Assessing Student Programming Knowledge with Web-based Dynamic Parameterized Quizzes

Sharad Pathak and Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
Pittsburgh PA 15260
boysp_21@yahoo.com; peterb@mail.sis.pitt.edu

Abstract: Web-based knowledge assessment with on-line quizzes, is one of the oldest and most popular ways of using Web for educational purposes. Almost all the major Web courseware management system, support authoring and delivery of on-line quizzes, made of static questions. Our work attempts to address several known shortcomings of static questions. We suggest an approach and a system, to author and deliver Web-based dynamic parameterized quizzes for programming-related subjects. The system has been used in practical classes for two semesters with both desktop and mobile computers and for both assessment and self-assessment modes. This paper briefly describes the system and summarizes the results of its formal classroom evaluation.

Introduction

Objective tests and quizzes are among the most widely used and well-developed tools in higher education. Web-based knowledge assessment with on-line quizzes was the first to be implemented and currently the most well developed and most popular interactive component of Web-based education (Brusilovsky & Miller, 1999; Brusilovsky & Miller, 2001). All major Web courseware management systems, support authoring and delivery of on-line quizzes made from static questions. Lot of efforts have been put by vendors of such systems to support larger variety of question types, make questions authoring easier for a teacher, and provide support in managing quizzes and pools of questions. Yet, with all these powerful authoring tools some of the inherent problems of this traditional assessment tool - quizzes made of static questions can't be solved.

All these problems come from an obvious fact: when author creates one static question, one and only one question is created. Modern authoring tools made creating of Web quizzes much easier than it used to be[1], however, creating every question still involves a large amount of time because, if questions are taken seriously, creating the content takes much more time than entering this content into the system. Thus, every question is an expensive component of teaching material, and the number of questions that can be created for a particular course is relatively small. In this context, it is quite natural, that in both assessment and self-assessment contexts all students in the class are using the same set of static questions[2]. It creates at least two well-known problems:

- In an assessment context, the use of the same set of questions for the whole class provokes cheating. Cheating is quite an issue even for the classroom quizzes. For take-home and Web-based quizzes cheating become a major problem that actually prevents the teacher to rely on results of this kind of quizzes. Cheating is an issue not only within the class, but also between the classes as well. Sometimes same questions are being used in several classes (during the same term on during different terms) the students are known to accumulate and pass correct answers to their fellow students.
- In the self-assessment context, where the cheating is not a problem, the students usually suffer from the lack of material: There are usually too few questions to work with and one question can be answered only once.

[1] A number of our colleagues who were creating quizzes with the top-of-the-line Blackboard 5.1 system noted that the old "paper and pencil" approach is better. Creating quizzes in a text processor is faster (especially taking into account slow Web-based authoring tools). Automatic grading saves time in large classes, however, in many cases, quizzes are graded by Teaching Assistants and thus this saving does not affect the professors themselves.

[2] A more progressive approach where every student is getting a subset of a pool of questions developed for a quiz is used very rarely exactly because questions are expensive to develop.

A known remedy for this approach is the use of *parameterized questions.* With this approach, an author creates a pattern of a question. At the presentation time, the pattern can be instantiated with parameters taken from a particular set. Thus, every question pattern is able to produce a large or even unlimited number of different questions. Parameterized questions are currently a focus of one of the promising research directions in the field of Web-based education (WBE). A number of pioneer systems such as CAPA (Kashy et al., 1997), WebAssign (Titus, Martin & Beichner, 1998), EEAP282 (Merat & Chung, 1997), or Mallard (Graham, Swafford & Brown, 1997) have explored the use of parameterized questions in different settings. The early results are very encouraging. For example, the authors of the CAPA systems have carefully monitored the use of their system and reported that the use of parameterized questions can practically eliminate cheating (Kashy et al., 1997).

