From WebEx to NavEx: Interactive Access to Annotated Program Examples

Peter Brusilovsky and Michael Yudelson

Abstract— **This paper reviews our work on providing students interactive access to annotated program examples. We review our experience with WebEx, the system that allows students to explore examples line by line. After that we present NavEx, an adaptive environment for accessing interactive programming examples. NavEx enhances WebEx with a specific kind of adaptive navigation support known as adaptive annotation. The classroom study of NavEx discovered that adaptive navigation support can visibly increase student motivation to work with non-mandatory educational content. NavEx boosted the overall amount of work done and the average length of a session. In addition, various features of NavEx were highly regarded by the students.**

Index Terms— **Adaptive systems, Computer science education, Educational technology, Hypertext systems, Programming, User modeling**

I. Introduction

PROBLEM-solving examples play an important role in teaching many engineering disciplines. In the area of teaching programming, program examples in the form of small meaningful programs help students to understand syntax, semantics and the pragmatics of programming languages, as well as to 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 these examples and use them as models for solving assigned problems, teachers often include the code from these examples in their handouts and may even make it accessible online. Unfortunately, these study tools are not a substitute for an interactive example presented during the lecture. While the example code 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.

The project presented in this paper attempted to solve this problem by offering the students an opportunity to explore programming examples as well as their explanations by using Web-based interactive tools outside of class. Over the course of the project we developed and evaluated two systems for Web-based access to examples – WebEx and NavEx. WebEx provided basic access to explained examples, while NavEx extended the power of WebEx by providing personalized guidance. This paper presents an account of our project. Sections II and III introduce WebEx and NavEx and review the students subjective feedback about the systems. Section IV compares these systems and examines the nature of the increased impact of NavEx. At the end we summarize the results and discuss our future research plans.

Manuscript received November 11, 2007. This work was supported in part by the University of Pittsburgh Innovation in Education Grant and by National Science Foundation under Grants OCI-0426021 and IIS-0447083.

P. Brusilovsky is with the School of Information Sciences, University of Pittsburgh, PA 15260 USA (phone: 412-624-9404; fax: 412-624-2788; e-mail: peterb @ pitt.edi).

M. Yudelson is with the School of Information Sciences, University of Pittsburgh, PA 15260 USA (e-mail: mvy3 @ pitt.edu).

II. WebEx: Exploring Annotated Program Examples

A. The Motivation

The goal of WebEx, a Web-based tool for exploring programming examples, was to turn explained examples into first class educational material that the students can explore anytime, anywhere and at their own pace. To achieve this goal, WebEx replaced the bare code of programming examples offered on the course Web site with *interactive explained examples*. This idea of explained examples was motivated by an approach to example explanations used in several programming textbooks and sometimes referred to as "dissections" [1]. In this approach, an author of an example supplies textual explanations for each important line in the example program. The explanations serve at least two different purposes. First, they explain the meaning of each program line and its role of in the overall solution of a programming problem. Second, the comments on a particular way of using language constructs in every line of code thus bridge the gap between student general knowledge about programming language constructs and the practical skills of their use for solving programming problems.

In a typical programming textbook a dissected example is provided in a format where each line of code is followed by explanations (which can vary from a line or two to several paragraphs of text). This format has a clear problem: even in textbooks that use some special font and color for the lines of code, the code is hard to comprehend since the lines of code are spread over the explanations. The explanations are not easy to comprehend either. Usually, a student has a problem with just a few lines of code in a program and need explanations for just these lines. Presenting all explanations at once distracts the student from concentrating on the most needed explanations. Finally, reading through a large "dissection" is a rather passive kind of learning.

B. The WebEx Interface

WebEx, a Web-based tool for interactive exploration of programming examples, was designed to overcome the problems listed above. A program example in WebEx appears to be just the same as it looked in a program editor (Figure 1). The only visible difference is the presence of green or white navigation bullets to the left of each line. A green bullet indicates the availability of explanations for this line of code. A white bullet marks that there are no explanations for that line. Clicking on a green bullet opens an explanations note for the selected line. In the spirit of good hypertext, the WebEx interface lets the user use his or her preferred browsing strategy. Some users may choose to browse the example line by line. Other students may concentrate on the most hard-to-understand lines and selectively read explanations for these lines. When exploring an example, such students can go straight to a troublesome line while ignoring other unnecessary explanations.

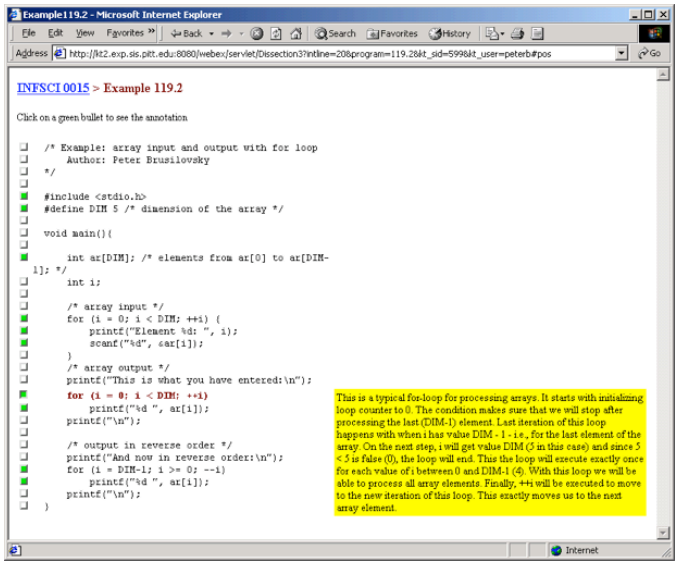

Fig. 1. In WebEx example explanations are shown one at a time next to the line being explained (the bullet and the font for this line is changed to provide the context)

