA, WebCT, Inc., available online at http://www.webct.com

17 Verhoeven, B., Cardinaels, K., Van Durm, R., Duval, E. and Olivié, H. (2001) 'Experiences with the ARIADNE pedagogical document repository', in *Proc. of ED-MEDIA'2001 – World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Tampere, Finland, AACE, pp.1949–1954.

18 Ahanger, G. and Little, T.D.C. (1997) 'Easy Ed: an integration of technologies for multimedia education', in S. Lobodzinski and I. Tomek (Eds.) *Proc. of WebNet'97, World Conference of the WWW, Internet and Intranet*, Toronto, Canada, AACE, pp.15–20.

19 Caumanns, J. (1998) 'A bottom-up approach to multimedia teachware', in B.P. Goettl, H.M. Halff, C.L. Redfield and V.J. Shute (Eds.) *Intelligent Tutoring Systems*, Springer-Verlag, Berlin, pp.116–125.

20 Kettel, L., Thomson, J. and Greer, J.(2000) 'Generating individualized hypermedia applications', in *Proc. of Workshop on Adaptive and Intelligent Web-based Education Systems at 5th International Conference on Intelligent Tutoring Systems (ITS'2000),* Montreal, Canada.

21 Masthoff, J. (2002) 'Automatic generation of a navigation structure for adaptive web-based instruction', in P. Brusilovsky, N. Henze and E. Millán (Eds.) *Proc. of Workshop on Adaptive Systems for Web-Based Education at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2002) Proceedings,* Málaga, Spain, pp.81–91.

22 Melis, E. Andres, E., Büdenberder, J., Frishauf, A., Goguadse, G., Libbrecht, P., Pollet, M. and Ullrich, C. (2001) 'ActiveMath: a generic and adaptive web-based learning environment', *International Journal of Artificial Intelligence in Education,* Vol. 12, No. 4, pp.385-407.

23 Diessel, T., Lehmann, A. and Vassileva, J. (1994) 'Individualised course generation: a marriage between CAL and ICAL', *Computers and Education,* Vol. 22, Nos.1/2, pp.57–64.

24 Peachey, D.R. and McCalla, G.I. (1986) 'Using planning techniques in intelligent tutoring systems', *International Journal on the Man-Machine Studies,* Vol. 24, pp.77–98.

25 Van Marcke, K.(1992) 'Instructional expertise', in C. Frasson, G. Gauthier. and G.I. McCalla, (Eds.) *Proc. of Second International Conference, ITS'92*, Berlin, Springer-Verlag, pp.234–243.

26 Mitrovic, A., Djordjevic, S. and Stoimenov, L. (1996) 'INSTRUCT: modeling students by asking questions', *User Modeling and User Adapted Interaction,* Vol. 6, No. 4, pp.273–302.

27 Vassileva, J. (1990) 'A classification and synthesis of student modelling techniques in intelligent computer-assisted instruction', in D.H. Norrie and H.W. Six (Eds.) *Proc. of 3rd International Conference, ICCAL'90,* Berlin, Springer-Verlag, pp.202–213.

28 Calderhead, J. (1991) 'Representations of teachers' knowledge', in P. Goodyear (Ed.) *Teaching Knowledge and Intelligent Tutoring*, Ablex, Norwood, N.J., pp.269–278.

29 Leinhardt, G. (1998) 'Situated knowledge and expertise in teaching', in J. Calderhead (Ed.) *Teachers' Professional Training*, Falmer, London, pp.146–168.

30 Van Marcke, K. (1992) 'A generic task model for instruction', in S. Dijkstra (Ed.) *Instructional Models for Computer-Based Learning Environments, NATO ASI Series*, Vol. F104, Springer-Verlag, Berlin, pp.234-243.

31 Vassileva, J. (1995) 'Dynamic courseware generation: at the cross point of CAL, ITS and authoring', in *Proc. of International Conference on Computers in Education, ICCE'95*, Singapore, AACE, pp.290–297.

32 Vassileva, J. (1998) DCG + GTE 'Dynamic courseware generation with teaching expertise', *Instructional Science,* Vol. 26, Nos. 3/4, pp.317–332.

33 Vassileva, J. (1997) 'Dynamic course generation on the WWW', in B.du Boulay, and R. Mizoguchi (Eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems*, IOS, Amsterdam, pp.498–505.

34 Vassileva, J. and Deters, R. (1998) 'Dynamic courseware generation on the WWW', *British Journal of Educational Technology,* Vol. 29, 1, pp.5–14.

35 Brusilovsky, P., Eklund, J. and Schwarz, E. (1998) 'Web-based education for all: a tool for developing adaptive courseware', *Computer Networks and ISDN Systems*, Vol. 30, Nos. 1–7, pp.291–300.

36 Brusilovsky, P. (2002) 'Developing adaptive educational hypermedia systems: from design models to authoring tools', in T. Murray, S. Blessing and S. Ainsworth (Eds.) *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive, and Intelligent Educational Software,* Ablex, Norwood, In Press.

37 IEEE LTCS WG11(2001) *The Semantic Document v3.4*, Computer Managed Instruction Working Group of the IEEE Learning Technology Standards Committee, available online at http://ltsc.ieee.org/wg11/index.html

38 IEEE LTCS WG12 (2001) *LOM: Working Draft Document v6.1*, Learning Object Metadata Working Group of the IEEE Learning Technology Standards Committee, available online at http://ltsc.ieee.org/wg12/doc.html