Adaptive Visualization Component of a Distributed Web-based Adaptive Educational System

Peter Brusilovsky and Hoah-Der Su

School of Information Sciences
University of Pittsburgh
Pittsburgh PA 15260
peterb@mail.sis.pitt.edu

Abstract. Adaptive visualization is a technology that can enhance the power of program visualization. The idea of adaptive visualization is to adapt the level of details in a visualization to the level of student knowledge about these constructs. This paper presents an adaptive visualization system, WADEIn, that was developed to explore visualization of expression execution during program execution - a under-explored area in visualization research. WADEIn has been designed as a component of our distributed Web-based adaptive educational system KnowledgeTree, however it also can be used as a standalone educational tool. The system has been pilot-tested in the context of a real university course with 40 students and is available on the Web for public use.

1 Introduction

Program visualization is one of the most powerful educational tools in computer science education. It can provide a clear visual metaphor for understanding complicated concepts and uncover the dynamics of important processes that are usually hidden from the student's eye. Many papers and projects have been devoted to visualization of program execution. Visualization has been explored in the context of machine level languages [9], various high level languages [12; 13; 18], and algorithms and data structures [1]. While several studies show the positive value of visualization, some other studies have demonstrated that visualization is not a silver bullet [10; 17]: often in the presence of a well-developed visualization the students still fail to understand what is happening inside a program or an algorithm. In our past research, we have explored several ways to improve the efficiency of visualization [3; 7]. One of the directions we have explored was adaptive visualization.

Adaptive visualization is based on an assumption that a student may have different level of knowledge of different *elements* of a program or an algorithm that is being visualized. For the case of a program, the student may know some high-level language constructs or machine level language commands better than others. For the case of algorithm animation the student may understand some steps of an algorithm better than others. In this context, regular visualization that animates all constructs or steps for each user with the same level of details may not be the best approach. For a troublesome construct the level of detail may not be deep enough to for the student to understand its behavior. At the same time, by showing a visualization of a well-understood construct with unnecessary details, the visualization system distracts the student and make it harder to focus on and thus comprehend the behavior of the constructs that are still poorly understood by the student.

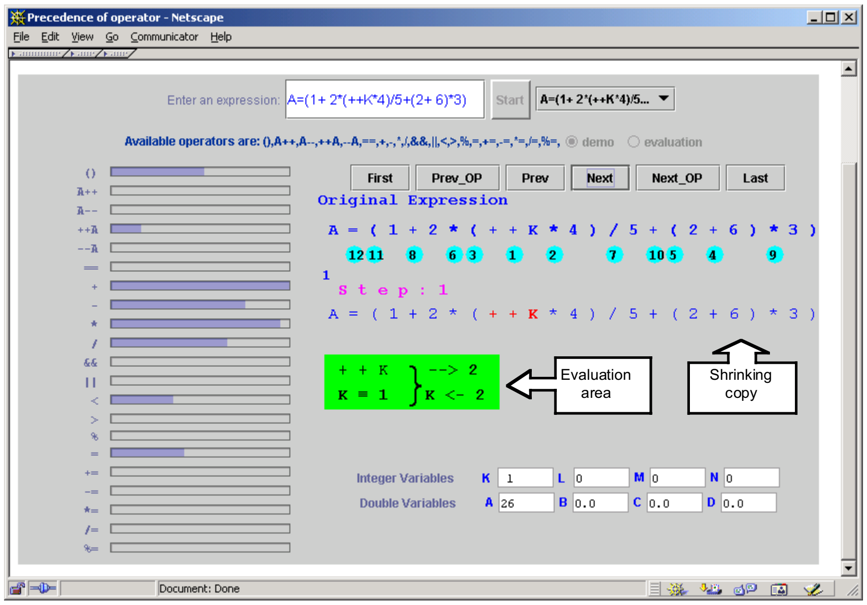
Adaptive visualization matches the level of details in visualization of each construct or step to the level of student knowledge about it. The less the level of understanding of a construct, the greater the level of details in visualization. Naturally, with a demonstrated increase in student knowledge about specific constructs, the level of visualization of those constructs should decrease. This approach allows a student to focus attention on the least understood components while still being able to understand the whole visualization. Our experimental evaluation of several kinds of enhanced visualization in the context of program debugging has confirmed our hypothesis that adaptive visualization can increase the power of visualization [2].

