NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples

Michael YUDELSON, Peter BRUSILOVSKY
School of Information Sciences, University of Pittsburgh
135 N. Bellefield Ave., Pittsburgh, PA 15260 USA
{peterb, mvy3}@pitt.edu

Abstract. This paper presents NavEx, an adaptive environment for accessing interactive programming examples. NavEx implements a specific kind of adaptive navigation support known as adaptive annotation. The classroom study of NavEx confirmed that adaptive navigation support can visibly increase student motivation to work with non-mandatory educational content. NavEx boosted the overall amount of work and the average length of a session. In addition, various features of NavEx were highly regarded by the students.

Keywords: adaptive environments (web-based and other), motivation and engagement in learning, web-based learning platforms.

1. Introduction

Program examples in the form of small but complete programs play an important role in teaching programming. Program examples help students to understand syntax, semantics and the pragmatics of programming languages, and provide useful problem-solving cases. Experienced teachers of programming-related courses prepare several program examples for every lecture and spend a reasonable fraction of lecture time analyzing these examples. To let the students further explore the examples and use them as models for solving assigned problems, teachers often include the code of the examples in their handouts and even make the code accessible online. Unfortunately, these study tools are not a substitute for an interactive example presentation during the lecture. While the code of the example is still there, the explanations are not. For the students who failed to understand the example in class or who missed the class, the power of the example is lost.

Our system WebEx (Web Examples) developed in 2001 [1] attempted to enhance the value of online program examples by providing *explained examples*. The authoring component of WebEx allowed a teacher to prepare an explained example by adding a written comment for every line of it. The delivery component (see right frame on Figure 1) allowed a student to explore explained examples interactively. Lines with available comments were indicated by green bullets. A click on a bullet opened a comment for the line. This design preserved the structure of an example while allowing the students to selectively open comments for the lines that were not understood. Over the last 4 years we have developed a large set of explained examples for WebEx, used it for several semesters in two different programming-related courses, and run several classroom studies.

In the course of classroom studies of WebEx, the system proved itself as an important course tool. Students rated the system highly, with its ability to support

interactive exploration of examples. Many students actively used the system through the course, exploring many examples from different lectures. Yet, a sizeable fraction of students used the system on only a few occasions. Knowing this pattern from our past work on adaptive hypermedia [2], we hypothesized that the students might need some kind of adaptive navigation support that would suggest the most relevant example to explore at any given time. Indeed, with dozens of interactive examples available at the same time, it's not easy to select one to explore. Moreover, WebEx examples were scattered over the course portal with several examples assigned to every lecture. While this organization supported example exploration after a lecture, the abundance of examples made the search for the right example harder.

The experience of ELM-ART [3] demonstrated that the proper adaptive navigation support can significantly increase the amount of student work with a non-mandatory educational content. To gain additional evidence in favor of adaptive navigation support in our context, we solicited student feedback about the need of adaptation in the Spring 2003 study of WebEx. One of the questions in our WebEx questionnaire explained possible adaptive navigation support functionality and asked the students whether this functionality is useful. Almost 70% of respondents (out of 28) rated adaptive navigation support as at least a useful feature (almost 30% rated it as very useful).

This data encouraged us to enhance the original WebEx system with adaptive navigation support. The work on NavEx, an adaptive version of WebEx started in the Fall of 2003 and an early prototype [4] was pilot-tested in Spring 2004. This paper describes the final version of NavEx, which was completed and evaluated in a classroom study in the Fall 2004 semester. The following sections present the interface of NavEx, explain how its adaptive functionality is implemented, and report the results of our classroom study. In brief, the study confirmed positive student attitude toward our adaptive navigation support and demonstrated that one of our specific adaptive navigation support approaches caused impressive growth in system usage.

