Course Sequencing for Static Courses? Applying ITS Techniques in Large-Scale Web-based Education

Peter Brusilovsky

Carnegie Technology Education and
HCI Institute, Carnegie Mellon University
4615 Forbes Avenue, Pittsburgh, PA 15213, USA
plb@cs.cmu.edu

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 as a core technology for concept-based course maintenance. This paper describes a concept-based course maintenance system that we have developed for Carnegie Technology Education. The system can check the consistency and quality of a course at any moment of its life and also assist course developers in some routine operations. The core of this system is a refined approach to indexing the course material and a set of "scripts" for performing different operations.

1 Introduction

Course sequencing is one of the oldest technology in the field of intelligent tutoring systems (ITS). The idea of course sequencing is to generate an individualized course for each student by dynamically selecting the most optimal teaching operation (presentation, example, question, or problem) at any moment of education. An ITS with course sequencing represents knowledge about the subject as a network of concepts where each concept represents a small pieces of subject knowledge. The learning material is stored in a database of teaching operations. Each teaching operation is indexed by concepts it deals with. The driving force behind any sequencing mechanism is a student model that is a weighed 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 data base 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 ITS can sequence only one kind of teaching operations. For example, a number of sequencing systems including the oldest sequencing systems [2; 14] and some others [8; 12; 15] can only manipulate the order of problems or questions. In this case it is usually called task sequencing. A number of systems can

do sequencing of *lessons* that are reasonably big chunks of educational material complete with presentation and assessment [3; 9]. Most advanced systems are able to sequence several kinds of teaching operations such as presentation, examples, and assessments [7; 13].

One could say that sequencing is an excellent technology for distance education. In the situation where students can learn the subject at their own pace, it looks like a great idea to have each student to learn the subject by the most beneficial individualized way. Indeed, sequencing is now the most popular technology in research-level Web-based ITS [4]. However, there is a significant distance 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 like ELM-ART [Brusilovsky, 1996 #732] and 2L670 [De Bra, 1998 #1178]. In a modern large-scale Web-based education context a single course provider operates tens to hundreds of courses that has to be delivered to thousands of students grouped in classes. The biggest concern of a provider is the problem of maintenance. To avoid problems with installing and supporting multiple Web-based education (WBE) systems and teaching the stuff how to use these systems, all serious providers tend to choose one single course management system (CMI). Naturally the providers are choosing modern commercial CMIs such as TopClass [17] or WebCT [18] that can support main needs of a course provider from course material delivery to discussion forums to generation of various course reports. Unfortunately, current CMI systems leave no space for dynamic sequencing. The course model behind all these systems is a static sequence (or a tree) of modules followed by static quizzes and assignments.

Could we find any use for the course sequencing ideas in this rigid context of large-scale Web-based education? The answer is yes. We can suggest at least two meaningful ways to do “sequencing of static courses”. First way is dynamic generation of the course before the students hit it. Instead of generating a course incrementally piece by piece, as in traditional sequencing context, the whole course could be generated in one shot. While courses produced by this one-shot generation are not as adaptive as incrementally sequenced courses, they still could be very well tuned for individual students taking into account their individual learning needs and starting level of knowledge. A good example of this approach is DCG system [16]. A similar approach was also described in [1; 10]. Since a course generated with a DCG-like system is static, it could be delivered by a regular CMI system.

While DCG-like approach fits technically to large-scale WBE context, it still has two problems. First problem is that in most places Web-based education is still class-based. 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 at the same date. Naturally, for a class-based WBE an individually generated course will not work. This problem could be solved relatively easy by generating courses that are adapted to the whole class of users. While the product of generation should be rather called customized course than adaptive course, this approach allows a very good level of individualization, especially for the case of reasonably homogeneous classes. We think that in the future systems that can produce courses on demand from the same body of teaching material would be very popular since that will enable a course

provider to accommodate to the needs of different customers. A DCG-like approach has, however, another problem – a bootstrapping one. To produce the first customized course a provider need 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 startup price of developing such a rich course in addition to the price of the system is a big obstacle to using DCG-like approach.

The second approach to static sequencing suggested in this paper is least ambitious. We suggest to use a course sequencing mechanism as a core of a course maintenance system for static courses developed by a team of authors in a traditional way. The very idea is simple. Since a sequencing mechanism can evaluate several possible options for the "next steps" (i.e., presentation, example, assignment) in a dynamic course and select the best one, it can also check whether the predefined "next step" in a static course is a good one. If the next step is not really good, it can report problems. For example, it can find a situation when an assessment requires knowledge that are not presented yet or, vice versa, when presented knowledge are never assessed. These kinds of course checking are absolutely necessary for any serious course developer team such as Carnegie Technology Education, a WBE "arm" of Carnegie Mellon University. Large-scale modern courses include hundred to thousands of learning items that are produced by a team of developers. Through the life 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 for software engineering.