Our current work continues our research on adaptive visualization in the slightly different context of visualization of expression evaluation in C programming language. For the students of our programming and data structure course based on C language, the expression evaluation is one of the most difficult to understand parts. They have problems with both understanding the order of operator execution in a C expression and understanding the semantics of operators. To help the students, we have developed a Web-based Adaptive Expression Interpreter (WADEIn, pronounced as wade-in). WADEIn has been designed as a component of our distributed Web-based adaptive educational system KnowledgeTree; however it also can be used as a standalone educational tool. The system has been pilot-tested in the context of a real university course with 40 students and is available on the Web for public use. WADEIn and the technology of adaptive visualization used in it are the central topics of this paper. The following sections present, in order: the user interface, the architecture of WADEIn, and its use in two contexts: as a component of KnowledgeTree and as a standalone tool. At the end we discuss our research contribution and the plans of future work.

2 WADEIn: The User's View

The front-end of WADEIn is an expression interpreter that can work in either exploration or evaluation mode. In *exploration mode* the user can observe the process of evaluating a C expression step-by-step. An expression can be typed in or one can be selected from a menu of suggested expressions. At the beginning of evaluation, the system indicates the order in which various operations in the expression will be performed (Fig. 1). After that, the system starts visualizing the execution of each operation. The goal here is to show the results and the process of executing an operator. To show the results, the system visualizes a "shrinking" copy of the original expression and the values of all involved variables.

The execution of every operation is split in several sub-steps. At first, the system highlights the operations to be executed and its operands on the "shrinking" copy of the expression and re-writes it to the *evaluation area* field (gray rectangle in the center of the window). Next, it shows the *value* of the expression (2 on Fig. 1). In case of assignment and increment/decrement operations, WADEIn also shows the new value of the variable involved (Fig. 1). On the next sub-step, it replaces the whole highlighted operation in the "shrinking" copy with the calculated value. If a variable has changed its value as a side effect of evaluation, it also changes the value of the variable (Fig. 2). On this sub-step the system may use animation by "flying" the numbers from the working field to their destinations in an expression or in the variable area. Finally, the system removes all highlighting, "shrinks" the simplified expression, and prepares for the next step.



Precedence of operator - Netscape
File Edit View Go Communicator Help
Start
First
Prev_OP
Prev
Next
Next_OP
Last
() A++ A-- ++A --A == + - * / && || < > % = += -= *= /= %=
Evaluation area
Shrinking copy
Document: Done

Enter an expression: A=(1+ 2*(++K*4)/5+(2+ 6)*3)

A=(1+ 2*(++K*4)/5...

Available operators are: (),A++,A--,++A,--A,==,+,-,*,/,&&,||,<,>,%,=,+=,-=,*=,/=,%=, [✓ demo] evaluation

```
Original Expression

A = ( 1 + 2 * ( + + K * 4 ) / 5 + ( 2 + 6 ) * 3 )
  12 11  8   6 3  1   2     7   10 5   4     9
1
 S t e p : 1
A = ( 1 + 2 * ( + + K * 4 ) / 5 + ( 2 + 6 ) * 3 )

+ + K  } --> 2
K = 1  } K <- 2
```

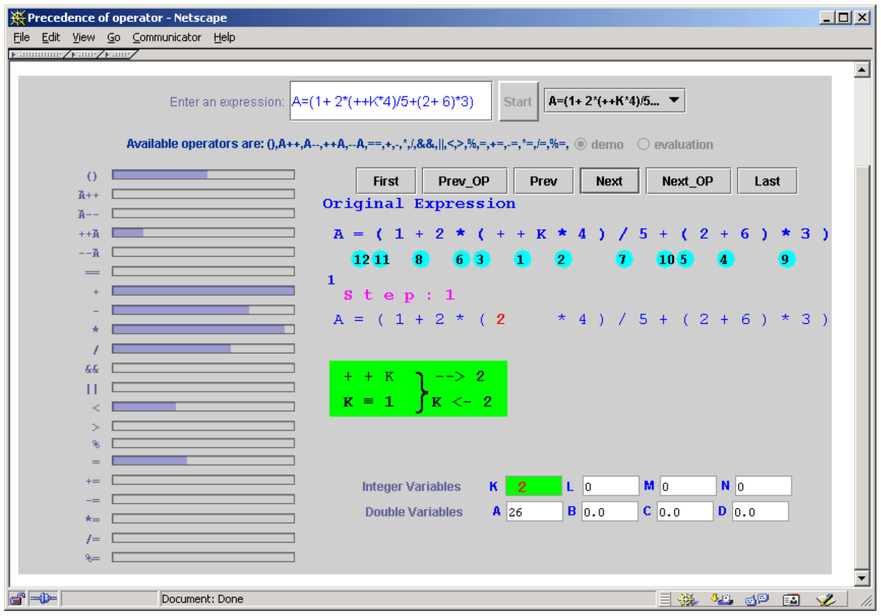
Integer Variables	K 1	L	0	M	0	N	0
Double Variables	A 26	B	0.0	C	0.0	D	0.0