2. NavEx: The Interface

The goal of our NavEx system (Navigation to Examples) is to provide adaptive navigation support in order to access a relatively large set (over 60) of interactive programming examples. Capitalizing on our positive experience with ISIS-Tutor [5], ELM-ART [3] and InterBook [2] we decided to apply a specific kind of adaptive navigation support known as adaptive annotation. With adaptive annotation, a system provides adaptive visual cues for every link to educational content. These visual cues (for example, a special icon or a special anchor font color) provide additional information about the content behind the links helping a student to choose most relevant proper link to follow. One important kind of adaptive annotation pioneered in ISIS-Tutor is zone-based annotation, which divides all educational content into three "zones": 1) sufficiently known, 2) new and ready for exploration, and 3) new, but not-yet-ready. This kind of annotation was later applied in ELM-ART, InterBook, AHA! [6], KBS-HyperBook [7], and many other systems. Another kind of adaptive annotation pioneered in InterBook [3] is progress-based annotation, which shows current progress achieved while working with an educational object. This kind of annotation is currently less popular and is only used in a few systems such as INSPIRE [8].

While the prototype version of NavEx [4] used only zone-based annotation, the current version attempts to combine zone-based and performance-based annotation in a single adaptive icon. The goal of adaptive annotation in NavEx is to provide three types of information to students:

- Categorize examples as being either: ones the student is *ready for* or *not yet ready* to explore;

- Delineate is the student's *progress* within the examples (showing number of explored annotated code lines);
- Emphasize *the most relevant* examples for the student given her past interaction with NavEx or WebEx (all of interaction with WebEx is taken into consideration by NavEx).

The NavEx interface is shown on **Figure 1**. The left side displays a list of annotated links to all the code examples available for a student in the current course. The right side displays the name of the current example, the menu buttons (such as 'reload', 'hide left frame', and 'help'), and the annotated code example.