Our own work capitalizes on the experience of early systems with parameterized questions and is aimed at extending their application area to a non-traditional and challenging subject of programming languages. We have developed an approach and a system, to author and deliver Web-based dynamic parameterized quizzes for programming-related subjects. The system has been used in a practical class for two semesters with both desktop and mobile computers and for both assessment and self-assessment purposes. This paper briefly describes the system and summarizes the results of its formal classroom evaluation.

Parameterized Questions for Programming-related Subjects

Traditionally parameterized questions are used in math-based courses such as physics and various other math related subjects. In these application areas, a correct answer to a parameterized question can be specified by a formula that includes one or more question parameters. In this context, creating parameterized questions and checking the correctness of student answers is relatively easy: the question author provides the interval for the possible values of a randomized parameter (or parameters) and specifies the formula to calculate the correct answer. The quiz system generates a question and stores its parameters along with the content of the question. To check the student answer, the correct answer for the given set of parameters is calculated using the provided formula and compared with the given answer. The formula-based approach is simple and powerful. It even enables the system to recognize most typical misconceptions like an author may provide a "wrong" formula that corresponds to a misconception.

Unfortunately, the formula-based approach can't be used in programming except for very simple cases like calculating numerical expressions. Our challenge was to design a different approach that can work for a large variety of questions in programming courses. While analyzing several large pools of questions created for courses based on languages like Lisp, C, and Java, we found that a large portion of questions in these pools are code evaluation questions. In a code evaluation question, the user is provided with some fragment of a program and is asked to predict the value of a particular variable or a string to be printed at the end of execution of this fragment. These kinds of questions are very important as they enable the teacher to check the student's understanding of the semantics of programming language constructs. Semantics of a programming language is an important body of knowledge in itself and also a prerequisite to a higher level programming skills.

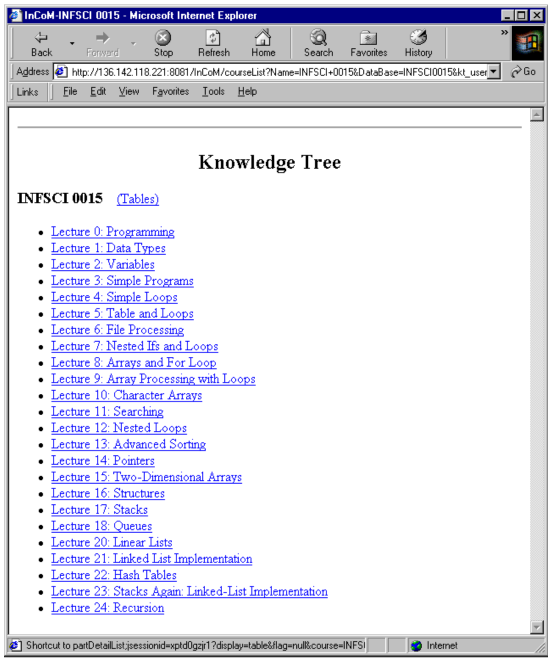
In traditional Web-based courses, code evaluation questions are very popular. They are authored much like any other kind of question, in the form of multiple choices or fill in the blanks using the tools that are provided for developing of static questions. They are evaluated by comparing the answer given by a student with the answer pre-specified by the teacher. However, it is known that code evaluation questions allow a different approach to checking the correctness of student's answers. Using a language interpreter or compiler the system can run the given fragment of code and calculate the correct answer without the need of a teacher to provide it. Then it can compare the calculated answer with student's answer. We have used this approach is several earlier systems such as ITEM/IP (Brusilovsky, 1992), ILEARN (Brusilovsky, 1994), and ELM-ART (Weber & Brusilovsky, 2001) to save some sizeable amount of question development time and to avoid authoring errors.

When developing our system QuizPACK (Quizzes for Parameterized Assessment of C Knowledge) we have attempted to combine our earlier work on code evaluation questions with the ideas of parameterized questions. Our motivation was to create a system that would allow teachers to create parameterized code evaluation questions and quizzes easier than creating static questions with all the existing authoring tools. The idea of QuizPACK is very simple. A teacher provides the core content of the question: a parameterized fragment of code to be evaluated and an expression that have to be evaluated by the student at the end of the fragment execution. The system does the rest: randomly generates a question parameter, creates a presentation of the parameterized question in a Web-based quiz, gets the student's input, compares the student's answer with the result of running the parameterized code "behind the stage", and records the results into a server-side database.