Fig. 1. The user starts working with an expression. Numbers in circles show the order of calculation. The applet starts visualizing the execution of the first operation ++K (the current value of variable K shown below is 1).

The level of detail in executing an operation depends on the user's current level of knowledge about it. For the minimal level of knowledge (1.0), the system will perform all sub-steps and will show the animation in a slow motion. As the user learns an operation increasingly better and his or her knowledge level improves from 1 to 5, the system degrades gracefully the level of detail in the visualization by increasing the speed of animation and removing some sub-steps. For the maximal level of knowledge (5.0) there will be no sub-steps and no animation - the operation will be executed in one step.

To control the process of expression interpretation, the user has six buttons. *First* and *Last* let the user move to the beginning or the end of expression execution; *Next_OP* and *Prev_OP* let the user move in one step to the beginning of interpretation of the next or previous operation; *Next* and *Prev* move the user one adaptive step forward and backwards. Normally, the user would use only *Next* button (or simply hit

Return key). Other buttons can be used to watch the same operations and sub-steps again and again forwards and backwards or to skip sub-steps or operations.



Enter an expression: A=(1+ 2*(++K*4)/5+(2+ 6)*3)

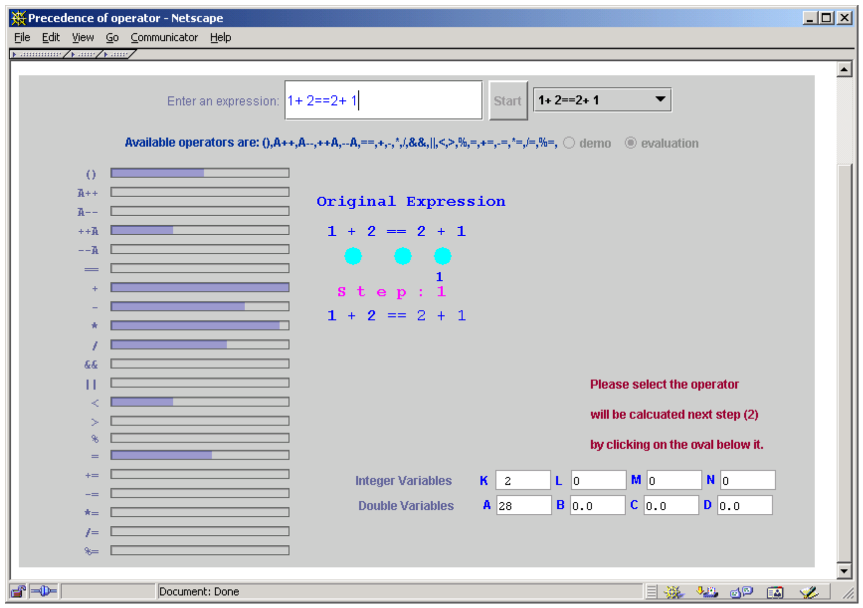
Available operators are: (),A++,A--,++A,--A,==,+,-,*,/,&&,||,<,>,%,=,+=,-=,*=,/=,%=,

```
Original Expression

 A = ( 1 + 2 * ( + + K * 4 ) / 5 + ( 2 + 6 ) * 3 )
      12 11   8   6 3   1   2     7   10 5   4     9
1
 S t e p : 1
 A = ( 1 + 2 * ( 2       * 4 ) / 5 + ( 2 + 6 ) * 3 )

 + + K  } --> 2
 K = 1  } K <- 2
```

Integer Variables	K	2	L	0	M	0	N	0
Double Variables	A	26	B	0.0	C	0.0	D	0.0

Fig. 2. The results of calculating an operation are "flying in" to replace the original operation and operands in a working expression and the old value of the variable involved.