WebEx approach offers several benefits over the traditional book format. First, the code of the example is shown as an easy-to-grasp single chunk, instead of being distributed among the comments. Second, explanations are shown one by one, helping the student to concentrate on one thing. Third, instead of being a passive reading activity, student work becomes an interactive exploration with every example. As an extra benefit, every action of the student in this environment can be recorded, thus providing a teacher with the opportunity to monitor student activity, which gives educational researchers a powerful tool to explore student work-with-examples in the programming domain.

C. WebEx Implementation

Over the course of the project we have developed several versions of WebEx using different technologies. The first version was implemented using Microsoft Access and Microsoft Active Server Pages. It featured standalone authoring tools, also based on Access. We used this version to check the feasibility of our approach in a formative classroom study. This version was presented at the WebNet'01 conference [2] and received enthusiastic feedback. The second and third versions were developed completely in Java, using Java Servlets for the student and author interfaces, while a MySQL database stored the examples. Altogether, these components formed a WebEx server.

All versions provided transparent Web-based access to examples. Every annotated example stored on a WebEx server was accessible by a unique URL. This provided for flexible use of the examples. A teacher may decide to place links to examples directly on a course Web page, send them by E-mail, or add them to a Course Management System (CMS) such as Blackboard. The system also supports the individual logging of user actions, but to switch on this logging, an example must be called with logging parameters, such as user name, user group, and user modeling server. Unfortunately, commercial CMS such as Blackboard were not able at that time to pass user parameters to interactive learning resources. To support the use of WebEx with full logging (which was critical for our user studies) we implemented a learning portal [3] and a communication architecture [4] called KnowledgeTree. With KnowledgeTree, a teacher is able to structure the course as a sequence of lectures (topics). For every lecture he or she can specify the objectives and add links to relevant learning activities. When a student selects such a link in KnowledgeTree, the portal requests the selected object from the corresponding content server (for example, WebEx server) and passes on the student parameters that enable the server to trace the student's work. The server immediately displays the requested example in a separate window, as shown on Figure 1.

D. Classroom Studies of WebEx

An educational system such as WebEx is typically evaluated from three different prospects: student performance, system usage, and subjective feedback. However, our main goal was to evaluate the system in a real full-semester classroom study. In this context, the classic performance evaluation approach (form two groups of students with and without access to WebEx and evaluate increase in their knowledge) would have found it difficult to provide reliable data: In the classroom, WebEx provides a good chance for the student to increase their knowledge, but does not guarantee it. In addition, we can't control the amount of work done with WebEx and other sources of learning the students may use in the context of a regular course. As a result, it is hard, in the context of the classroom, to expect a reliable correlation between the presence of WebEx and knowledge increase.

In contrast, the remaining two evaluation prospects were quite appropriate to use in the classroom. The goal of WebEx has been to encourage students to explore examples and to help them selectively access annotations provided by the

teacher. The usage analysis, discovered by examining the log of student actions, can show how much and how frequently the students use the system. In addition, a comparison of system usage by different categories of students (i.e., gender, grade) or with different versions of the system, allows us to discover which student category appreciates the system most and which version is more attractive. A subjective evaluation can measure student opinion about the system as a whole as well as to distinguish between its different parts. If opinion is positive, we may derive from this that the students are benefiting from the system. This is not ultimate proof, but it is good evidence of success. In addition, subjective questionnaires provide a great amount of data for improving the system. Naturally, we used a combination of subjective feedback and usage analysis to evaluate WebEx. In this paper, the results of the subjective evaluation of WebEx are presented below, while the results of usage analysis are presented in section 4 (as mentioned above, usage analysis is most interesting in the context of comparing several versions of the system).

We ran several classroom studies of WebEx (Spring 2002, Fall 2002, and Spring 2003) in the context of two different undergraduate programming courses taught at the University of Pittsburgh. The first study used the first version of WebEx and the three other studies used the second full-featured version of the system. All studies had about the same format. The students were encouraged to work with WebEx examples for a few days and then answer a brief questionnaire. We considered the first study to be a formative one. The questionnaire was quite short and administered anonymously. The second and third studies were mostly comprehensive; we considered them to be a combination of formative and summative evaluations. The original set of questions was extended, the students had more time to work with the system, and we preserved the student identities, in order to be able to analyze the profile of student answers in conjunction with their performance and demographic data.

All subjects in our studies were students of undergraduate courses on Introductory Programming. The use of the system was voluntary and not rewarded by grades directly, however students who used the system were able to receive 3 extra credit points for filling in the questionnaire. In total, 18 students filled in the questionnaire in Fall 2002 and 28 in Spring 2003.

The remaining part of this section provides the analysis of four multiple-choice questions that were the most relevant to the focus of this paper, including student choices and their free-form feedback, which was also solicited by the questionnaire. The questions and the answer options are listed in Table 1. The questions were designed in a Likert 4-point style where answer 4 always corresponded to a very positive opinion, answer 3 to a positive, answer 2 to neutral, and answer 1 to negative. Note that instead of the Likert homogeneous scale, we choose to provide a separate set of four answers for each question, which we considered as more meaningful, when collecting student feedback.