The current version of QuizPACK can support a whole programming-related courses based on C or C++ programming language. It supports simple questions based on a few lines of code as well as very advanced questions with the code, which include multiple functions and header files. QuizPACK allows the teacher to prepare full quizzes of different length in the form of static sequence of parameterized questions. It also allows a teacher to display various kinds of reports about the student's performance on quizzes. The following sections of the paper presents the student's and the teacher's side of QuizPACK and reports some result of its evaluation in a Data Structure course based on C language.

Students Interface of QuizPACK

Students are accessing the quizzes using our KnowledgeTree learning portal. The KnowledgeTree portal allows the teacher to create a course support Web site that can use course materials distributed among different servers. With knowledgeTree, a teacher is able to specify the objectives, readings, and other critical information for every lecture and to specify relevant learning activities of different kinds. In particular, a teacher can specify a set of quizzes to support the lectures. When a student selects a quiz in KnowledgeTree, the portal request the selected quiz from the quiz server and passes on the student parameters to the server. The server immediately loads the first question of the quiz in a separate window.



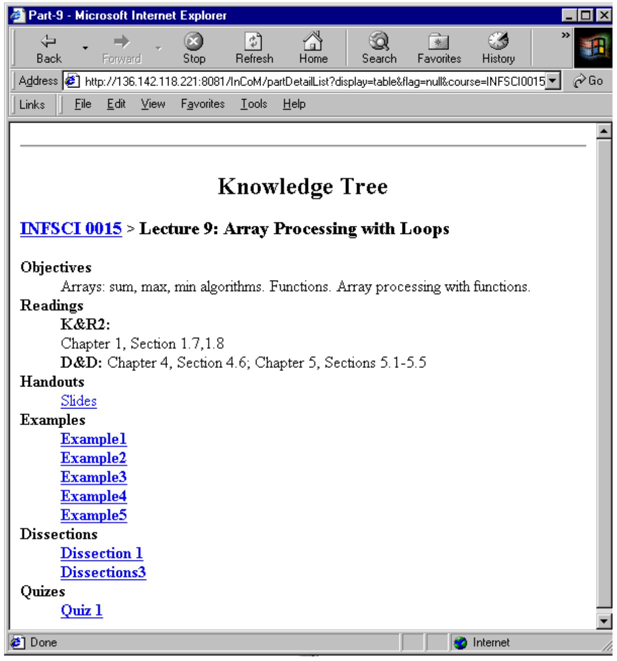
http://136.142.118.221:8081/InCoM/courseList?Name=INFSCI+0015&DataBase=INFSCI0015&kt_user

Knowledge Tree

INFSCI 0015 (Tables)

- Lecture 0: Programming
- Lecture 1: Data Types
- Lecture 2: Variables
- Lecture 3: Simple Programs
- Lecture 4: Simple Loops
- Lecture 5: Table and Loops
- Lecture 6: File Processing
- Lecture 7: Nested Ifs and Loops
- Lecture 8: Arrays and For Loop
- Lecture 9: Array Processing with Loops
- Lecture 10: Character Arrays
- Lecture 11: Searching
- Lecture 12: Nested Loops
- Lecture 13: Advanced Sorting
- Lecture 14: Pointers
- Lecture 15: Two-Dimensional Arrays
- Lecture 16: Structures
- Lecture 17: Stacks
- Lecture 18: Queues
- Lecture 20: Linear Lists
- Lecture 21: Linked List Implementation
- Lecture 22: Hash Tables
- Lecture 23: Stacks Again: Linked-List Implementation
- Lecture 24: Recursion

Shortcut to partDetailList;jsessionid=xptd0gzjr1?display=table&flag=null&course=INFSI



http://136.142.118.221:8081/InCoM/partDetailList?display=table&flag=null&course=INFSCI0015

