Personalized Guidance for Example Selection in an Explanatory Visualization System

Gayathri Krishnamoorthy and Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
Pittsburgh PA 15260
gak7@pitt.edu, peterb@pitt.edu

Abstract: A number of teams are pursuing research in the field of adaptive visualization. This paper provides a solution for offering students personalized guidance, through the Variable Scope Explanatory Visualization system, VARScope. The paper explains how the provision of an adaptive panel with annotated links in the form of visual cues can be used to suggest examples matching the student's current knowledge of variable scope. The logic behind the decision to suggest appropriate and inappropriate examples is also explained, and an evaluation of the system is included. In short, the paper offers a good demonstration of how a system can be personalized with relative ease, using an existing student model server.

1. Introduction

Program visualization is one of the most powerful tools in computer science education. Visualization can provide a clear visual metaphor for understanding complicated concepts and uncovering the dynamics of important processes that are usually hidden from the student's eye. Explanatory visualization systems can further increase the power of visualization by augmenting visualization with natural language explanations. The role of the explanations is to explain what is going on, why it happens, and how it relates to the semantics of the specific programming language (Brusilovsky & Spring 2004). Currently there are two types of systems providing explanatory visualization. One type of system is able to generate explanations on the fly (Kostadinov & Kumar 2003; Shah & Kumar 2002; Shanmugasundaram et al. 2004), while the other type is able to interactively present explanations created by a human author (Brusilovsky & Spring 2004; Lahtinen & Ahoniemi 2006). The first type of systems is typically used to deliver educational problems while the second type of system is used to present pre-authored examples. Since systems with pre-authored explanations are easier to create, the number of available examples with explanatory visualization is growing every year. The abundance of available examples creates a problem: how can one select examples that are the most relevant to the current state of the student's knowledge and learning goals, given dozens of potentially relevant examples?

This paper proposes a specific solution for the example selection problem: personalized guidance using adaptive link annotation techniques—developed in the field of adaptive hypermedia (Brusilovsky 2001). The system VARScope, presented in this paper demonstrates how adaptive link annotation can guide students to the most appropriate pre-authored examples that present various aspects of variable scope in the C programming language. VARScope also demonstrates how an existing explanatory visualization system can be extended with personalized guidance functionality. VARScope was built to help students in an introductory programming course learn the concept of variable scope and its use in C programming. According to the instructors' survey, variable scope is one of the critical topics in introductory programming and one in which explanatory visualization can be helpful for students (Brusilovsky et al. 2006). Unlike some earlier problem-oriented systems created to help students learn this topic (Kumar 2000), our work focused on explanatory visualizations of scope examples. The first version of VARScope, which was the result of a student course project, provided a number of interactive explained examples, but had no guidance functionality. The version presented in this paper combines explanatory visualization as well as personalized guidance.

The paper explains how a personalized version of the system can be developed with relatively ease, using an available student model server such as CUMULATE (Brusilovsky, Sosnovsky & Shcherbinina 2005). The paper provides a description of the VARScope interface, explaining the various panels used to offer different functionalities. It also portrays how personalized guidance is visualized in VARScope. The next section 'Learning with VARScope' summarizes the procedure followed while exploring examples with VARScope. 'The Implementation' section explains the logic behind example suggestion, which offers personalized guidance, and how this was realized using an available student model server. 'The Evaluation' section explains the result of our evaluation of the performance of VARScope and finally 'the summary' provides some future enhancements that might be achieved with VARScope.

2. The Interface of the VARScope System

The VARScope interface is shown in (Fig. 1). The interface is conveniently categorized into four main sections according to their functionality, namely, the "Adaptive section" on the left, the "Code Window" in the middle, the "Demo" section on the right, with the "Control Panel" at the bottom.