To show to the user the system's opinion about his or her knowledge, we use progress indicator bars also known as a *skillometer*. This convenient interface feature that makes the user model viewable for a student was introduced in Carnegie Mellon cognitive tutors [16] and has been used since that in some other ITS [19]. One of our assumptions is that even passive watching of the visual execution of various operators contributes to student's knowledge about these operators. After visual execution of every expression, the progress indicators are updated. A more reliable way to check student knowledge and to update the student model is the *evaluation mode*. The student’s work with an expression in evaluation mode starts with a request to indicate the order of execution of the operations in the expression (Fig. 3). After that, the system shows the correct of execution, and starts evaluating an expression (Fig. 4). The process of evaluation is quite similar to the evaluation in an exploration mode (the execution of every operation is adaptively visualized), but two aspects are different. First, the students have no freedom in navigation through the solution. Only two actions are possible - quit an exercise and move to the next operator (Fig. 4). Second, if the student knowledge of the current operation is low, the system will not calculate the result of the operation, but instead requests it from the user (Fig. 4). If the user makes an error, the correct result is provided, so the calculation of the expression will always be correct.



Precedence of operator - Netscape

Enter an expression: 1+ 2==2+ 1

Available operators are: (),A++,A--,++A,--A,==,+,-,*,/,&&,||,<,>,%,=,+=,-=,*=,/=,%=, demo [✓ evaluation]

Original Expression

1 + 2 == 2 + 1

Step : 1

1 + 2 == 2 + 1

Please select the operator

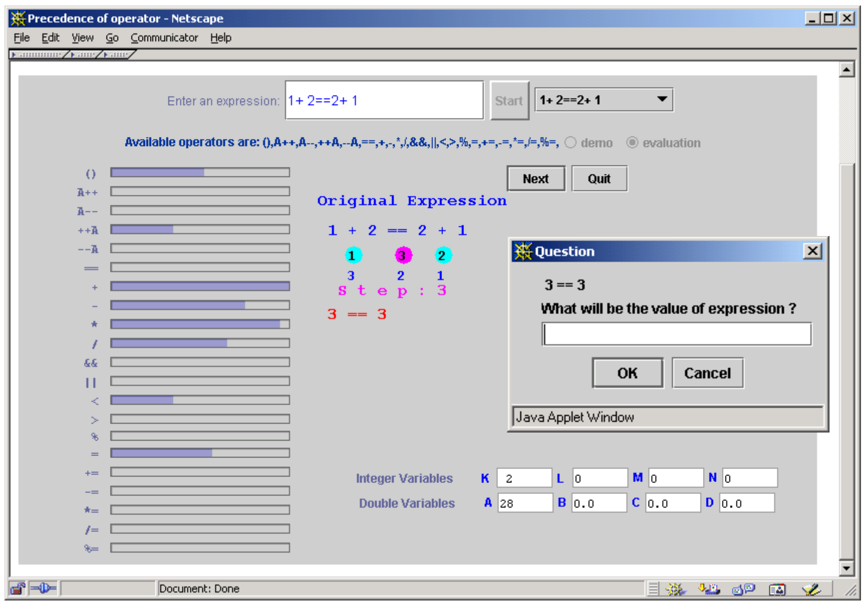
will be calcuated next step (2)

by clicking on the oval below it.

Integer Variables	K	2	L	0	M	0	N	0
Double Variables	A	28	B	0.0	C	0.0	D	0.0

Document: Done

Fig. 3. At the beginning of executing an expression in the evaluation mode, the system requests the user to mark the order in which the operators will be executed.



Precedence of operator - Netscape

Enter an expression: 1+ 2==2+ 1

Available operators are: (),A++,A--,++A,--A,==,+,-,*,/,&&,||,<,>,%,=,+=,-=,*=,/=,%=, demo [✓ evaluation]

Original Expression

1 + 2 == 2 + 1

Step : 3

3 == 3

Question

3 == 3

What will be the value of expression ?

Java Applet Window

Integer Variables	K	2	L	0	M	0	N	0
Double Variables	A	28	B	0.0	C	0.0	D	0.0

Document: Done

Fig. 4. In evaluation mode, the system requests the result of each operation from the user if his or her level of knowledge about this operation is low.

3 The Implementation

Architecturally, WADEIn is a distributed client-server application written in Java. It consists of a client-side applet and a servlet-based server side. The server side maintains authentication and provides a top-level interface for the user. It also accepts the information about the student progress from the applet, provides information about the student to all components and hosts the student model updating interface.

