

Accessing Interactive Examples with Adaptive Navigation Support

Michael Yudelson University of Texas at Dallas mv011000@utdallas.edu
Peter Brusilovsky, Sergey Sosnovsky University of Pittsburgh {peterb, sas15}@pitt.edu

Abstract

This paper discusses a need to provide adaptive navigation support for students accessing large numbers of interactive examples in Web-enhanced education. We introduce the system NavEx that is able to provide adaptive annotation of programming examples without any need of manual indexing. NavEx uses innovative algorithms for prerequisite/outcome indexing of learning material and an efficient knowledge-based approach for adaptive annotation.

1. Web-based Examples in a Programming Course

It has often been claimed that humans use solutions to previous problems to solve new problems or planning tasks. Experienced teachers of programming courses present and explain one or more example programs during every lecture. In Web-based and Web-enhanced education the code of these examples is provided on the Web along with other kinds of learning material so that the students can study and run this code on their own. Our system WebEx [2] attempts to integrate these two approaches to example presentation. WebEx serves interactive explained examples via the Web. The idea of WebEx is simple: an author of an example can provide textual explanations for each line of the code. The students can browse these comments at their own pace and order by clicking on the commented lines.

The first version of WebEx has been implemented in 2001 [2]. Since that WebEx was used every semester in undergraduate programming courses at the University of Pittsburgh. Each semester WebEx offers an incrementally larger subset of examples from the classroom in the commented interactive format. We have run several questionnaire-based evaluations of WebEx over the last semesters. In their free-form answers to the first two questionnaires a few students expressed interests to see more commented programming examples than just 3-5 examples offered for each lecture. To evaluate the need for it, students were asked what kind of examples they want to be presented with WebEx. The answers ranged from none at all, to all examples from the lecture, to additional examples beyond those provided at the lecture. The results demonstrated that a solid fraction of students wanted to see more ex-

plained examples than just the examples from the lecture. Moreover, the proportion of students who want "more examples" was growing as long as we were making more and more lecture examples available each semester. For example, in the Fall 2002 semester, only 39% of respondents were interested in examples "beyond lectures" while 61% were interested to have lecture examples only. In the Spring 2003 semester, 50% voted for "more examples" and 50% wanted nothing but lecture examples (in both semesters no one selected "no examples" option).

To help the student in selecting the "right" example, we have decided to apply prerequisite-based annotation, a specific adaptive navigation support technology [1]. This technology was first explored in ISIS-Tutor system [5] and found very efficient. Since then, it was used in ELM-ART [11], InterBook [4], ACE [10], and other systems. With this technology, each learning object is described in terms of domain concepts. Some concepts serve as learning goals and others as prerequisites to the object. Tracking the student current level of knowledge, an adaptive system can dynamically classify examples as being relevant or not. The status of each object is shown by adaptive annotation using a traffic light metaphor.

This paper presents our system NavEx (Navigation to Examples) that was designed to explore the idea of providing adaptive navigation support for accessing programming examples. Further sections of the paper introduce the technologies developed to implement NavEx.

3. Indexing Examples in NavEx

For NavEx system we have developed the indexing component, which parses C code of each example and generates the list of relevant concepts. 51 concepts have been determined for the subset of C language learnt during the course. Each programming structure in an example can be indexed by one or more concepts.

The next stage is dividing concepts related to each example into prerequisite and outcome concepts. NavEx uses an original algorithm for automatic identification of outcome concepts. This algorithm adapts to an instructor-specific way of teaching the course. The source of knowledge for this algorithm is a sequence of example groups. Each group is formed by examples introduced in the same lecture. Groups are as lectures in the course.