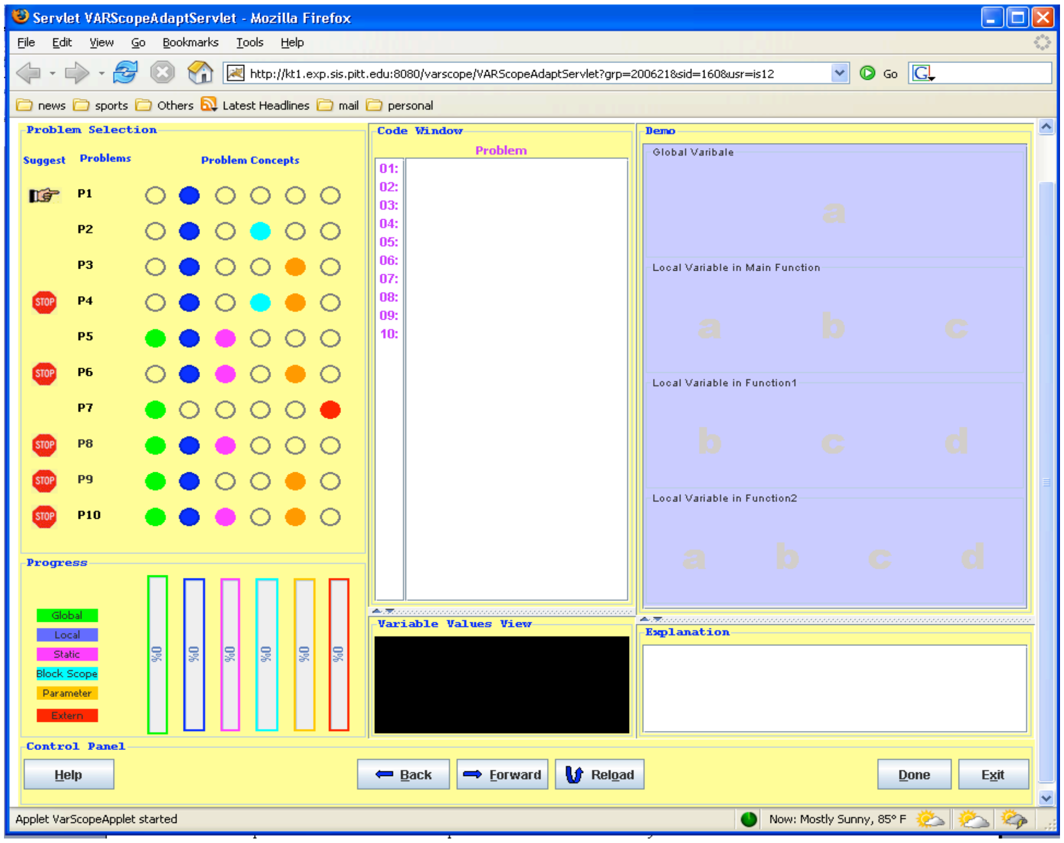

Servlet VARScopeAdaptServlet - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://kt1.exp.sis.pitt.edu:8080/varscope/VARScopeAdaptServlet?grp=200621&sid=160&usr=is12
news sports Others Latest Headlines mail personal
Problem Selection
Suggest
Problems
Problem Concepts
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
Progress
Global
Local
Static
Block Scope
Parameter
Extern
0%
Code Window
Problem
Variable Values View
Demo
Global Varibale
Local Variable in Main Function
Local Variable in Function1
Local Variable in Function2
Explanation
Control Panel
Help
Back
Forward
Reload
Done
Exit
Applet VarScopeApplet started
Now: Mostly Sunny, 85° F