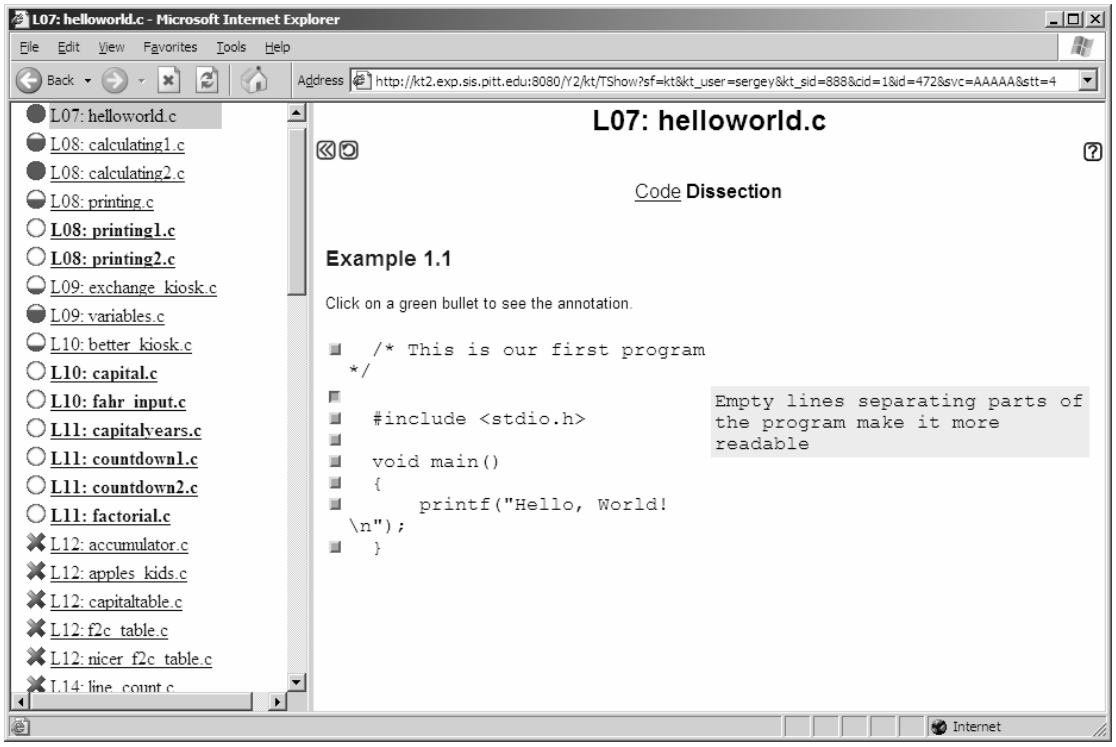

Figure 1. NavEx interface

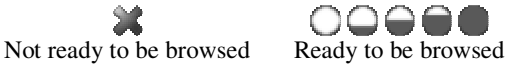

Figure 2. Annotation of the examples

Students click on links in the left frame to select an example and browse annotated code, by clicking again on colored bullets, in order to obtain teacher's comments. Each link to an example in the left frame is supplied with an icon that conveys information about (1) 'readiness' of the student to browse the example, and (2) the student's progress within the example. If the student is 'not ready' to browse the example then a red X bullet is displayed (**Figure 2**). If the student is 'ready' to browse the example then a green round bullet is shown. Depending on the student's progress, the green bullet will be empty, partially or wholly filled. There are 5 discrete progress measures from 0% to 100%, with 25% increments (**Figure 2**). An empty green bullet denotes examples that are available, yet not browsed by the student. The relevance of the example is marked by the font style. If the example is relevant its link is displayed in bold font, otherwise it is in regular font (**Figure**

1). The fact that the example is 'not ready' or 'not recommended' doesn't prevent the user from actually browsing it. All of the annotated examples are available for exploration and it is up to a student as to whether to follow the suggestions expressed by annotations or not.

3. NavEx: The Implementation and the Internal Mechanisms

The annotation of examples is compiled, based on the domain model concepts. Each of the examples is indexed with such concepts before it is added to the system. The indexing goes through two stages. First, concepts are extracted from each of the examples by a fully-automatic operation-level parser. Second, for each of the examples, the set of concepts is split into prerequisite concepts and outcome concepts. The splitting algorithm, besides example-concept pairs, requires examples to be grouped by lecture. Indexing algorithms are discussed in more detail in [9]. Supplying each example with two sets of concept - prerequisites and outcomes – plays a two-fold role. First, the concept separation helps to define the learning goals (focus) of the examples in terms of outcomes. Second, concept separation is used for partial ordering of the examples. Thus, an example that has a certain concept as a prerequisite will be placed after an example that has the said concept as outcome.

Once the example is in the system, its annotation for the current user is determined by counting whether or not the current user has mastered the prerequisite concepts. If all of the prerequisite concepts are mastered (or the example simply has no prerequisite concepts) – the example is considered 'ready to be browsed.' If the prerequisite concepts are not mastered – the example is marked as 'not ready to be browsed'. The progress of the student within the example is measured by counting the number of clicks on annotated lines of code example code the user has done with the example.

The relevance of the examples is calculated based on the 'threshold' parameter. The 'threshold' (calculated for each of the examples individually) is the amount of clicks that has to be done by student for the system to conclude that s/he 'knows' the example and declare all of concepts corresponding to example to be mastered. The threshold amount of clicks is calculated as:

$$threshold = 0.8 * [\ (all\_concepts - mastered\_concepts) / all\_concepts\ ] * all\_clicks$$

Namely, the total number of clicks possible (for current example) is multiplied by user has to click 80% of the ratio of currently not-mastered concepts (to mastered concepts out of the current example) to all concepts (of the current example). This gives the number of clicks 'left' for user to do and he has to make 80% of those to 'master' the example. Only clicks on distinct code lines are counted.total clicks possible. E.g. if there are 10 clicks possible on the lines of the code example and there are 10 concepts assigned to the example: 5 prerequisite (all mastered) and 5 outcomes (none mastered), then the user has to make 0.8 * (5/10) * 10 = 4 clicks to 'master' the example. As soon as some concepts are declared mastered the 'readiness' of all other examples is recalculated and the mastery of the concepts is propagated further.

