# Evaluating the Educational Impact of Visualization

Report of the Working Group on "Evaluating the Educational Impact of Visualization"


Thomas Naps (co-chair)
U Wisconsin Oshkosh
naps@uwosh.edu

Guido Rößling (co-chair)
TU Darmstadt, Germany
roessling@acm.org

Jay Anderson
Franklin & Marshall College
jay.anderson@fandm.edu

Stephen Cooper
St. Joseph's University
scooper@sju.edu

Wanda Dann
Ithaca College
wpdann@ithaca.edu

Rudolf Fleischer
Hong Kong U Sc. & Techn.
rudolf@cs.ust.hk

Boris Koldehofe
Chalmers U Techn., Sweden
khofer@cs.chalmers.se

Ari Korhonen
Helsinki U Techn., Finland
archie@cs.hut.fi

Marja Kuittinen
Joensuu University, Finland
marja@cs.joensuu.fi

Charles Leska
Randolph-Macon College
cleska@rmc.edu

Lauri Malmi
Helsinki U Tech., Finland
lma@cs.hut.fi

Myles McNally
Alma College
mcnally@alma.edu

Jarmo Rantakokko
Uppsala U, Sweden
jarmo@tdb.uu.se

Rockford J. Ross
Montana State University
ross@coe.montana.edu



## ABSTRACT

The educational impact of visualization depends not only on how well students learn when they use it, but also on how widely it is used by instructors. Instructors believe that visualization helps students learn. The integration of visualization techniques in classroom instruction, however, has fallen far short of its potential. This paper considers this disconnect, identifying its cause in a failure to understand the needs of a key member in the hierarchy of stakeholders, namely the instructor. We describe these needs and offer guidelines for both the effective deployment of visualizations and the evaluation of instructor satisfaction. We then consider different forms of evaluation and the impact of student learning styles on learner outcomes.



## Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer & Information Science EducationComputer Science Education




*ITiCSE* 2003 Thessaloniki, Greece




## General Terms

Algorithms

## Keywords

Visualization, Animation, Pedagogy


## 1. INTRODUCTION

The educational impact of visualization includes two components: the enhancement of learning with visualization, and the deployment of visualization in the classroom. Previous surveys [35] show a significant disconnect between the intuitive belief that visualization enhances a student's learning and the willingness and ability of instructors to deploy visualization in their classrooms. A key impediment to the adoption of visualizations by instructors is the time required to learn, install, and develop visualizations and then integrate them into a course. Additionally, there is also a perceived lack of effective visualization development tools and software.

Whereas studies have begun to show the conditions under which visualization enhances student learning [35], the overall educational impact of visualization is and will be minimal until more instructors are induced to integrate visualization techniques in their classes. By continuing to explore the effect of visualization on student learning, and by overcoming the impediments to the deployment of visualization, we expect to raise the positive impact of visualization in computer science education.

Understanding the disappointing integration of visualiza-

tion techniques in the educational process requires that we identify the stakeholders in the instructional use of visualization. We recognize four different *roles* associated with visualization, similar to Price *et al.* [39]:

- *Visualization Tool Developers* develop tools for visualizing contents, such as algorithms, data structures or program execution. Such tools may provide visualizations directly, such as *Jeliot 2000* [32], or may be meta tools that allow others to design desired visualizations, for example *JAWAA* [2] or ANIMAL [43].
- *Visualization Designers* specify the mapping from the abstract concepts to their visual representation by creating specific visualizations – either by using visualization meta tools or by other more direct methods.
- *Instructors* incorporate visualizations into their teaching materials and methodology.
- *Learners* view and hopefully interact with the visualization in one or more of the engagement levels described in [35].

An individual may take on many of these roles. For example, instructors will often also act as visualization designers to generate their own specialized visualizations.