This paper describes a concept-based course maintenance system developed at Carnegie Technology Education. The system can check the consistency and quality of a course at any moment of course life and also assist course developers in some routine operations. The core of this system is a refined approach to indexing the course material and a set of "scripts" for performing different operations. Next section describes the indexing part and the section after that talks about scripts. We conclude with some speculation about prospects of our work.

2 Content indexing

There are several possible ways to index the content from very advanced and powerful to very simple. The reported approach supports the functionality that we find essential while being still simple enough to used by course developers.

The simplest indexing approach could be referred as "plain" prerequisite-outcome concept indexing. It is used in systems like Piano-Tutor [9] or InterBook [5]. Plain indexing associate a teaching operation with two sets of concepts - prerequisite and outcome concepts. Plain approach does not distinguish different types of teaching operations and use 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 modeling (all systems with plain indexing known to the author use about 50 concepts).

The reported approach uses three extensions of plain indexing approach: typed items, advanced concept roles, and links between concepts. Typed items let the system distinguish several types of teaching operations. Advanced concept roles can specify more roles of the teaching operations in regard to concepts. Both mechanisms let the course developer specify more knowledge about the content and support more powerful algorithms. The impact of links between concepts is a more precise student modeling, prerequisite tracking, and richer navigation. Negative side of all three extensions is increased authoring time. In particular, developing a connected concept model of a domain takes considerable time of several domain experts. 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 to help course designers with developing and modifying courses.

The core of our framework is formed by concepts – elementary pieces of learning material. The size of a concept is not fixed and may depend of a course. We have several kinds of teaching operations in our courses – presentations, examples, assignments, and multiple-choice questions. The type of the item is a part of the index for the item. Concept-role pairs form the rest of the index. We use four kinds of roles (in comparison with only two in InterBook and Piano-Tutor): light prerequisite, strong prerequisite, light outcome and strong outcome. In comparison with “real” or strong prerequisites and outcomes that tells that “deep” knowledge of a concept are produced or demanded by a learning item, the light prerequisites and outcomes deal with surface knowledge about a concept. We have to introduce these four roles to accommodate the needs of real courses.

The course concepts are connected to form a heterarchy. We use one non-typed parent-child link. This link has to express the value usually expressed by “part-of” and “attribute-of” links. Creating a parent-child hierarchy without the need to type links is relatively easy The meaning of this hierarchy is simple – the knowledge of a parent concept is a sum of knowledge of child concepts plus some “integration extra”.

3 The use of indexing for courseware engineering

3.1 Prerequisite checking

Prerequisite checking is the 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. Prerequisite check for linear courses is performed by a sequencing engine that simulates the process of teaching with a student model. It 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 following item. The following prerequisite problems could 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 into 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 into an example are learned to the required level right in the section where an 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 at 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. Most of programming concepts could be introduced independently from other concepts. That's why there could be many possible ways to teach the same subject. However, adopting a particular approach to teaching the subject usually results in lots of indirect prerequisites "hardwired" into educational material. One example of indirect prerequisites is presentation-level prerequisites: A concept *A* does not depend of concept *B*, but the way of presentation of *A* chosen by the author required understanding of *B*. Another case is example-level or problem-level prerequisites. A concept *A* does not depend of 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 to keep in mind. The only way to ensure that the course is built or redesigned with no prerequisite conflicts is careful prerequisite checking.

3.2 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 placed after the checked item) 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 be often resolved by restructuring the material, the latter indicates a need to expand the course material.

3.3 Consolidation of presentations

In a well-designed course each concept has to be presented in full 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 hardly 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's quite desirable to have several examples for each concept).