The client and the server side maintain two-way communication, however, each way is implemented differently. Server to client communication is maintained by parameter passing. The interpreter applet is very flexible and can be controlled by multiple parameters. The parameters define the mode of work (exploration, evaluation, or both), the starting level of user knowledge, the order and subset of operations visible on the progress indicator bar, the expressions that will be shown in the menu of expressions to choose, and a few other things. This flexibility allows the applet to be used in several contexts. In WADEIn systems the set of parameters are generated by a presentation server. More exactly, the server generates the whole HTML page that embeds the applet, including necessary parameters in the <applet> tag. In particular, it uses parameters to pass the current level of student knowledge about each operator. However, the requirement to use the interpreter applet with a server seriously reduces the applicability of this applet in a real classroom. The mechanism of parameters lets teachers who are interested to use the interpreter to create a static embedding page to host the applet. The values of parameters on this static page can be set to tune the applet to the needs of the class. In this context the applet will be able to function without the server component (though the student model will not be stored from session to session).

Client to server communication is essentially reporting the results of the student's work back to the server. After the student completes the work with an expression, the applet sends all information in the form of raw events to the user modeling server (next section explains it in more details). The communication with the user modeling server is implemented using a simple http-based protocol that is very similar to the one we have used in our earlier distributed system PAT-InterBook [6].

4 The Student Modeling

The student modeling component of WADEIn was developed following our *centralized user modeling* approach [4; 5]. One of the main ideas of this approach is that an educational system is composed of several adaptive components that all use the same central student model. The central student model assembles the information about the student from multiple sources. The student model is formed on the basis of the domain model that is a network of elementary knowledge elements. It stores an evidence of student knowledge for each of these knowledge elements separately. In our case, every C operator is an independent knowledge element. At the same time, the central student model is not a classic overlay that "cooks" all evidences about student knowledge of an element into a single number. In our central model the information is stored in a relatively "raw" form that avoids information loss and distinguishes different sources and events. This is important since the student model is used by different adaptive components that have different needs. Processing a flow of events into a classic vector overlay is usually done for the needs of one of the components and may lose an information important for other components.

WADEIn uses two different sources that can produce four kinds of events for the central student model. The main source is the interpreter applet that produces three types of events. The first type is "the student has seen a visual execution of an operator". The parameter for the event is the level of visualization. The second event is "the student has performed an operation". The third is "the student has identified the order of execution for an operator". The parameter for the latter two events is correctness. These events are sent to the user modeling server after the completion of every expression.

The second source of information is the student himself. The student model server maintains an open student model [8] and provides an interface where the students can self-evaluate their knowledge of every operator. All these events are stored independently and can be retrieved by any client of the central student model.

As mentioned above, the interpreter applet itself uses a regular overlay model for its own needs - the level of knowledge of each operator is modeled and visualized by progress indicators as a real number from 1 to 5. This overlay model is obtained as a *projection* of the central student model by applying a polynomial formula (1).

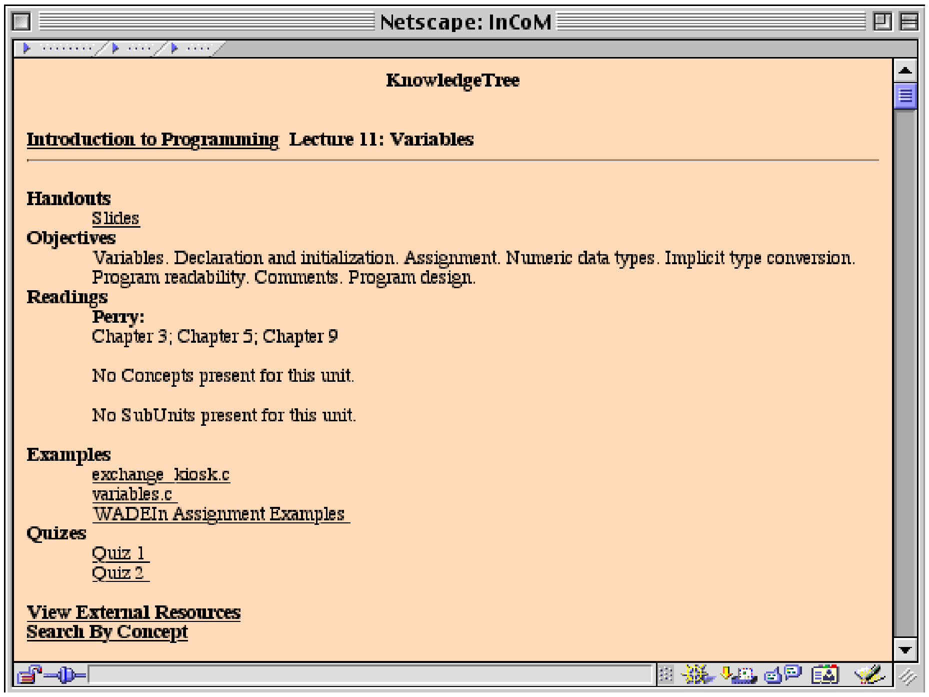
$$K = a_1 N_{user} + a_2 N_{seen} + a_3 N_{eval} \qquad (1)$$