The threshold is only used to determine the minimal amount of work the student has to do with the individual example to learn the underlying concepts. The annotation of the examples reflects the absolute amount of student's work and is not related to the threshold. Since all of the examples share the pool of concepts, it might turn out that at some point there will be one or more examples whose concepts are mastered, yet the student has never browsed those. As mentioned in a previous section, students can browse examples that are annotated as 'not ready to be browsed'. In extreme cases, the student can browse an example, which contains only concepts that are not yet mastered. To master those concepts

while browsing such an example, the student will have to do an extensive amount of clicks, as determined by the threshold.

The NavEx interface is implemented as a server-side solution written in Java. All knowledge and data are stored in a relational database. NavEx is considered to be a value-added service of the KnowledgeTree architecture [10], and uses several protocols, including student modeling and transparent authentication. As a typical value-added service, NavEx resides between E-Learning portals and reusable content objects, providing additional value for teachers and students who use this content through the portal. Unlike other kinds of value-added services, such as annotation services, the value added by NavEx is the ability to adapt to the course goals and student knowledge. With NavEx, teachers can bypass the time-consuming process of selecting examples for each course lecture that meet goal and prerequisite restrictions. Students receive adaptive guidance in selecting examples that are most relevant to their learning goals and knowledge.

4. A Classroom Study of NavEx

A classroom study of NavEx was performed in the context of an undergraduate programming course in the Fall 2004 semester in the School of Information Sciences at the University of Pittsburgh. NavEx was made available to all students taking this course in the second half of the semester, after the midterm exam. There were totally 15 students working with the system. Before the introduction of NavEx the students were able to explore code examples with the original WebEx (i.e., without adaptive guidance) directly through the Knowledge Tree portal. After the introduction, they were able to use both methods of access – with adaptive navigation support through NavEx and without it through the portal and WebEx. User activity collection procedures does not depend on the way students access code examples. Student work with both WebEx and NavEx was equally considered for user modeling.

4.1 Log Analysis

Our main source of data for the study was the user activity log. The log recorded every user click (i.e., every example and code line accessed). Note that the log data gave clear evidence as to whether a student accessed a specific example through NavEx or through WebEx. Since students used WebEx and NavEx in parallel (the use of NavEx was not enforced), a natural way to evaluate the influence of adaptation was to compare the usage profiles of WebEx and NavEx. Analysis of the data showed that NavEx, though introduced late in the course, was considered as a strong alternative to WebEx. After the introduction of NavEx, 56% of example browsing activity was generated by NavEx users. Only 30% of the students didn't use NavEx at all.

Since different students used different "mixtures" of WebEx and NavEx through the course, we decided to assess the added value of the adaptive navigation support by comparing these two systems on a session-by-session basis. A session is counted as a sequence of examples browsed by the student without any sizeable break. The result of this comparison demonstrated clearly the value of adaptive navigation support in increasing the amount of student work with examples.

First, the average session of non-NavEx users was 9.4±0.97 clicks, while NavEx users made an average of 29.6±4.65 clicks per session. That means that navigation support provided by NavEx encouraged students to click on 3.14 times more annotated code lines. Second, the average number of examples browsed per session of non-NavEx users was 1.78±0.15, while NavEx users browsed 2.95±0.46 examples per session. Thus NavEx motivates students to see an average of 1.66 examples more per session. And thirdly, the

average length of the non-NavEx user session is 225±33 seconds, while NavEx users have average session length of 885±266 seconds. Hence NavEx keeps students focused on examples 3.9 times longer.

Further evidence can be derived by comparing the example browsing statistics of Fall 2004 semester, when students could use adaptive guidance and Spring 2004 when they could not. Examples set in the Spring 2004 semester had only minor differences from the set of examples available in the Fall 2004 so we can assume that the students had the same external (i.e., tool-independent) motivation to use the tool. The only significant difference was that in the Fall 2004 semester students were able to use NavEx.

The comparison of student activity data of the two semesters demonstrated that the introduction of NavEx boosted the motivation of the students to work more with annotated code examples. The number of code lines accessed per session increased by about 11% from 14.22 in the Spring 2004 semester to 15.8 in the Fall 2004 semester (if we consider only NavEx users the number of clicks per session almost doubled). The average number of line accesses by students over a semester grew by 35% from 323.3 lines in the Spring 2004 semester to 435.9 in the Fall 2004 semester.