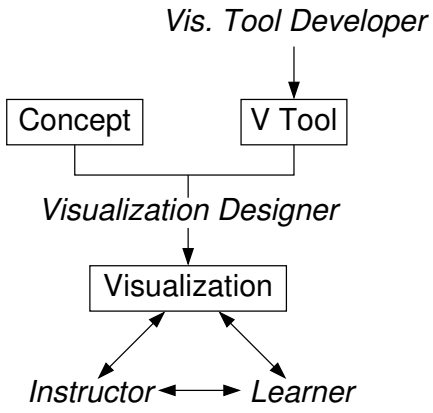


**Figure 1: Schematic View of User Roles.**

Figure 1 shows the different roles and how they can interact. Each role has specific expectations of visualizations. To gain more wide-spread usage, visualization tool developers are interested in optimizing their tool for the other three roles. Visualization designers strive to design visualizations that are valuable to a large audience. Instructors want to be able to integrate visualizations into their course materials to make both their teaching more satisfying and improve student motivation and learning. Students hopefully learn the concepts better or in a way that is "more fun" because of the visualizations.

Given this analysis of the stakeholders in instructional use of visualization, the reason for visualization's lack of impact begins to emerge. Visualization research has focused on the developer and designer while research in CS education has focused on the effectiveness of visualization to improve student learning. In contrast, virtually no research has focused on the needs of the instructor. As a result, support for deploying visualizations in the classroom is nearly always lacking.

Instructors face substantial impediments when they try to use visualization in their teaching. Based on a survey of SIGCSE members done by the 2002 Working Group [35], the top five impediments (listed by percentage of responses) are:

- 93%: time required to search for good examples
- 90%: time it takes to learn the new tools
- 90%: time it takes to develop visualizations
- 83%: lack of effective development tools
- 79%: time it takes to adapt visualizations to teaching approach and/or course content

Clearly, the amount of time involved in learning how to use visualization tools and in developing demonstrations and interactive lab materials discourages the use of visualization. Our thesis is that these impediments can be overcome by providing high quality support materials for instructors. The availability of good support materials will lead to increased instructor satisfaction, which consequently will lead to more widespread usage of visualization.

Working from this basic thesis, Sections 2 and 3 will offer guidance for enhancing the effective deployment of visualization tools in the classroom. Since impacting CS education requires both widespread use and improved student outcomes, such guidance will be broken down along these two lines. Section 2 will focus on the instructor satisfaction issues, and Section 3 will cover measurement of student outcomes. Section 2 will be of particular interest to developers and designers who seek feedback on how effective their tool is in teaching situations.

In future years, the combined results emanating from the evaluation techniques described will provide a more informative measure of the two aspects that combine to influence educational impact.

## 2. ADDRESSING INSTRUCTOR NEEDS

The introduction has made the case that visualization systems are perceived by instructors as being potentially beneficial to learning outcomes and motivation. However, such systems will see widespread use in the computer science curriculum only if instructors can be enticed to incorporate them into the fabric of their courses. This implies in turn that visualization tool developers must recognize and address the impediments instructors face in integrating visualizations into their teaching. In Section 2.1, we rely on the literature (see, for example, [7] and [33]) in providing a review of these impediments.

In Section 2.2, we offer advice to visualization system and tool designers for overcoming these impediments. Section 2.3 is devoted to techniques for making visualization software easy to locate and obtain. Finally, Section 2.4 explores ways of evaluating how successful a visualization system or tool is in meeting the needs of instructors.

## 2.1 Impediments Faced by Instructors

Without question, the main impediment to instructor adoption of visualizations for use in teaching and learning computer science concepts is *time*. This includes time to:

- Find
- Download
- Install
- Learn
- Develop visualizations (if the tool is one that assumes that the instructor will design his or her own visualizations via the tool)
- Adapt and integrate into a course
- Teach students to use visualizations
- Maintain and upgrade

Exacerbating this situation is the fact that this effort may all be done for just a couple of lectures and might need to be repeated for each concept to be visualized. Indeed, of the nine top impediments cited by instructors in the survey, six were time issues.