Here $N_{user,}$ is the user own evaluation, N_{seen} is the number of times the user have seen an evaluation and N_{eval} is the number of times the user has done a correct evaluation. For experimental purposes, weights $a_{1\text{-}3}$ can be provided as applet parameters.

5 WADEIn as a Component of KnowledgeTree

WADEIn was designed to serve as one of the *activity servers* for our KnowledgeTree learning portal. The KnowledgeTree portal allows a 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 for every lecture and to request relevant learning activities of different kinds from different activity servers. At runtime the portal will retrieve relevant activities from different activity servers according to the objectives of the lecture and student knowledge (Fig. 5). The kinds of learning activities currently provided by WADEIn are a sets of expressions to be evaluated in either exploration or evaluation mode. Each set is developed to practice one or more operations.

Currently, KnowledgeTree/WADEIn does not support full-featured adaptive sequencing, but it does support mastery learning [11]: the student can work with the expressions until the target level of knowledge for the operations to be learned is achieved. In the future we are planning to add full adaptive sequencing of expressions to the WADEIn system - i.e., an ability to generate a small set of most appropriate examples at any point of a student's work with the system. This will let us use the system not only in conjunction with a course but also for self-guided Web-based education.


Netscape: InCoM

KnowledgeTree

Introduction to Programming Lecture 11: Variables

Handouts
Slides
Objectives
Variables. Declaration and initialization. Assignment. Numeric data types. Implicit type conversion. Program readability. Comments. Program design.
Readings
Perry:
Chapter 3; Chapter 5; Chapter 9

No Concepts present for this unit.

No SubUnits present for this unit.

Examples
exchange_kiosk.c
variables.c
WADEIn Assignment Examples
Quizes
Quiz 1
Quiz 2

View External Resources
Search By Concept

Fig. 5. KnowledgeTree, a portal for accessing distributed Web-based course support material. The window shows a view of a lecture with associated learning material. Different items are usually served by different activity servers.

6 Conclusion

This paper presents WADEIn system that lets the students explore the process of calculating the value of expressions in C language and evaluates the student knowledge of various C operators. WADEIn uses adaptive visualization to let students focus their attention on less understood C operators. The system is designed to be a component of a large adaptive educational system based on centralized student modeling. Currently we use WADEIn as a component of a learning portal KnowledgeTree, however, the interpreter applet that is a core of the system could also be used without any portal or server. The system was pilot-tested in a practical C course and got very positive student feedback. Many students considered WADEIn as the most useful tool among all tools that are currently connected to the KnowledgeTree portal. Currently we are running a larger and more formal evaluation of the system in an introductory programming class. Our plans are to provide WADEIn as a free tool that could be used for teaching programming in many places.

In addition to the practical need, the work on WADEIn was stimulated by two research goals. Our first goal is to develop an open architecture for adaptive distributed educational systems. To achieve a progress in this direction, we need set of diverse adaptive components that can be used to explore various aspects of the architecture and evaluate different student modeling approaches. In this context, an adaptive visualization system serves as one of the components.

Our second goal is to explore adaptive visualization as a way to increase the educational value of program visualization. It has been shown in several experiments [10; 17] that the educational effect of observing visualization is unexpectedly low. Different approaches to make visualization work have been suggested [3; 14; 15]. Adaptive visualization is one of these approaches. Our earlier experiments show promising results [3] and we want to continue this direction of work. Along this direction we plan a series of experiments with WADEIn to evaluate different aspects of adaptive visualization.