Thus, adaptive navigation support succeeded as a tool that encourages the students to work more with examples. It appears that the students were simply more motivated to work with examples when adaptive navigation support was provided. We think that such increase of students' motivations can be attributed to the following reasons. First, navigation support allows students to see 'the big picture' – visualize their current progress with all of their examples and estimate whether the progress they made is enough to move further. Second, since students had all the examples grouped together, they were able to switch from one example to another in fewer clicks and were interested in exploring more examples.

4.2 Subjective Data Analysis

Our secondary source of evaluation data was a non-mandatory questionnaire administered at the end of the term that solicited students' opinions about key features of the system. Out of 15 students in the class, 10 completed the questionnaire.

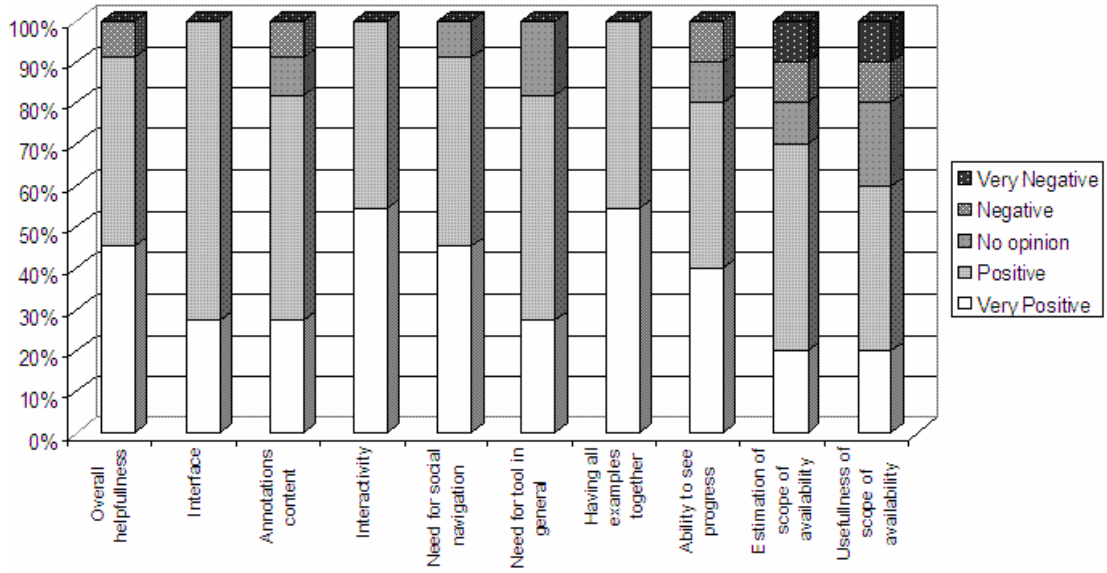

Figure 3. Subjective student evaluation of different features of NavEx

Some of the data obtained from processing the answers is shown in **Figure 3**. As it can be seen, 90% of students considered annotated examples with or without adaptive guidance helpful. 80% percent of students feel positive or strongly positive about the need for such a tool in general. All of the respondents positive or strongly positive evaluated the convenience to have all of the annotated code examples together. 100% of students positively or strongly positively evaluated the interface and the interactive nature of examples.

Two principal features of NavEx: progress indicator and the scope of availability ('readiness') were evaluated positively or strongly positively by a solid fraction of the students (80% for progress indicator and 60-70% for the scope of 'readiness'). The slight downfall of positive response about the scope of 'readiness' of examples' annotation we account to the fact that students started with NavEx in the middle of semester. At the time of their first logon, all of the examples were 'not ready to be browsed', yet at that time students were already familiar with almost half of them and had literally to 'get through' the red X's. Nevertheless, they did appreciate the scope of 'readiness' on the whole.