Figure 1: The VARScope interface at the beginning of work with the system

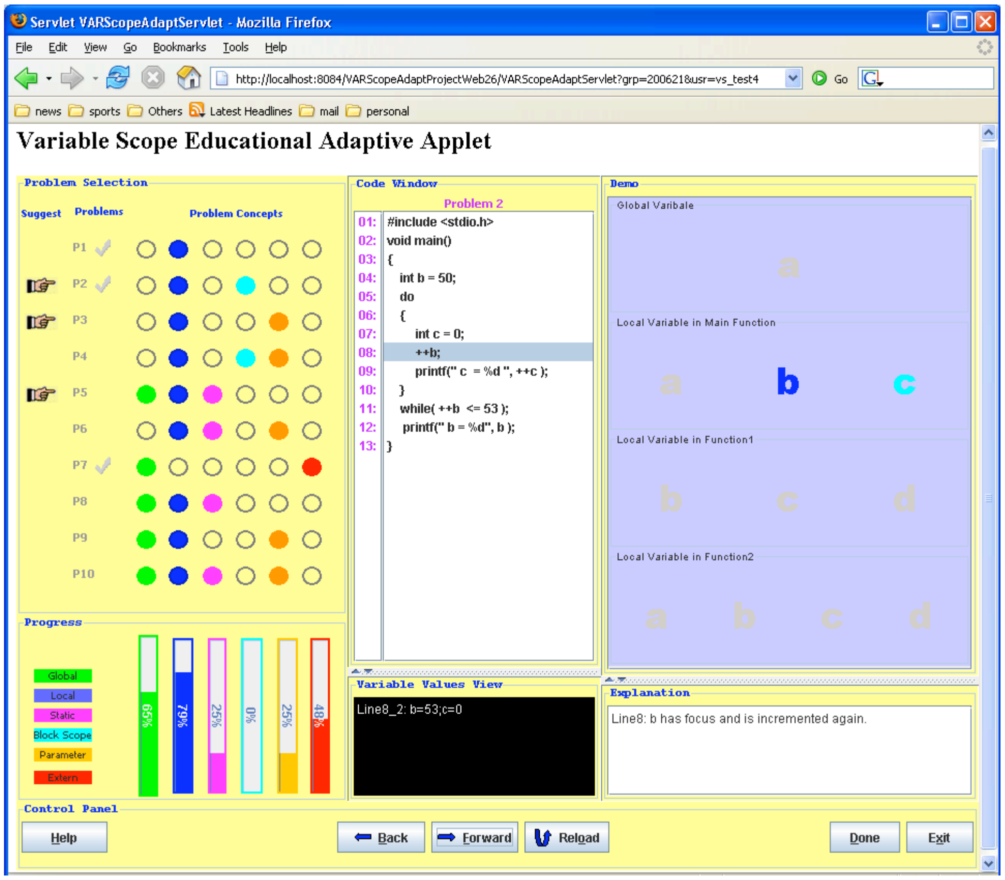
The adaptive navigation panel on the left side is the key to personalizing VARScope. It consists of the Example Selection and Progress panels. The *Example Selection panel* offers personalized guidance to students in the form of adaptive annotated links, suggesting the best examples and inappropriate examples at any given time, depending on the student's current knowledge of variable concepts. Visual cues in the form of a pointing hand and stop sign (Fig. 1) depict the most appropriate and inappropriate examples, respectively. These images were chosen to make visual perception easy for the student and guide them to the most appropriate examples. The fact that an example is considered by the system to be 'not appropriate' does not prevent the user from actually exploring it. All of the examples are available for exploration and it is up to the user to choose whether to follow the suggestions expressed by the annotations or not. Previously explored examples are marked with a checkmark.

To develop VARScope, we identified 6 domain concepts (Global, Local, Static, Block Scope, Parameter, and Extern) to capture knowledge about the domain, variable scope. Each example line was then indexed with these concepts, to track the student's knowledge of every domain concept. In the VARScope interface, each domain concept is uniquely color-coded and the concepts associated with each example are visualized using appropriately colored circles, in the form of a “concept grid.” The *progress panel* shows the progress made by the student on each concept and hence the knowledge gained, using a progress bar. The color selected for each concept is maintained across different panels, in order to familiarize the user with the concept color relation, as well as to give a pop-out effect.

The middle section consists of the Example *Code window* and the Variable Value View window. The code window is used to display example code when a student clicks on any example from the Example selection panel. Each line of code is clickable and the value of variables on the highlighted line is displayed on the Variable Value View window.

The right hand side consists of the *Demo window* and the *Explanation window,* which offers an explanatory visualization. The Demo window is further sub-divided into four different view panels. to systematically visualize the scope of variables within various functions (main, function1, function 2). At any given time, the Demo window visualizes the scope of different variables at a particular line of code, animating the active and hidden variables. The concept of the variable is depicted using the corresponding color chosen for the concept. At the same time, a detailed explanation of the code elaborating what is going on, why it happens and how it affects other variables, is displayed in the Explanation window, thus clarifying the concept to students.

The final section, at the bottom of the interface, displays the *Control Panel,* which captures the user’s input. It serves different functions, offering: 1) an entry point into the Help text, 2) a means of navigating the code via the Back, Forward and Reload buttons, 3) an exit from the example via the Done button and 4) exiting the system via the Exit button. The Help button opens a pop-up window, which contains explanation about the various panels used and mnemonic keys used.