Figure 2 provides an overview of the student answers for the four selected multiple-choice questions. It is easy to notice that the overall student opinion of the system was very positive. For every evaluated aspect, more than 70% of the students have chosen either very high or high options. No negative options were selected. We consider this to be very strong evidence that the system was successful. Note that we conducted a similar study in the same classroom evaluating another course support tool, Knowledge Sea [5]. We considered the results of the Knowledge Sea study to show reasonable success as well, however, the student feedback about Knowledge Sea was quite below their feedback about WebEx, with the percentage of positive answers just a bit over 60% and the percentage of strongly positive answers under 10%. WebEx was clearly a champion with our students.

Table 1: Four questions relevant to our discussion with their answers. Answer option 1 corresponds to a negative opinion, answer option 2 to a neutral, answer option 3 to a positive, and answer option 4 to a very positive

	Question
1. Value of Examples	I think that annotated examples: 4. can significantly improve my understanding 3. can help me in understanding 2. can sometimes be of help 1. can sometimes be of help
2. Interface	Considering the interface for the annotated examples I think that it: 4. is very good 3. is good 2. have some problems 1. have some major problems
3. Content	The content of annotations in the examples was: 4. very good and helpful 3. quite helpful overall 2. sometimes helpful, but useless most of the time 1. not helpful
4. Value of Interactivity	The interactive functionality of the dissections (an ability to click on a selected line and to see the attached comment) was: 4. very useful 3. useful 2. useful only in a few cases 1. useless (no value over dissections in a textbook)

Note that Help in Understanding was clearly the highest rated feature of the system. Almost 1/3 of the students selected a highly positive answer, stating that the system was able to significantly improve their understanding. Speaking about their understanding, some students commented: "I was pleased with this tool since it gave me a chance to view material I never had a chance to understand;" "I think that the dissections are almost necessary and very helpful as they can explain an example or problem and make you really understand the point of each line or part of the program." They were also clearly able to appreciate the difference between static examples and annotated interactive WebEx format: "Very good tool, helps

grasp reasoning that sample programs just don't do." Finally, the students also commented on the value of the system in re-capturing explanations missed in class: "this was a very good idea to help catch up on a missed class or a topic that was not very clear in class!"

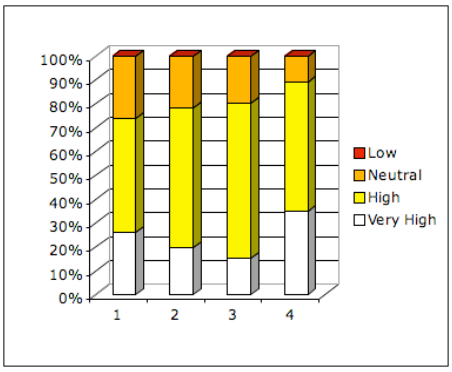

Fig. 2. The percentage of different answers for the questions from Table 1.

III. NavEx: Personalized Access to Examples

A. The Motivation

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 [6], we hypothesized that the students might need some personalized navigation support to guide them to the most relevant examples 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.

Our experience with ELM-ART [7] demonstrated that the proper adaptive navigation support can significantly increase the amount of student work with non-mandatory educational content. To gain additional evidence in favor of adaptive navigation support in our context, we solicited student feedback about the need for adaptation, in the Spring 2003 study of WebEx. One of the questions in our WebEx questionnaire explained the potential of adaptive navigation support functionality and asked the students whether this functionality would be useful. Almost 70% of the respondents (out of 28) rated adaptive navigation support as at least a useful feature and almost 30% among them rated it as very useful.

This data encouraged us to enhance the original WebEx system with adaptive navigation support. The work on NavEx (Navigation to Examples), an adaptive version of WebEx started in the Fall of 2003. The *pilot* version [8] was completed and evaluated in Spring 2004. The second, more elaborated *production* version [9] 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 the classroom studies.

B. The NavEx Interface

As mentioned above, the goal of NavEx was to provide adaptive navigation support for a relatively large set (over 60) of interactive programming examples. Capitalizing on our positive experience with ISIS-Tutor [10], ELM-ART [8] and InterBook [11] 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