Knowledge Tree

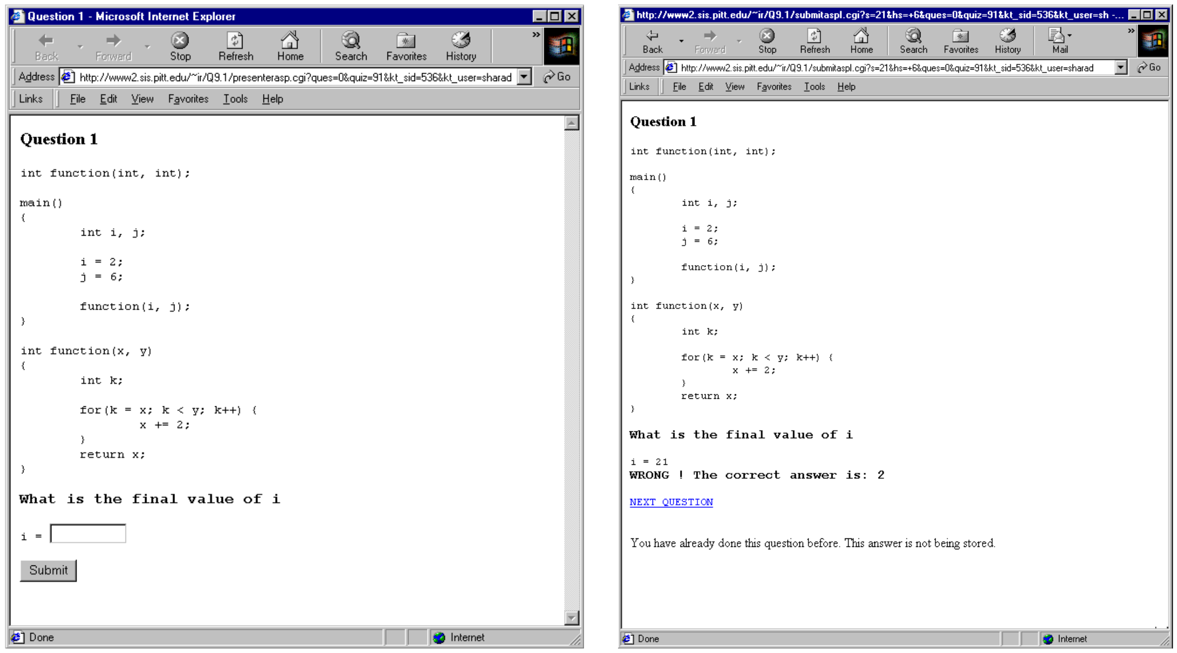
INFSCI 0015 > Lecture 9: Array Processing with Loops

Objectives
Arrays: sum, max, min algorithms. Functions. Array processing with functions.
Readings
K&R2:
Chapter 1, Section 1.7,1.8
D&D: Chapter 4, Section 4.6; Chapter 5, Sections 5.1-5.5
Handouts
Slides
Examples
Example1
Example2
Example3
Example4
Example5
Dissections
Dissection 1
Dissections3
Quizes
Quiz 1

Figure 1: KnowledgeTree, a portal for accessing distributed Web-based course support material. Left side: a list of lectures created by a teacher. Right side: a view of a lecture with associated learning material. Different items are usually served by different activity servers.