Servlet VARScopeAdaptServlet - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8084/VARScopeAdaptProjectWeb26/VARScopeAdaptServlet?grp=2006218usr=vs_test4
Go
news sports Others Latest Headlines mail personal
Variable Scope Educational Adaptive Applet
Problem Selection
Suggest
Problems
Problem Concepts
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
Progress
Global
Local
Static
Block Scope
Parameter
Extern
65%
79%
25%
0%
25%
48%
Code Window
Problem 2
Variable Values View
Demo
Global Varibale
a
Local Variable in Main Function
a b c
Local Variable in Function1
b c d
Local Variable in Function2
a b c d
Explanation
Control Panel
Help
Back
Forward
Reload
Done
Exit

```
01: #include <stdio.h>
02: void main()
03: {
04:    int b = 50;
05:    do
06:    {
07:       int c = 0;
08:       ++b;
09:       printf(" c  = %d ", ++c );
10:    }
11:    while( ++b  <= 53 );
12:     printf(" b = %d", b );
13: }
```

```
Line8_2: b=53;c=0
```

Line8: b has focus and is incremented again.

Figure 2: The VARScope in the middle of user work. Progress bars show estimation of user knowledge. Completed examples are marked. The user is exploring line 8 in example 2 where local variable b and block variable c are visible.

3. Learning with VARScope

Students access VARScope by using a link from the Knowledge Tree portal (Brusilovsky 2004). Along with the link, the system records the student's name and generates a personalized example selection interface with annotated links. The examples already seen are marked with tick marks. The student's current progress level on various variable concepts is also indicated on the progress bars.

The student can then click on one of the suggested examples. This displays the chosen example in the code window and locks other example links. The student can then navigate through the example lines. The system highlights the current example line being explored, provides a natural language explanation of the line, to clarify the variable concepts seen, displays values of variables and visualizes different variables visible at that particular line in the Demo panel (Fig. 2). The student can explore each example line by either clicking directly on the code line or by using the Forward or Back buttons in the Control Panel. Information about each line exploration by the student is passed on to the student model to calculate the student's knowledge gain.

The student can keep advancing through the example by hitting the Forward button. The system also provides a way to go back and browse the preceding line of code, via the Backward button. The example exploration can be re-started from the beginning using the Reload button.

Once a student has understood the example and concepts shown and is comfortable to proceed with other examples, he/she hits the Done button. This action acquires progress level updates, examples-seen updates and calculates the next set of example recommendations. The interface is refreshed with the data thus acquired and the student proceeds with the next example, following the same steps.

4. The Implementation

VARScope is a client-server system written in the Java language. It consists of an applet on the client-side and two servlets, an initiator servlet and a relay servlet, for server-side functionality. The initiator servlet is used to the load the applet that displays the user-interface. The relay servlet is primarily used for handling communications with the student modeling server. Using the KnowledgeTree architecture protocols (Brusilovsky 2004), it sends information about the user's example exploration. After each example is completed, it retrieves the user's updated level of knowledge and the list of examples already explored by the user. This information is used to make decisions for personal guidance.

The VARScope html link, activated by the browser, contains user and group information and calls the initiator servlet with these user and group parameters. The servlet then loads the applet and passes on the user and group information to it. The applet in turn passes this information on to the relay servlet. The relay servlet communicates with the Student Model Server, CUMULATE, using the ADAPT2 http protocol to retrieve the user's current knowledge about the concepts identified and the examples already explored. The retrieved information is then passed back to the applet.

The applet uses the information received from student model for multiple functions and updates the user interface. First, it calculates the most and least relevant examples for the user, at that instant, in order to offer personalized guidance. Secondly, it updates the progress bar for all concepts. Third, it marks the examples already explored by the user, with a tick mark.

The personalized example suggestion works as follows: The number of domain concepts in each example is used to evaluate the difficulty of the example and hence influence the example suggestion. The difficulty of each example has been categorized into Levels 1, 2, 3 and 4, with the categorization being based on whether the example contains 1, 2, 3 or more than 3 concepts, respectively. Level 1 is on the easier side and is suitable for first-time exploration. Level 4 is the hardest, demanding the exploration of a set of lower level examples prior to its exploration. Each of the 10 examples used in VARScope has been categorized within one of the four difficulty levels. In general, unexplored, lower-level examples are given greater precedence over higher-level examples as being the most appropriate. Examples are then suggested, based on difficulty level and the user's current knowledge of the concepts. Examples already seen by the user are excluded. The methodology used is as follows: Examples already explored by the user are identified and categorized based on difficulty levels. The user's comfort level is calculated based on the number of examples explored under each difficulty level. This comfort level forms the basis for suggesting appropriate and avoiding inappropriate examples. If some examples are unexplored under the user's current comfort level, those examples are suggested as appropriate examples. Otherwise, appropriate examples are taken from the subsequent (higher difficulty) level. However, there are few exceptions to this general rule. Unexplored examples with known concepts are given a higher precedence over examples with unknown concepts, even if they have a higher difficulty level. Examples whose difficulty level is *much* higher than the user's current comfort level are considered inappropriate.