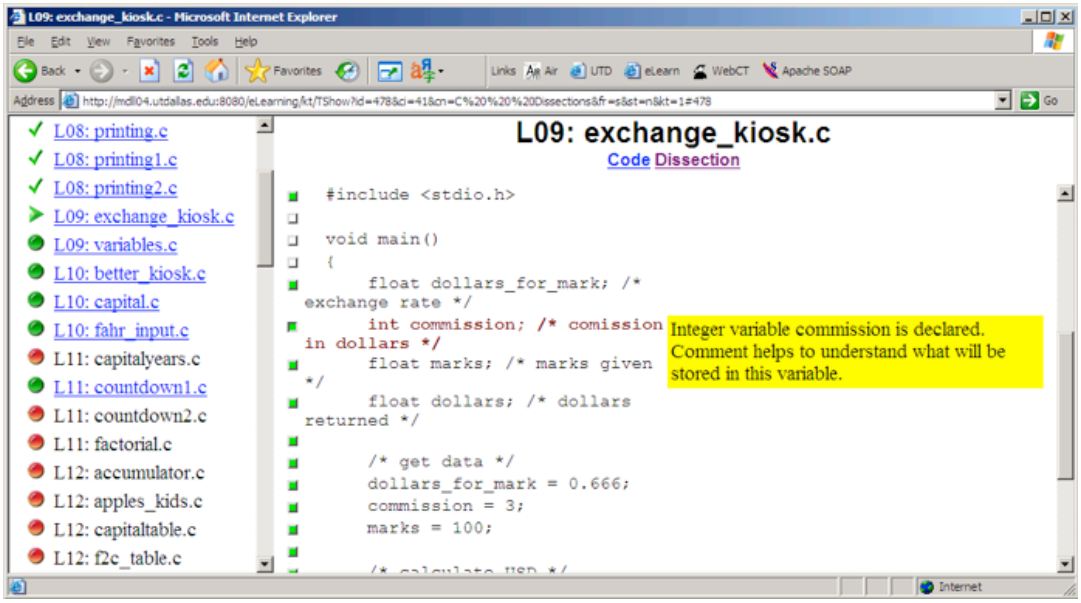

Fig. 3. The interface of the pilot version of NavEx

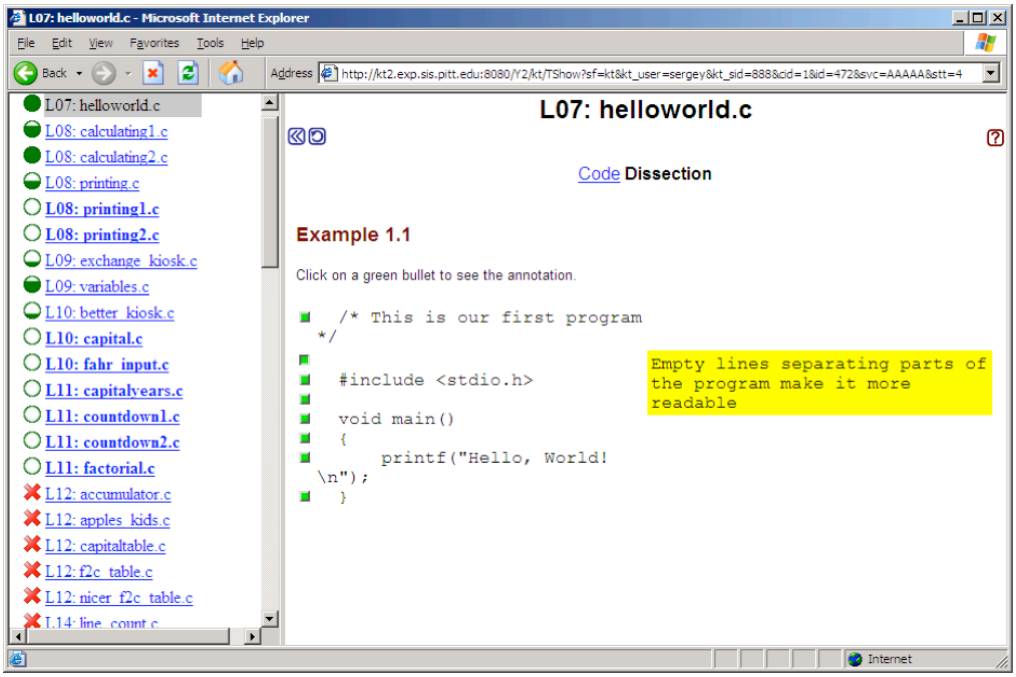

Fig. 4. The interface of the production version of NavEx

example, a special icon or a special font color for a linked anchor) provide additional information about the content behind the links, thus helping a student to choose the most relevant 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 [7], InterBook [11], KBS-HyperBook [12], and many other systems. Another kind of adaptive annotation pioneered in InterBook [11] 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 [13].

NavEx went through two design stages: pilot and production. The pilot version of NavEx [8] used only zone-based annotation. The second, production version [9], attempted to combine zone-based and performance-based annotation in a single adaptive icon.

The interface of the pilot version is shown in Figure 3. 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 and the annotated code example. Students click on links in the left frame to select an example. Once an example is selected, they click on colored bullets in the right frame next to example lines to selectively explore the teacher's comments.

Navigation support is provided in the left frame, in the form of adaptive icons. Check mark annotations denote "sufficiently known" examples, green bullet annotations – examples user is advised to work on, and red bullets discourage user from working with these annotated examples. The fact that the example is '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.

In the production version of the NavEx interface (Figure 4) we changed the annotation schema. The green bullet, which denoted "ready to be learned" examples was replaced with a fillable circle. Depending on the student's progress, the circle is empty, partially or wholly filled. There are 5 discrete progress measures from 0% to 100%, with 25% increments (Figure 5). 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 - in regular font.

The red bullet was changed to a more explicit red X mark. In addition menu buttons were added to the top of the interface window (such as 'reload', 'hide left frame', and 'help').

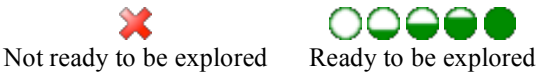

Fig. 5. Annotation of the examples

C. *NavEx Implementation*

NavEx is implemented as a value-added service. It aggregates WebEx examples and serves as a single point to access them. The interaction of NavEx and WebEx is shown in Figure 6. The main content frame of NavEx presents the original WebEx interface. The adaptive guidance component of NavEx takes care of the left navigation frame and the top menu frame.