The parameterized question is presented to the student just like a normal static fill-in question. First, the program fragment to be evaluated is presented then the expression to be evaluated, then form box to write down the answer (Figure 2, left). One or more constants that the student can see in the body of the question are, actually, instantiated parameters. They are different for different students taking the quiz as well as for the same student attempting the question several times. In a properly designed question, the value of the expression to be evaluated depends on the instantiated parameters and thus will also be different for different students. Every student is supposed to mentally execute the program, evaluate the questioned expression, type in his answer in the textbox provided, and hit "submit" button. After that the system generates an evaluation screen for the student. The evaluation screen repeats the content of the question followed by the correctness of the student's answer followed by the correct answer if the student's answer was incorrect, and a link to the next question (Figure 2, right). This screen lets the student to re-think the question and the answer. In particular, the student may want to attempt the same question again by using the Back button and reloading the question screen. The student can attempt the same question many times, though in evaluation context only the results of the first attempt are recorded (Figure 2, right). Students may finish working with the quiz anytime simply by closing the quiz window and returning to the KnolwedgeTree portal. Note that the student interface does not support elaborated graphics and uses the minimal required space to present the question and the feedback. It was done to enable working with questions on multiple handheld wireless devices. As a result, the QuizPACK perfectly works on small Windows CE computers like HP Jornada and even on Palm-based organizers.

We welcome the readers to try QuizPACK system by logging in to KnowledgeTree system at http://dbpc.sis.pitt.edu/sharad/kt/login.html. The system allows self-registration. We suggest the use of your e-mail address for your login name to avoid name conflicts.


Question 1 - Microsoft Internet Explorer
Back
Forward
Stop
Refresh
Home
Search
Favorites
History
Address
http://www2.sis.pitt.edu/~ir/Q9.1/presenterasp.cgi?ques=0&quiz=91&kt_sid=536&kt_user=sharad
Go
Links
File
Edit
View
Favorites
Tools
Help
Submit
Done
Internet
http://www2.sis.pitt.edu/~ir/Q9.1/submitaspl.cgi?s=21&hs=+6&ques=0&quiz=91&kt_sid=536&kt_user=sh -...
Mail
http://www2.sis.pitt.edu/~ir/Q9.1/submitaspl.cgi?s=21&hs=+6&ques=0&quiz=91&kt_sid=536&kt_user=sharad

Question 1

```
int function(int, int);

main()
{
        int i, j;

        i = 2;
        j = 6;

        function(i, j);
}

int function(x, y)
{
        int k;

        for(k = x; k < y; k++) {
                x += 2;
        }
        return x;
}
```

What is the final value of i

i =

Question 1

```
int function(int, int);

main()
{
        int i, j;

        i = 2;
        j = 6;

        function(i, j);
}

int function(x, y)
{
        int k;

        for(k = x; k < y; k++) {
                x += 2;
        }
        return x;
}
```

What is the final value of i

```
i = 21
```
WRONG ! The correct answer is: 2

NEXT QUESTION

You have already done this question before. This answer is not being stored.

Figure 2. The user interface of QuizPACK.

QuizPACK from the Teachers Point of View

Teachers use QuizPACK to publish online parameterized quizzes for the students. There are three major steps a teacher has to complete in preparing a quiz.

- First, the teacher has to provide the programs to be evaluated. The code of each program has to reside in a separate file. The place for the randomized parameter in the code of the question, is denoted by a pseudo-variable Z. During the question generation phase, which takes place only in the browser window when the student requests it, QuizPACK will substitute every occurrence of Z variable will by an integer randomized within the interval provided by the teacher. In effect, a teacher defines a pattern of a question. A pattern is a simple way for a teacher to give different questions of the same difficulty level to all the students without being biased with the choice of questions for different students.
- Second, the teacher has to specify a quiz in a special quiz "index file". Each line in this file corresponds to one question. For each question the teacher has to specify the name of the file with question code, the lower and the upper limits for randomized variable, an expression to be evaluated by the students, and a type of this expression (i.e., *int* or *float*). With this approach, the code of the question in QuizPACK is separated from the pedagogic information such as randomization limits and expression. It allows the author to use the same question code in different quizzes with possibly different parameters.