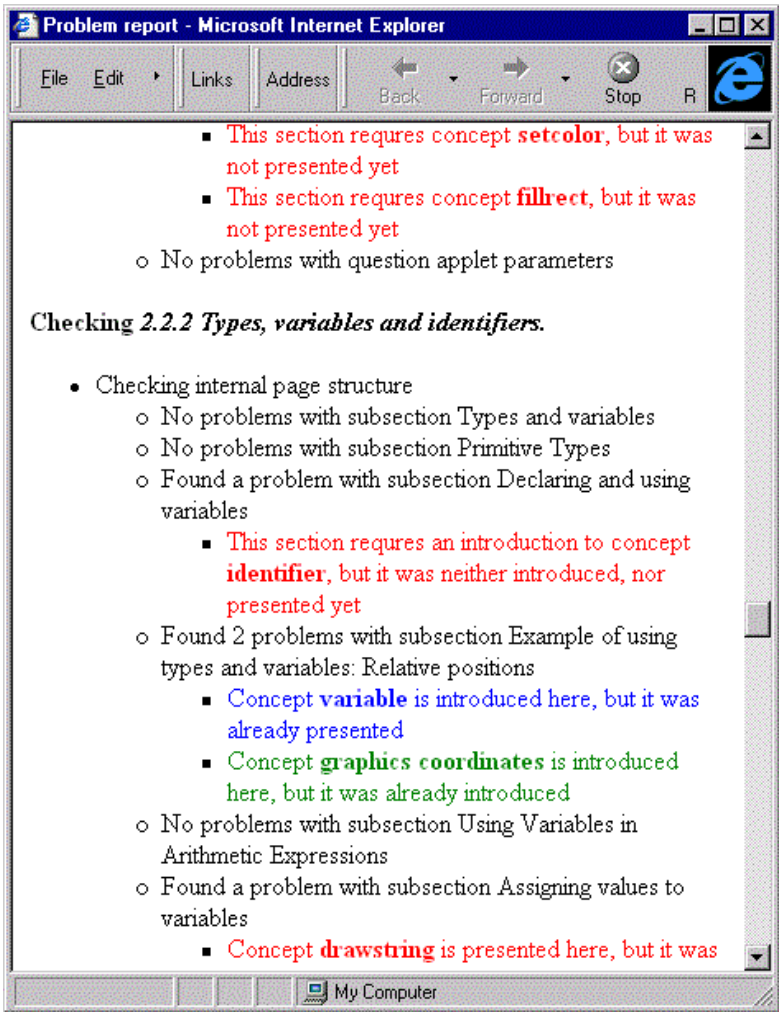

Fig. 1. A fragment of a problem report for a Java course

3.4 Question placement and repositioning

Well-designed questions have one or two outcome concepts (question 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 re-structured the questions can be automatically repositioned.

3.5 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 can never be assessed. The identified deficit could drive the question design process. Same procedure can also ensure that the questions in the pool are reasonably evenly distributed among the section concepts (to avoid the situation where 80% of questions are testing 20% of concepts).

3.6 Matching presentations with examples and exercises

It is possible to check to what extent examples and exercises matches their place in the course and to what extent they cover the presented content. It 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) about 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. Small mismatch between presentations, examples, and concepts is not a problem, but bigger mismatch in either direction is a sign of poorly designed section and an indication that something has to be redesigned.

3.7 Checking course design 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 are really introduced; how the set of target concepts is supported by questions, examples, and exercises.

3.8 Presentation density and sectioning

While different concepts may require different amount of presentation, the overall 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 number of concepts presented in subsections of the same level significantly differs.

3.9 Controlling the difficulty of examples and exercises

Prerequisite indexing of exercises and examples specifies minimal requirements for the concept level that have to be met to make an example or an exercise ready to be taken. Its legal, however that some concepts have *higher* level of knowledge then it 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 reach this exercise when he or she has already seen several examples with this concept or even solved an exercise involving this concept. It makes this example easier for that student. Generally, we can estimate difficulty or learning item by measuring a difficulty between the target state of the goal concepts and the starting state. If all goal concepts or an exercise have been already used in earlier solved exercises, the exercise is quite simple. If none of them have even been used in examples, the exercise is very difficult. The difficulty of an exercise is not a constant – it depends on the place of the exercise in the course. It makes sense to control the difficulty of examples and exercises in the course to make sure that none 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 with difficulty closed to optimal. We can't use this finding directly since our courses are static – all students go the same way. But it is quite likely that different groups of users can handle different difficulties. It could be used for making better-targeted courses for special categories of users.

4 Implementation and first experience

The first version of the system was completed in 1999 and evaluated on one of CTE courses. With a help of the system we were able to find and fix a number of problems in the course. The system is written in Java and supports prerequisite checking, finding content "holes", consolidation of presentations, and question placement and repositioning. Currently the system is not completely interactive. The author has to specify the course structure along with concept tags in a separate file. The situation with question indexing is different - here concept tags are stored as a part of a

question. Checking scripts are simply called from a command line. An interactive (GUI-based) version of the system is being developed.

The system was used to check two real courses. While the system turned out to be very useful, we have encountered a problem. In addition to a revealing good number of real large and small hidden problems the system has also reported a number of problems that no real teacher would count as a problem. It turned out that the course consistency rules behind the system are too rigid. In real life teachers can perfectly tolerate a number of small inconsistencies in the course. Moreover, in some cases the course may be formally "inconsistent" with a purpose. A teacher may want to provoke student thinking by presenting an example that is based on a material that is not yet presented but could be understood by analogy with the learned material. Our quick answer to this problem was color coding the course problem report (Figure 1). In particular, the messages that always report a real problem in the course are colored red not to be missed. The messages that report a problem that often may be tolerable are colored green. We use three to four colors in our reports. A real solution to this problem would be a more precise set of checking rules that is adapted to the course "teaching approach" and, probably, a better indexing.