The guidance component employs a *concept-based* navigation support mechanism [9], which takes into account the list of programming concepts presented by each example. To generate this list of each example, we developed the

automatic concept parser and an algorithm, which separates the list of concepts associated with each example into prerequisite concepts and outcome concepts. The process of separation is defined by the structure of a specific course, which is defined by having the teacher assign examples to the ordered sequence of lectures. For the examples associated with the first lecture, all of the concepts are considered outcome concepts. For each next lecture, the example concepts that do overlap with example concepts of the previous lectures are considered prerequisites, while the remaining concepts are considered outcome concepts. The algorithm advances through lectures sequentially until all example concepts are effectively split. The indexing and concept separation algorithms are discussed in more detail in [14].

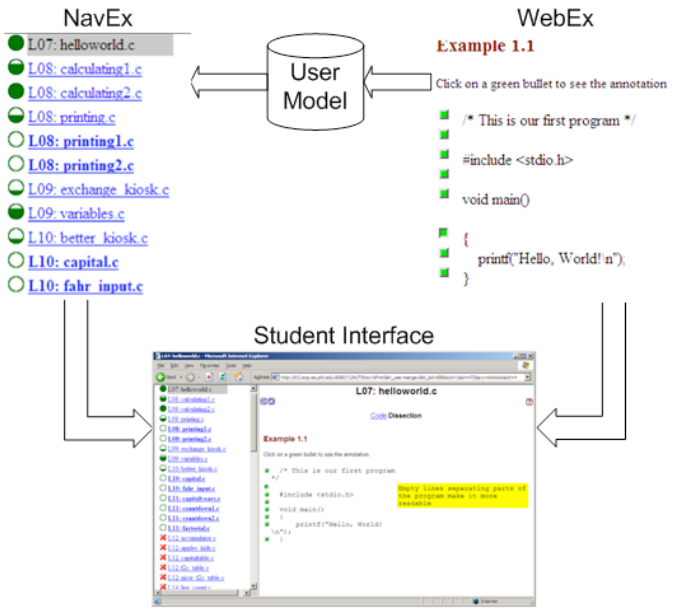

Fig. 6. NavEx and WebEx interaction

The prerequisites and outcome concepts and the current state of the individual user model determine which icon is shown next to the example link. Since the user model is constantly updated, the icon is selected dynamically.

When a user explores examples by opening teacher comments, WebEx sends events to the user model, one event for each line explored. Once an example is *sufficiently explored*, all of the example concepts are considered known. Sufficiency of exploration is defined by the relative amount of not-yet-known concepts (number of not known over the number of all concepts in the example) and also calculated dynamically. The smaller is the fraction of not known concepts, the fewer lines have to be explored to reach the sufficiency threshold. The exact formulas for user modeling and icon selection can be found in [9].

As more and more concepts become known, more and more examples become available for browsing. If all the prerequisite concepts of an example are known, the red X icon is replaced by a green bullet. The filling level of the bullet denotes the percentage of lines explored by the user. In the beginning there are very few examples available for browsing. In our case there was one "hello world" example. The more examples the user browses, the more concepts are learned, and thus more new examples are uncovered. However, the red X icon does not prevent user from actually accessing the example. The user is free to make own choice even if it is against the system's recommendation.

D. *A Classroom Study of NavEx*

A classroom study of both NavEx interfaces was performed in the context of an undergraduate programming course in the Spring 2004 and Fall 2004 semester in the School of Information Sciences at the University of Pittsburgh. In Spring 2004, NavEx was made available to students taking this course during the last weeks of the semester. In the Fall 2004 semester, it was made available after the midterm exam.

There were 23 active students working with NavEx in the first and 11 active students working in the second study. 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 using the original WebEx. Student work with both WebEx and NavEx was considered equally for the purposes of user modeling.

As in the WebEx studies, we collected subjective user feedback in the form of questionnaires. The questionnaires designed for WebEx were extended with additional questions about the NavEx component. In the Spring 2004 semester, we added two additional questions (#5 and #6, Table 2) and in Fall 2004 two more on top of that (#7 and #8, Table 2).

The summary of the answers for the questionnaire about the pilot version is shown in Figure 7. As we can see, the students' attitude to the core of the questions overlapping with WebEx questionnaires remains quite positive. The newly added feature of pilot NavEx – the "readiness" annotation – seems to be reasonably well received as well: positive responses were given by 75%-80% of students, although the percentage of strongly positive answers about these features is rather low.

The summary of questionnaire answers about the production version is shown in Figure 8. Feedback here, in general, is even more positive. Very positive responses to the questions about the value of the examples in general and their interactivity went up to 40% and 50%, respectively. Very positive responses to the interface and the content both crossed the previously unmatched 20% boundary line. Students were unanimous in voting on the positive side for having all examples together (100% of positive and very positive responses). About 80% of them liked the fact that their progress was now explicitly stated.