- The last step is to compile a quiz. QuizPACK system is developed so that it can work on any Unix system with variety of Web servers. The system itself is a collection of several prototype data files and several commands (in Unix sense). The commands are all written in C and can be recompiled on any Unix platform. The authoring interface for compiling a quiz consists of just two commands. The *newquiz* command generates a folder (catalog) for a new quiz with all required prototype files in it. The name of this folder is also the name of the quiz. This is the folder where teacher have to place the question files and the quiz "index file" for the quiz. Naturally, these files can be simply created inside the quiz folder using any editor, moved from other folders or transported by FTP from another computer (including Macs and PCs). After all files are added to the quiz folder, the author has to execute *compile* command that generates and compiles a set of CGI programs for the quiz. More exactly, two CGI programs are generated for each question - one to present the question and the other to evaluate the answer. The code of the original question is included into both of these programs. Thus, the same code provided by the teacher is used for both - presenting the question to the student and for evaluating the answer. If the author will decide to change or update the quiz, the *compile* command should be used to re-generate the quiz.

QuizPACK authoring aproach allows the authors to create and compile quizzes remotely using Telnet and FTP tools. Note that the authoring side of QuizPACK does not use traditional Web form-based authoring approach, but relies on generic text editing and communication tools. For teachers of programming subjects it's a benefit, not a shortcoming. A full-featured randomized quiz can be produced very quickly, with the use of familiar tools on their preferred platform and with minimal authoring efforts over creating the code for questions. Teachers can publish the questions from their homes and then can just e-mail the students to take the quiz. They don't need to come to the class and distribute the papers to the students to take the quiz.

QuizPACK use a specific http-based protocol to communicate the results of the student work with every question to a server specified by a teacher. We have successfully used this http-based approach to support communication between distributed components of educational software in our previous work (Brusilovsky, Ritter & Schwarz, 1997). With this approach any http server can be used to log or store the results of the student's work. For the needs of our own course we have implemented a simple storage server using Microsoft Internet Information Server and Microsoft Access database. The database stores results of the student's work in a very simple format. This is a convenient solution since allows to produce easily a variety Active Server Pages (ASP) scripts for creating student progress reports. With these reports, teachers can evaluate the performance of the class or a particular student on a quiz or question level by calling the URL for one of the ASP scripts.

Evaluation of the System

During the Fall semester of 2002 we have performed a formative evaluation of QuizPACK in a real university course "Data Structures and Programming Techniques" at the University of Pittsburgh. The course was based on C programming language. During the first part of the course, we have use QuizPACK for in-class assessment mode using wireless handheld computers. During the second part of the course we have used the system for out-of-class self-assessment mode using regular desktop computers. The analysis of the in-class use of the system is beyond the scope of this paper since the results were significantly influenced by the use of wireless computers. Below we discuss some results of the evaluation of QuizPACK in the self-assessment mode.

QuizPACK was used to serve self-assessment quizzes of 5 to 10 questions each for almost every lecture of the second part of the course. The questions were prepared to let the student practice the knowledge of programming constructs and data structures introduced in the corresponding lectures. In addition to self-assessment computer quizzes we also used regular paper-and-pencil assessment quizzes at the beginning of every lecture. A few weeks before the end of the course we have prepared a short 9-question assessment tool to collect the students feedback about QuizPACK. The students were informed about the questionnaire and were encouraged to fill it in. The participation in the study was voluntary. All participants were rewarded extra credit points for their participation. In total, 27 students of 40 choose to participate. It's important to note that all students participated in the study have taken at least three self-assessment quizzes over the duration of the course.

Overall, the students have got a very positive feeling about the self-assessment questions. 48% of respondents thought that it "can significantly help", 37% thought that it "can help", 15% thought that it "can sometimes be of help" and none found them useless. The ability to take a parameterized quiz several times was also highly evaluated. 56% found this feature "very useful", 26% found it useful, 19% found it "sometimes useful": and none found it "useless".