5 Prospects

We plan to continue the work on course maintenance system adding features and checking it with incrementally larger volumes of course material. We see a very important mission in this process. The outcome of this process is not only consistent courses of higher quality, but also a large volume of carefully indexed learning material. Thus we are decreasing bootstrapping cost of more flexible sequencing technologies. We hope that this process will eventually lead to the acceptance of more flexible approaches in large-scale Web-based education: first, to a DCG-like course customization and later to real course sequencing.

References

1. Ahanger, G. and Little, T. D. C.: Easy Ed: An integration of technologies for multimedia education. In: Lobodzinski, S. and Tomek, I. (eds.) Proc. of WebNet'97, World Conference of the WWW, Internet and Intranet, Toronto, Canada, AACE (1997) 15-20
2. Barr, A., Beard, M., and Atkinson, R. C.: The computer as tutorial laboratory: the Stanford BIP project. International Journal on the Man-Machine Studies **8**, 5 (1976) 567-596
3. Brusilovsky, P.: ILEARN: An intelligent system for teaching and learning about UNIX. In: Proc. of SUUG International Open Systems Conference, Moscow, Russia, ICSTI (1994) 35-41
4. Brusilovsky, P.: Adaptive and Intelligent Technologies for Web-based Education. Künstliche Intelligenz , 4 (1999) 19-25

5. Brusilovsky, P., Eklund, J., and Schwarz, E.**:** Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems. **30**, 1-7 (1998) 291-300
6. Brusilovsky, P., Schwarz, E., and Weber, G.: ELM-ART: An intelligent tutoring system on World Wide Web. In: Frasson, C., Gauthier, G. and Lesgold, A. (eds.) Intelligent Tutoring Systems. Lecture Notes in Computer Science, Vol. 1086. Springer Verlag, Berlin (1996) 261-269
7. Brusilovsky, P. L.: A framework for intelligent knowledge sequencing and task sequencing. In: Frasson, C., Gauthier, G. and McCalla, G. I. (eds.) Intelligent Tutoring Systems. Springer-Verlag, Berlin (1992) 499-506
8. Brusilovsky, V.: Task sequencing in an intelligent learning environment for calculus. In: Proc. of Seventh International PEG Conference, Edinburgh (1993) 57-62
9. Capell, P. and Dannenberg, R. B.: Instructional design and intelligent tutoring: Theory and the precision of design. Journal of Artificial Intelligence in Education **4**, 1 (1993) 95-121
10. Caumanns, J.: A bottom-up approach to multimedia teachware. In: Goettl, B. P., Halff, H. M., Redfield, C. L. and Shute, V. J. (eds.) Intelligent Tutoring Systems. Springer-Verlag, Berlin (1998) 116-125
11. De Bra, P. and Calvi, L.: 2L670: A flexible adaptive hypertext courseware system. In: Grønbæk, K., Mylonas, E. and Shipman III, F. M. (eds.) Proc. of Ninth ACM International Hypertext Conference (Hypertext'98), Pittsburgh, USA, ACM Press (1998) 283-284
12. Eliot, C., Neiman, D., and Lamar, M.: Medtec: A Web-based intelligent tutor for basic anatomy. In: Lobodzinski, S. and Tomek, I. (eds.) Proc. of WebNet'97, World Conference of the WWW, Internet and Intranet, Toronto, Canada, AACE (1997) 161-165
13. Khuwaja, R., Desmarais, M., and Cheng, R.: Intelligent Guide: Combining user knowledge assessment with pedagogical guidance. In: Frasson, C., Gauthier, G. and Lesgold, A. (eds.) Intelligent Tutoring Systems. Lecture Notes in Computer Science, Vol. 1086. Springer Verlag, Berlin (1996) 225-233
14. McArthur, D., Stasz, C., Hotta, J., Peter, O., and Burdorf, C.: Skill-oriented task sequencing in an intelligent tutor for basic algebra. Instructional Science **17**, 4 (1988) 281-307
15. Rios, A., Pérez de la Cruz, J. L., and Conejo, R.: SIETTE: Intelligent evaluation system using tests for TeleEducation. In: Proc. of Workshop "WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems, San Antonio, TX (1998), available online at http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/rios.html
16. Vassileva, J.: Dynamic Course Generation on the WWW. In: Boulay, B. d. and Mizoguchi, R. (eds.) Artificial Intelligence in Education: Knowledge and Media in Learning Systems. IOS, Amsterdam (1997) 498-505
17. WBT Systems: TopClass, Dublin, Ireland, WBT Systems (1999) available online at http://www.wbtsystems.com/
18. WebCT: World Wide Web Course Tools, Vancouver, Canada, WebCT Educational Technologies (1999) available online at http://www.webct.com