Table 2: Four additional questions added to questionnaire from Table 1

	Question

5. Value of Complexity Mark	NavEx estimated whether you were ready to understand a specific example and warned you about not ready to be explored examples using a red X icon. Regardless of the correctness of this estimation in the current version of NavEx, I think that it is useful to see "not ready" warning next to too complicated examples. 4. Strongly Agree 3. Agree 2. Neutral 1. Disagree
6. Quality of Complexity Estimation	NavEx estimated whether you were ready to understand a specific example and warned you about not ready to be explored examples using a red X icon. I think that NavEx estimation was mostly correct: 4. Strongly Agree 3. Agree 2. Neutral 1. Disagree
7. Value of Having Examples Together	The ability to access all dissections from all the lectures using the joint list of dissection on the left side of NavEx interface was helpful. 4. Strongly Agree 3. Agree 2. Neutral 1. Disagree
8. Value of Progress Meter	The ability to see my own progress in NavEx (the percentage of each example that I have already analyzed that was shown using fillable green bullets) was helpful. 4. Strongly Agree 3. Agree 2. Neutral 1. Disagree

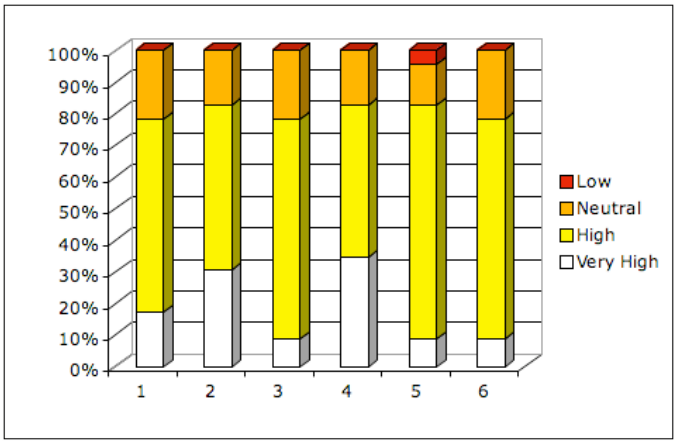

Fig. 7. Subjective evaluation of the pilot version of NavEx in Spring 2004

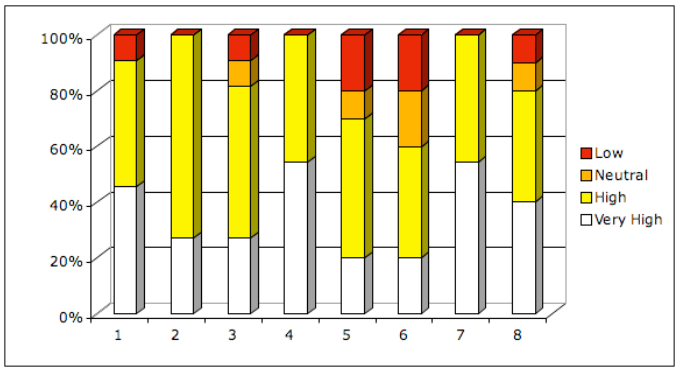

Fig. 8. Subjective student evaluation of different features of the production version of NavEx in the Fall 2004

The "readiness" indicator was appreciated less than other aspects with over 20% negative answers and less than 60% positive ones . We hypothesize that this could be attributed to the fact that NavEx was introduced in the middle of the Fall 2004 semester, when the students had already made good progress. NavEx was not aware of that progress, and students saw discouraging red X's telling them that they were not ready to explore certain examples, while their actual mastery of the material was already beyond this.

IV. NavEx vs. WebEx: The Motivational Value of Navigation Support

With two versions of example access being explored over 3 years, it was most interesting for us to compare NavEx and WebEx. We started with the subjective data. Comparison of questionnaire answers to four main questions (Table 1), which were used in all studies, is shown in Figure 9. Data covers the four semesters when the questionnaires were collected, namely, Fall 2002 and Fall 2003, when only WebEx was used, Spring 2004, when WebEx and the pilot version of NavEx were used, and Fall 2004, when WebEx and the production version of NavEx were used. Height of bars represents the amount of positive feedback given to a question.

Figure 9 shows that user feedback about the pilot version of NavEx was quite comparable with the feedback about WebEx; however feedback about the production version of NavEx is visibly more positive. Over 90% of students positively evaluated system's help in understanding the subject (question 1). Opinion about the interface and the interactive nature of example exploration reached the 100% positive level. The only place where the difference between the systems is negligible is on the question about content, which is not surprising, since the content did not change.

It is hard to say why the pilot version of NavEx was not received very enthusiastically. It could be the very late introduction of the system or the lack of progress-based support (or both). However, it is apparent that the production version, which was available for half of the course's duration, made an impressive impact.

To examine the nature of this impact, we decided to examine possible differences in user interaction between the original WebEx and the production version of NavEx. Our

main sources of interaction data were the user activity logs. The logs recorded every user click (i.e., every examined line). For each click, the log contained information about user identity, course, time of access, example accessed, and code line examined. The log record also indicated whether a student accessed a specific example through NavEx or through WebEx, during semesters when both NavEx and WebEx were available to the students.

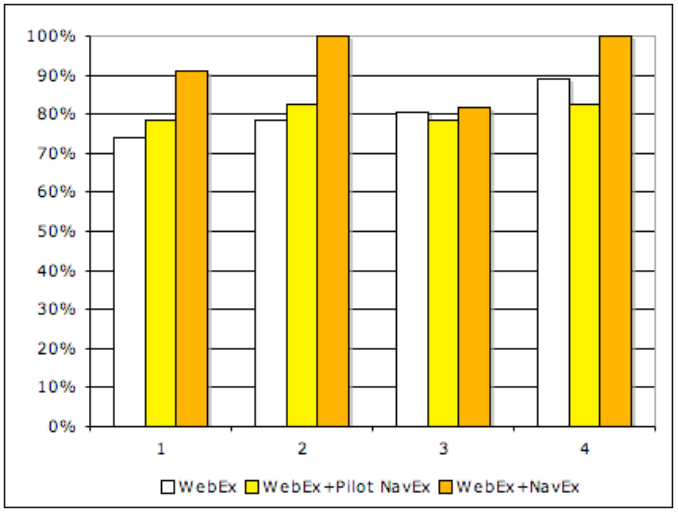

Fig. 9. Comparison of positive feedback in Fall 2002, Spring 2003 (WebEx), Spring 2004 (WebEx and pilot NavEx), and Fall 2004 (WebEx and production NavEx) semesters for the "overlapping" questions 1 through 4.