The user's exploration of every example line is communicated to the student-modeling server via the relay servlet. The relay servlet sends all information in the form of raw events, using parameter-value pairs. The student model hence updates the student model's knowledge level of the concept.

5. The Evaluation

One of the most important steps of any successful interactive system design is to analyze how the intended users will use the system that has been developed. In order to measure the success of the VARScope design and personalized guidance functionality, usability studies were conducted in the form of Questionnaires and Think-Aloud analyses. Six subjects having a wide range of knowledge in computer languages, from beginners to experts, participated in the Questionnaire. Three students, two being novices in programming (the target users of VARScope) participated in the Think-Aloud.

The Questionnaire had questions carefully framed, in order to capture subjective impressions on the specific goals that had been set for the system. The 11 questions in the questionnaire addressed the six main goals of the system, namely, easy navigability, learning goal achievement, interface design, ability to show progress, example suggestion, and amount of help provided. The questions queried the acceptance level (Strongly Agree, Agree, Neutral, Disagree, and Strongly Disagree) of the subject regarding specific aspects of the interface. Another section of the questionnaire had 4 additional questions, capturing general comments about the interface. The data obtained were plotted using a column chart for analysis (Fig. 3).

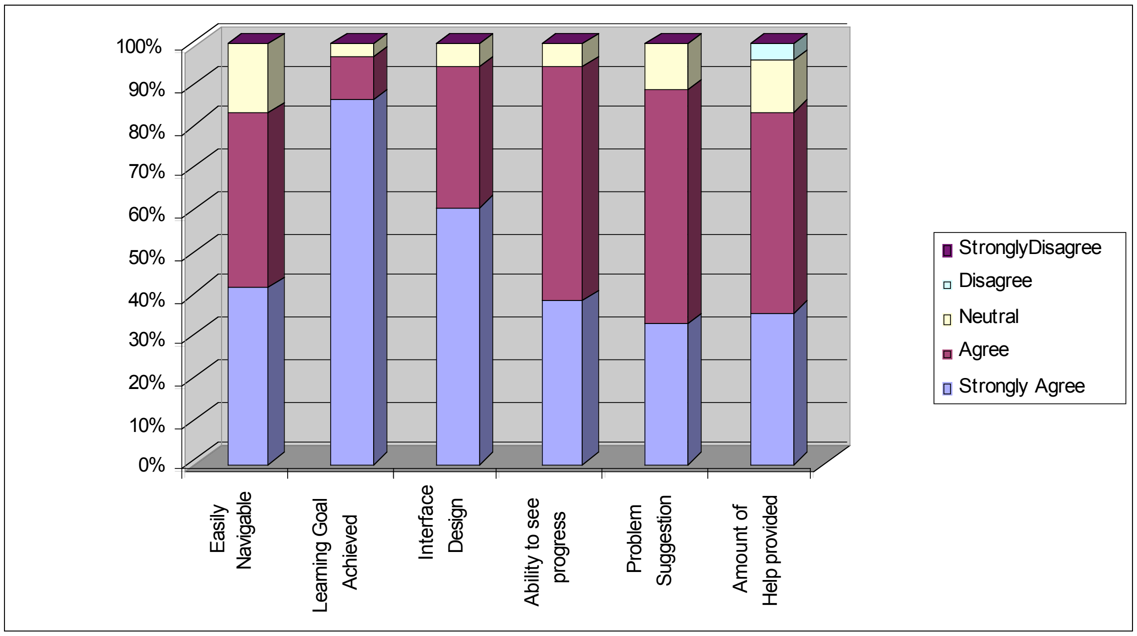


Figure 3: Subjective Student evaluation of different features of VARScope

As can be seen in (Fig. 3), 80% of the subjects found VARScope easy to navigate. Over 90% of the subjects agreed that the learning goal was achieved. This shows that the system was successful in accomplishing its purpose. The 90% satisfaction with the overall design of the interface shows its user friendliness. About 90% was achieved for the 'ability to see progress,' showing that subjects clearly liked the idea of visual feedback about the knowledge gained. The 85% achievement in example suggestion is recognition that subjects liked the idea of having the system suggest useful examples adaptively based on his/her knowledge. This, in turn, proves the success of the personalized guidance feature. About 80% of the subjects were satisfied with the amount of help the text provided.

Analysis of questionnaire and think aloud data, points out some of the most impressive and least impressive features of the interface. Subjects most liked the demo panel animating the variable scopes with a clear explanation of the line of code being focused on. They were also impressed with the progress bar constantly giving feedback about the knowledge gain and hence performance. The hand and stop symbols were considered to be the most appropriate visual cues for link annotation.

Subjects also had a chance to suggest improvements to the system. Some of the suggestions provided are as follows: The subjects requested the addition of a feedback mechanism for example completion. Subjects also suggested the provision of a separate panel explaining the different steps to be followed in sequence, when using VARScope. Another feature requested was to provide a stack trace display of function call.