The outcomes of this algorithm are a fully indexed set of lecture examples and a sequence of learning goals associated with the course lectures. This sequence represents a specific approach to teaching C programming employed by the instructor [3]. Once the course examples are indexed and the goal sequence is constructed, any programming example can be properly indexed by the algorithm and even associated with a specific lecture in the course. More exactly, an association with a specific lecture is the first step in this process. The example is associated with the last lecture that introduces example concepts (i.e., the lecture with the largest number that has at least one example concept in its pedagogical goal). After that, the example is indexed as belonging to this lecture. It is important to stress again that outcome identification is adapted to a specific way of teaching a course "mined" from the original sequence of examples. It has been known for a long time that different instructors presenting the same programming language may use very different orders of concept presentation [7]. Naturally, example sequencing in a course has to be adapted to the instructor's way of teaching.

4. Providing Adaptive Navigation Support to Examples

Adaptation of navigation in NavEx is done on the basis of the overlay user model [6]. The current status of each example is presented in the form of adaptive annotations. Already mastered examples are annotated with green check marks, available examples are annotated with green bullets, unavailable – with red bullets.

When user clicks on example link and reviews the example code the outcome concepts of the example change their state to "learned". The availability of new examples is determined by checking whether any of the previously unavailable examples now have all of their prerequisite concepts mastered. The knowledge-based adaptive annotation approach used in NavEx is a variation of a popular adaptive annotation approach introduced originally in ISIS-Tutor system. This approach is known to be very efficient [5].

5. Summary and the Future Work

We have presented the NavEx system. NavEx uses an efficient adaptive prerequisite-based annotation approach. However, unlike other systems that use this approach, NavEx does not require any manual indexing of educational objects. As a result, NavEx can be used by virtually any teacher of a programming course. NavEx allow teachers unfamiliar with adaptive hypermedia adding their examples, exchanging their examples, or using examples from a digital library.

Our current goal is to improve the quality of adaptive navigation support in NavEx. The current version helps to distinguish used, ready, and not-ready examples. Next version will recognize examples that are too simple for the student and also recommend examples that are relevant to the current learning goal. We also plan to connect example exploration with self-assessment of knowledge using our system QuizPACK [9].

6. References

- [1]. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction* 6, 2-3 (1996) 87-129.
- [2]. Brusilovsky, P.: WebEx: Learning from examples in a programming course. In: Fowler, W. and Hasebrook, J. (eds.) *Proc. of WebNet'2001, World Conference of the WWW and Internet, Orlando, FL, AACE (2001)* 124-129.
- [3]. Brusilovsky, P.: Developing Adaptive Educational Hypermedia Systems: From Design Models to Authoring Tools. In: Murray, T., Blessing, S. and Ainsworth, S. (eds.): *Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software*. Ablex, Norwood (2003) In Press.
- [4]. 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
- [5]. Brusilovsky, P. and Pesin, L.: Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor. *Journal of Computing and Information Technology* 6, 1 (1998) 27-38.
- [6]. Lutz, R.: Plan diagrams as the basis for understanding and debugging pascal programs. In: Eisenstadt, M., Keane, M. T. and Rajan, T. (eds.): *Novice programming environments. Explorations in Human-Computer Interaction and Artificial Intelligence*. Lawrence Erlbaum Associates, Hove (1992) 243-285.
- [7]. Moffatt, D. V. and Moffatt, P. B.: Eighteen pascal texts: An objective comparison. *ACM SIGCSE bulletin* 14, 2 (1982) 2-10.
- [8]. M. Polson, J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*, Lawrence Erlbaum Associates, 1988.
- [9]. Sosnovsky, S., Shcherbinina, O., and Brusilovsky, P.: Web-based parameterized questions as a tool for learning. In: Rossett, A. (ed.) *Proc. of World Conference on E-Learning, E-Learn 2003, Phoenix, AZ, USA, AACE (2003)* 309-316.
- [10]. Specht, M. and Oppermann, R.: ACE - Adaptive Courseware Environment. *The New Review of Hypermedia and Multimedia* 4 (1998) 141-161.
- [11]. Weber, G. and Bögelsack, A.: Representation of programming episodes in the ELM model. In: Wender, K. F., Schmalhofer, F. and Böcker, H.-D. (eds.): *Cognition and Computer Programming*. Ablex, Norwood, NJ (1995) 1-26.