Students also had a chance to express their suggestions about the future use and development of the system. The idea of students being able to create their own dissections or add their own annotations to the code lines was supported by 70% of respondents (when such activity is an extra credit assignment), and strongly supported by 10% (when such activity is a regular assignment). 90% students expressed strong and very strong support for adding a social navigation feature. A substantial amount of students have also expressed certainty that NavEx should remain as one of the class tools available for students.

5. Summary and Future Work

This paper presented the NavEx system, which provides adaptive navigation support for students accessing interactive program examples. We implemented adaptive navigation support to encourage the students to work more with program examples. Our classroom study confirmed that adaptive navigation support can visibly increase student motivation to work with non-mandatory educational content. NavEx boosted the overall amount of work and the average length of a session. In addition, various features of NavEx were highly regarded by the students. Among two kinds of adaptive navigation support, performance-based annotation was appreciated more than zone-based annotation. However, it may have been influenced by the late introduction of the system.

We plan to perform further studies with NavEx to achieve a better understanding of the value of adaptive navigation support. In addition, we plan to extend the scope of adaptive annotation by providing an annotation of every commented line in an example – not only an example as a whole. To make it possible, we will apply social navigation techniques that we are currently exploring in the course of another project.

References

[1]. Brusilovsky, P. (2001) WebEx: Learning from examples in a programming course. In: W. Fowler and J. Hasebrook (eds.) Proceedings of WebNet'2001, World Conference of the WWW and Internet, Orlando, FL, October 23-27, 2001, AACE, pp. 124-129.

[2]. Brusilovsky, P., Eklund, J. (1998) A study of user-model based link annotation in educational hypermedia. In P. Carlson (ed.) Journal of Universal Computer Science 4 (4), Special Issue on Assessment Issues for Educational Software, 429-448, also available at http://www.iicm.edu/jucs_4_4/a_study_of_user.

[3]. Weber, G., Brusilovsky, P. (2001) ELM-ART: An adaptive versatile system for Web-based instruction. In P. Brusilovsky and C. Peylo (eds.), International Journal of Artificial Intelligence in Education 12 (4), Special Issue on Adaptive and Intelligent Web-based Educational Systems, 351-384, also available at http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/weber.html.

[4]. Brusilovsky, P., Yudelson, M., and Sosnovsky, S. (2004) An adaptive E-learning service for accessing Interactive examples. In: J. Nall and R. Robson (eds.) Proceedings of World Conference on E-Learning, E-Learn 2004, Washington, DC, USA, November 1-5, 2004, AACE, pp. 2556-2561.
[5]. Brusilovsky, P. and Pesin, L. (1998) Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor. Journal of Computing and Information Technology 6 (1), 27-38.
[6]. De Bra, P., Calvi, L. (1998) AHA! An open Adaptive Hypermedia Architecture. In P. Brusilovsky and M. Milosavljevic (eds.), The New Review of Hypermedia and Multimedia 4, Special Issue on Adaptivity and user modeling in hypermedia systems, 115-139.
[7]. Henze, N., Nejdl, W. (2001) Adaptation in open corpus hypermedia. In P. Brusilovsky and C. Peylo (eds.), International Journal of Artificial Intelligence in Education 12 (4), Special Issue on Special Issue on Adaptive and Intelligent Web-based Educational Systems, 325-350, also available at http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/henze.html.
[8]. Papanikolaou, K. A., Grigoriadou, M., Kornilakis, H., and Magoulas, G. D. (2003) Personalising the interaction in a Web-based Educational Hypermedia System: the case of INSPIRE. User Modeling and User Adapted Interaction 13 (3), 213-267
[9]. Brusilovsky, P., Sosnovsky, S., Yudelson, M., Chavan, G. (2005) Interactive Authoring Support for Adaptive Educational Systems. In: Proceedings of 12th International Conference on Artificial Intelligence in Education (AIED'2005), Amsterdam, the Netherlands, this volume.
[10]. Brusilovsky, P. (2004) KnowledgeTree: A distributed architecture for adaptive e-learning. In: Proceedings of The Thirteenth International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), New York, NY, 17-22 May, 2004, ACM Press, pp. 104-113.