The comparative analysis of the WebEx and NavEx system usage covered three semesters worth of log data (Fall 2003, Spring 2004, and Fall 2004). We examined system usage with respect to 3 variables: number of clicks (lines explored), number of examples explored, and number of lectures covered (since each example belonged to one of the course lectures). We have looked at these variables from two perspectives: overall per-user average and per-user per-session average.

Table 3: Analysis of the means of the variables for semesters when WebEx and NavEx were used.

		WebEx	NavEx+ WebEx	p-value
Overall statistics	Clicks	34.76±6.66	171.90±65.56	<.001***
	Examples	5.66±0.87	18.10±4.32	<.001***
	Lectures	3.52±0.42	8.20±1.23	<.001***
Average session statistics	Clicks	7.85±0.87	9.49±1.28	.122
	Examples	1.56±0.12	2.03±0.22	.013*
	Lectures	1.20±0.05	1.37±0.10	.020*

* p-value <.05, *** p-value <.001

The analysis (Table 3) has shown an impressive growth of system usage. During the semester when NavEx was made available, students explored nearly 5 times more code lines (≈35 vs. ≈170), and accessed 3 times more examples (≈6 vs. ≈18), which covered twice as many lectures (≈4 vs. ≈8). All of the differences are significant. Although an average session of NavEx usage is not significantly longer than the average session of WebEx in terms of clicks, the students tend to come back to NavEx three times more often then they do to WebEx. In a sense NavEx becomes "addictive," once students are exposed to it. As for the number of examples explored and lectures covered, NavEx significantly surpasses WebEx even within a single session.

We have also investigated whether the increase in the number of lines and examples explored happened mainly due to the growth of activity with the examples associated with the current lecture or because the navigation support of NavEx encouraged students to access under-explored examples associated with previous lectures. Results show that the amount of user clicks on examples of past lectures that were not in the current focus of the class were only 25% percent of the total number of clicks for WebEx. The same value for NavEx is significantly higher – 51% (Table 4). Also the backtracking distance – how far in terms of lectures students go back – is approximately twice as large for NavEx (≈18 vs. ≈9).

Table 4: Summary of WebEx and NavEx usage logs.

	WebEx	NavEx+ WebEx	p-value
Back-track ratio	0.24±0.05	0.51±0.08	.005**
Back-track distance	8.73±1.90	17.64±2.51	.002**

** p-value <.01

V. Summary and Future Work

This paper gives an overview and the evaluation results of our 5-year project focused on a new kind of educational content: interactive explained program examples. We argue in favor of the importance of this kind of content and present our attempts to deliver this content to students of regular introductory programming courses, using two systems: WebEx and NavEx. WebEx provided a Web-based interface to interactively explore explained examples. NavEx extended WebEx with adaptive guidance, provided by adaptive link annotation. This technology was applied to guide students to the most appropriate examples and to encourage students to explore code examples more frequently.

The classroom studies of WebEx and NavEx, also summarized in this paper, demonstrated that interactive program examples are highly praised by students for helping them understand programming concepts. The students also very positively rated the system interface, interactive nature, and specific interface features. The introduction of NavEx further increased student satisfaction with the system, confirming that adaptive navigation support is a valuable feature in the context of example exploration.

We also discovered that the provision of adaptive guidance, in the form of adaptive link annotation, significantly increases student motivation to work with interactive examples. With NavEx, students accessed nearly 5 times more example lines and explored 3 times more examples. This finding provides

further evidence that adaptive guidance can significantly increase the amount of student work with non-mandatory educational content. Originally discovered in the context of student work with self-assessment questions [15] this phenomenon can be now generalized to other types of Web-based interactive content. This finding is very important for practitioners interested in using rich interactive educational content, since the lack of student motivation to explore and use new content is considered to be one of the major stumbling points to using modern technology within education [16].

We plan to continue our exploration of interactive examples. Currently, we plan to expand the work presented in this paper in three directions. First, we want to test the value of this technology in other domains, beyond its original scope of introductory programming. This semester, we are running a study of WebEx in the context of a database course where it is used to deliver explained-examples of SQL queries.

Second, we plan to investigate whether social navigation support mechanisms [17] could to some extent replace the concept-based navigation support mechanisms explored in NavEx. In a situation where concept extraction from examples is not possible, simple social mechanisms, such as footprints-based navigation support [18] could provide much needed guidance.

Finally, we are expanding our authoring system to allow the student authoring of annotated examples and peer review. Our early exploration [19] demonstrated that a properly engineered peer review process enables a community of students to produce good explanations for program examples. We consider this direction of research as important, since student involvement in the authoring process can resolve a potential bottleneck of having an insufficient number of examples.

REFERENCES

[1] A. Kelley and I. Pohl, C by Dissection : The Essentials of C Programming. New York: Addison-Wesley, 1995.