The last of the questions in the questionnaire allowed the students to provide an unstructured feedback. We have specially welcomed critical feedback and suggestions. While we have not asked about praise, many students have provided a very encouraging feedback from unspecific: "The self-assessment quizzes were of great help" to very specific: "The quizzes helped me to figure out link list implementation". Students also appreciated the parameterized nature of the quizzes: "You can also take the self-quizzes over and over again unlike the [paper-and-pencil] in class quizzes. You learn more by taking different quizzes over and you didn't have to stress yourself over getting the answer right the first time". The students have also pointed out several problems and gave a number of suggestions. One of the most often mentioned suggestion was to provide self-assessment quizzes for every single course lecture, not just for a subset of them: "There were a few areas, I had trouble with, but there were no quizzes to go along with them". Many students also wished the tool not only reported the write answer, but also provide a detailed explanation of how this was obtained. We think that, such an explanation, can be generated by an explanatory visualization tool, and we are planning to explore this direction also. Finally, a number of students commented that they prefer a quiz format where all questions are shown on one page (as in paper-in-pencil quizzes). This format allows them to see the whole set of questions, take them in arbitrary order, and re-consider the answers before submitting the whole set.

Summary

We have reported an approach and a tool QuizPACK that enable the teachers to produce easily, parameterized quizzes for programming subjects. Parameterized quizzes greatly increase the productivity of teachers enabling them to generate a range of questions where regular tool allows them to produce only one question at a time. Parameterized quizzes prevent cheating in an assessment mode and allow the students to take the same question over and over in the process of mastering the subject. The tool was used in an undergraduate programming course and was very positively evaluated by the students. We are planning to continue the work in this direction taking into account student suggestions provided during the first formative evaluation.

References

Brusilovsky, P. (1994) ILEARN: An intelligent system for teaching and learning about UNIX. In: Proceedings of SUUG International Open Systems Conference, Moscow, Russia, April 25-29, 1994, ICSTI, pp. 35-41.

Brusilovsky, P. and Miller, P. (1999) Web-based testing for distance education. In: P. De Bra and J. Leggett (eds.) Proceedings of WebNet'99, World Conference of the WWW and Internet, Honolulu, HI, Oct. 24-30, 1999, AACE, pp. 149-154.

Brusilovsky, P. and Miller, P. (2001) Course Delivery Systems for the Virtual University. In: T. Tschang and T. Della Senta (eds.): *Access to Knowledge: New Information Technologies and the Emergence of the Virtual University*. Amsterdam: Elsevier Science, pp. 167-206.

Brusilovsky, P., Ritter, S., and Schwarz, E. (1997) Distributed intelligent tutoring on the Web. In: B. du Boulay and R. Mizoguchi (eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems*. (Proceedings of AI-ED'97, 8th World Conference on Artificial Intelligence in Education, 18-22 August 1997) Amsterdam: IOS, pp. 482-489.

Brusilovsky, P. L. (1992) Intelligent Tutor, Environment and Manual for Introductory Programming. *Educational and Training Technology International* 29 (1), 26-34.

Graham, C. R., Swafford, M. L., and Brown, D. J. (1997) Mallard: A Java Enhanced Learning Environment. In: S. Lobodzinski and I. Tomek (eds.) Proceedings of WebNet'97, World Conference of the WWW, Internet and Intranet, Toronto, Canada, November 1-5, 1997, AACE, pp. 634-636.

Kashy, E., Thoennessen, M., Tsai, Y., Davis, N. E., and Wolfe, S. L. (1997) Using networked tools to enhanse student success rates in large classes. In: Proceedings of FIE'97, Frontiers in Education Conference, Pittsburgh, PA, November 5-8, 1997, Stipes Publishing L.L.C., pp. 233-237.

Merat, F. L. and Chung, D. (1997) World Wide Web approach to teaching microprocessors. In: Proceedings of FIE'97, Frontiers in Education Conference, Pittsburgh, PA, November 5-8, 1997, Stipes Publishing L.L.C., pp. 838-841.

Titus, A. P., Martin, L. W., and Beichner, R. J. (1998) Web-based testing in physics education: Methods and opportunities. *Computers in Physics* 12 (Mar/Apr), 117-123, http://www.webassign.net.

Weber, G. and 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 Special Issue on Adaptive and Intelligent Web-based Educational Systems, To appear.