References

1. Brown, M. H. and Najork, M. A.: Collaborative Active Textbooks: A Web-Based Algorithm Animation System for an Electronic Classroom. In: Proc. of IEEE Symposium on Visual Languages (VL'96), Boulder, CO (1996) 266-275, available online at http://www.research.digital.com/SRC/JCAT/vl96
2. Brusilovsky, P.: Program visualization as a debugging tool for novices. In: Proc. of INTERCHI'93 (Adjunct proceedings), Amsterdam (1993) 29-30
3. Brusilovsky, P.: Explanatory visualization in an educational programming environment: connecting examples with general knowledge. In: Blumenthal, B., Gornostaev, J. and Unger, C. (eds.) Human-Computer Interaction. Lecture Notes in Computer Science, Vol. 876. Springer-Verlag, Berlin (1994) 202-212
4. Brusilovsky, P.: Student model centered architecture for intelligent learning environment. In: Proc. of Fourth International Conference on User Modeling, Hyannis, MA, MITRE (1994) 31-36
5. Brusilovsky, P.: Intelligent learning environments for programming: The case for integration and adaptation. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education, Washington, DC, AACE (1995) 1-8, available online at http://www.contrib.andrew.cmu.edu/~plb/papers/AIED-95.html
6. Brusilovsky, P., Ritter, S., and Schwarz, E.: Distributed intelligent tutoring on the Web. In: du Boulay, B. and Mizoguchi, R. (eds.) Artificial Intelligence in Education: Knowledge and Media in Learning Systems. IOS, Amsterdam (1997) 482-489
7. Brusilovsky, P. L.: Adaptive visualization in an intelligent programming environment. In: Gornostaev, J. (ed.) Proc. of East-West International Conference on Human-Computer Interaction, Moscow, ICSTI (1992) 46-50

8. Bull, S., Brna, P., and Pain, H.: Extending the scope of the student model. User Modeling and User-Adapted Interaction **6**, 1 (1995) 45-65
9. Butler, J. E. and Brockman, J. B.: A Web-based learning tool that simulates a simple computer architecture. SIGCSE Bulletin - inroads **33**, 2 (2001) 47-50
10. Byrne, M. D., Catarambone, R., and Stasko, J. T.: Evaluating animations as student aids in learning computer algorithms. Computers & Education **33**, 5 (1999) 253-278
11. Corbett, A. T. and Anderson, J. R.: Student modeling and mastery learning in a computer-based programming tutor. In: Frasson, C., Gauthier, G. and McCalla, G. I. (eds.) Intelligent Tutoring Systems. Springer-Verlag, Berlin (1992) 413-420
12. Domingue, J. and Mulholland, P.: An Effective Web Based Software Visualization Learning Environment. Journal of Visual Languages and Computing **9**, 5 (1998) 485-508
13. Haajanen, J., Pesonius, M., Sutinen, E., Tarhio, J., Teräsvirta, T., and Vanninen, P.: Animation of user algorithms on the Web. In: Proc. of VL '97, IEEE Symposium on Visual Languages, IEEE (1997) 360-367, available online at http://www.cs.helsinki.fi/research/aaps/Jeliot/vl.ps.gz
14. Hansen, S. R., Narayanan, N. H., and Schrimpsher, D.: Helping learners visualize and comprehend algorithms. Interactive Multimedia: Electronic Journal of Computer-Enhanced Learning **2**, 1 (2000)
15. Hundhausen, C. D. and Douglas, S. A.: Using Visualizations to Learn Algorithms: Should Students Construct Their Own, or View an Expert's? In: Proc. of IEEE Symposium on Visual Languages, Los Alamitos, CA, IEEE Computer Society Press (2000) 21-28, available online at http://lilt.ics.hawaii.edu/~hundhaus/writings/VL2000-Experiment.pdf
16. Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A.: Intelligent tutoring goes to school in the big city. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education, Washington, DC, AACE (1995) 421-428
17. Stasko, J., Badre, A., and Lewis, C.: Do Algorithm Animations Assist Learning? An Empirical Study and Analysis. In: Proc. of INTERCHI'93, New York, ACM (1993) 61-66
18. Tung, S.-H. S.: Visualizing Evaluation in Scheme. Lisp and Symbolic Computation **10**, 3 (1998) 201-222, available online at ftp://140.125.81.71/pub/tungsh/lasc.ps.Z
19. Weber, G. and Brusilovsky, P.: ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education **12**, 4 (2001) To appear