[2] P. Brusilovsky, "WebEx: Learning from examples in a programming course," in Proc WebNet'2001, World Conference of the WWW and Internet, Orlando, FL, 2001, pp. 124-129.

[3] P. Brusilovsky and H. Nijhawan, "A Framework for Adaptive E-Learning Based on Distributed Re-usable Learning Activities," in Proc World Conference on E-Learning, E-Learn 2002, Montreal, Canada, 2002, pp. 154-161.

[4] P. Brusilovsky, "KnowledgeTree: A distributed architecture for adaptive e-learning," in Proc 13th International World Wide Web Conference, WWW 2004 (Alternate track papers and posters), New York, NY, 2004, pp. 104-113.

[5] P. Brusilovsky and R. Rizzo (2002). Map-based horizontal navigation in educational hypertext. Journal of Digital Information [Online], vol. 3, Available: http://jodi.ecs.soton.ac.uk/Articles/v03/i01/Brusilovsky/

[6] P. Brusilovsky, "Adaptive hypermedia," User Modeling and User Adapted Interaction, vol. 11, pp. 87-110, 2001.

[7] G. Weber and P. Brusilovsky, "ELM-ART: An adaptive versatile system for Web-based instruction," International Journal of Artificial Intelligence in Education, vol. 12, pp. 351-384, 2001.

[8] P. Brusilovsky, M. Yudelson, and S. Sosnovsky, "An adaptive E-learning service for accessing Interactive examples," in Proc World Conference on E-Learning, E-Learn 2004, Washington, DC, 2004, pp. 2556-2561.

[9] M. Yudelson and P. Brusilovsky, "NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples," in Proc 12th International Conference on Artificial Intelligence in Education, AI-Ed'2005, Amsterdam, the Netherlands, 2005, pp. 710-717.

[10] P. Brusilovsky and L. Pesin, "An intelligent learning environment for CDS/ISIS users," in Proc The interdisciplinary workshop on complex learning in computer environments (CLCE94), Joensuu, Finland, 1994, pp. 29-33. Available: http://cs.joensuu.fi/~mtuki/www_clce.270296/Brusilov.html.

[11] P. Brusilovsky, J. Eklund, and E. Schwarz, "Web-based education for all: A tool for developing adaptive courseware," in Proc Seventh International World Wide Web Conference, Brisbane, Australia, 1998, pp. 291-300.

[12] N. Henze and W. Nejdl, "Adaptation in open corpus hypermedia," International Journal of Artificial Intelligence in Education, vol. 12, pp. 325-350, 2001.

[13] K. A. Papanikolaou, M. Grigoriadou, H. Kornilakis, and G. D. Magoulas, "Personalising the interaction in a Web-based Educational Hypermedia System: the case of INSPIRE," User Modeling and User Adapted Interaction, vol. 13, pp. 213-267, 2003.

[14] S. Sosnovsky, P. Brusilovsky, and M. Yudelson, "Supporting Adaptive Hypermedia Authors with Automated Content Indexing," in Proc Second International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia at the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004), Eindhoven, the Netherlands, 2004

[15] P. Brusilovsky and S. Sosnovsky, "Engaging students to work with self-assessment questions: A study of two approaches," in Proc 10th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'2005, Monte de Caparica, Portugal, 2005, pp. 251-255.

[16] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. Á. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," ACM SIGCSE bulletin, vol. 35, pp. 131–152, 2003.

[17] A. Dieberger, P. Dourish, K. Höök, P. Resnick, and A. Wexelblat, "Social navigation: Techniques for building more usable systems," interactions, vol. 7, pp. 36-45, 2000.

[18] P. Brusilovsky, G. Chavan, and R. Farzan, "Social adaptive navigation support for open corpus electronic textbooks," in Proc Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004), Eindhoven, the Netherlands, 2004, pp. 24-33.

[19] I.-H. Hsiao and P. Brusilovsky, "Collaborative Example Authoring System: The Value of Re-annotation based on Community Feedback," in Proc World Conference on E-Learning, E-Learn 2007, Quebec City, Canada, 2007, pp. 7122-7131.

Peter Brusilovsky received an M.S. degree in applied mathematics from the Moscow State University in 1983 and a Ph.D. in computer science from the same university in 1987.

He is currently an Associate Professor of Information Science and Intelligent Systems at the University of Pittsburgh, where he also heads the Personalized and Adaptive Web Systems Lab. In the past, he held various appointments at the Moscow State University, Sussex University, Tokyo Denki University, the University of Trier, Free University of Bolzano, the National College of Ireland, and Carnegie Mellon University.

Dr. Brusilovsky is a member of ACM and AACE. He is also the current President of User Modeling Inc., a professional organization of user modeling researchers. He is a recipient of the Alexander von Humboldt Fellowship, the NSF CAREER award, and the E.T.S. Walton award.

Michael V. Yudelson was born in December 1978 in Smolensk, Russia. He received his M.S. degree in computer-aided design from Ivanovo State Power University, Ivanovo, Russia in 2001. He defended his Candidate of Science dissertation in computer-aided design at the same university in 2004.

In 2004, he joined the Ph.D. program at the School of Information Science, University of Pittsburgh, where he also works as a graduate student researcher at the Personalized Adaptive Web Systems Lab.

Mr. Yudelson is a student member of ACM. He has received the James Chen Student Paper Award at the International Conference on User Modeling 2007, and the Best Young Researcher Paper award at the International Conference on AI in Education 2007.