6. Summary

This paper presented an explanatory visualization system, VARScope, that offers personalized guidance to help students select appropriate examples within the topic of variable scope. Visual cues advise students about the most and least appropriate examples. A progress indicator provides immediate feedback on knowledge gain and hence encourages the students. VARSCope can serve as a good demonstration of extending a collection of interactive examples with personalized guidance, using an available student model server.

The evaluation of the system through user studies provided very positive feedback from the students and hence shows the success of the personalized guidance feature added. One of the future enhancements could be to make the explanation provided for the program line personalized be based on the user's domain concept knowledge. Another enhancement might be to extend this "educational version" to develop an "evaluation version" that will measure the knowledge acquired by the user.

Acknowledgements

This research is supported by the National Science Foundation under Grant No. 0447083. The authors thank Shruti Parikh, Weidong Shi, Zhiwen Yu, and Guoliang Xiang, who developed the first non-adaptive version of this system.

References

Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User Adapted Interaction*, 11 (1/2), 87-110.

Brusilovsky, P. (2004). KnowledgeTree: A distributed architecture for adaptive e-learning. *The Thirteenth International World Wide Web Conference, WWW 2004* (Alternate track papers and posters), 17-22 May, 2004, ACM Press. 104-113.

Brusilovsky, P. et al. (2006). What should be visualized? Faculty perception of priority topics for program visualization. *SIGCSE Bulletin - inroads*, 38 (2), 44-48.

Brusilovsky, P., Sosnovsky, S., & Shcherbinina, O. (2005). User Modeling in a Distributed E-Learning Architecture. *10th International User Modeling Conference*, July 24-29, 2005, Springer Verlag. 387-391.

Brusilovsky, P., & Spring, M. (2004). Adaptive, Engaging, and Explanatory Visualization in a C Programming Course. *ED-MEDIA'2004 - World Conference on Educational Multimedia, Hypermedia and Telecommunication*s, June 21-26, 2004, AACE. 1264-1271.

Kostadinov, R., & Kumar, A.N. (2003). A Tutor for Learning Encapsulation in C++ Classes. *ED-MEDIA 2003, World Conference on Educational Multimedia, Hypermedia and Telecommunications*, June 23-28, 2003. 1311-1314.

Kumar, A.N. (2000). Dynamically generating problems on static scope. *The Fifth Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2000*, July 2000, ACM Press. 9-12.

Lahtinen, E., & Ahoniemi, T. (2006). Annotations for defining interactive instructions to interpreter based program visualization tools. *Fourth Program Visualization Workshop*, June 29-30, 2006. 34-38.

Shah, H., & Kumar, A.N. (2002). A tutoring system for parameter passing in programming languages. 7th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'2002, ACM Press. 170-174.

Shanmugasundaram, V. et al. (2004). Development of ViewJ - a Visualization Builder for Object Oriented Programming Development Environment. *ED-MEDIA'2004 - World Conference on Educational Multimedia, Hypermedia and Telecommunications*, June 21-26, 2004, AACE. 3828-3835.