A second major impediment identified in the literature is *platform dependence*. If visualization systems are designed to run on a particular platform (for example, a PC with Windows), it precludes their use on another system (for example, a PC with Linux). Indeed, platform dependence has many more subtle nuances that a visualization tool designer must address (for example, version of operating system or browser used).

A third major impediment highlighted in the references is *course integration issues*. How easy is it to incorporate a visualization into the fabric of a course and to adapt it to the way the concepts are presented in the classroom and in the accompanying textbooks or other class resources? If the visualization does not integrate well into a course, it will most likely not be used.

Notice that although the course integration issue is highlighted separately in the literature, it is actually captured in the time impediment list as well, as adaptation and integration of a visualization system into a course are rightly also identified as substantial time sinks.

## 2.2 Advice

Visualization systems run a wide gamut. At the two extremes are:

- Standalone, single purpose, undocumented, platform dependent visualization systems that do not engage the student beyond passive viewing
- Complete teaching and learning resources that incorporate visualizations seamlessly and become an integral resource used for an entire course

Many visualizations have fallen into the first category and are precisely the ones that lead to instructor frustration as measured in the survey. They should be avoided by designers of such systems at all costs. Systems like this, while they may be usable by the designer in a local course for a specific purpose, are sure to not gain widespread use.

Hypertextbooks are an example of the other end of the spectrum. They are envisioned to be complete teaching and learning resources that complement—or indeed supplant—traditional course teaching and learning resources (for example, textbooks). Hypertextbooks may include standard text, illustrations, pictures, video clips (for example, QuickTime movies), audio, and various paths through the material based on different learning needs. Most importantly, they would also include embedded visualization applets of key concepts and models that engage the learner in active learning of the subject. In this case, since the hypertextbook *is* the course teaching and learning resource, and since the hypertextbook runs in standard browsers (and likely distributed on CD or DVD), the issues of finding, downloading, installing, adapting and integrating into a course, and maintaining and upgrading are moot. Since the visualization applets are part of the fabric of the hypertextbook, they will be used quite naturally by instructors and students alike. The applets themselves must be designed so that they are easy to learn by both instructor and students – issues we discuss below. More can be read about hypertextbooks in [6, 40, 41, 27].

Between these two extremes lie a variety of other possibilities for visualization systems and visualization tool development that address many of the impediments listed earlier. It should be noted that these suggestions are not mutually exclusive. Many can be combined to address more of the impediments. We provide some suggestions next.

**Design for platform independence.** This will, obviously, eliminate the impediment of platform dependence. Platform independence is an elusive and likely unattainable goal in its ideal sense, but there are some choices that are better than others. For example, designing systems for the Web (or, more precisely, the Java Virtual Machine) is one possibility. An alternative is to ensure that a visualization system runs on all of the major platforms likely to be available in academic settings around the world. Some visualization systems designed in this manner have come to untimely ends. Those likely to be successful are those that are based on widely accepted and standardized software tools that themselves have been ported to many platforms, such as OpenGL for graphics.

**Capture larger concepts.** This will ameliorate the time impediments of searching for, downloading, installing, and learning a new tool for each new concept. A visualization system is likely to be more widely used if it allows for visualizations of an entire "module" of related concepts, such as all of searching and sorting rather than just a single algorithm or two. Furthermore, the treatment of each visualized concept will have a similar "look and feel" that allows instructors and students to focus on learning concepts instead of learning how to use a tool. For example, systems like Alice [9] provide a resource around which a course can be designed and that can be used for most, if not all, of a course.

**Map to existing teaching and learning resources.** Extending the previous observation, providing a visualization package that corresponds to an existing resource (for example, a textbook) will make the package more appealing to users of that textbook. If done

well, the seamless integration of the book with the visualization system will also eliminate most of the time required to adapt and integrate a visualization system into a particular course. It has the drawback, of course, of being primarily useful to those who choose to use that textbook. Integrating the system with text materials can further improve the adaption of the system. For example, Brown et al. state that a large part of the success of the *BALSA* system was due to its tight integration with the teaching materials [**?**].

**Design for flexibility.** One can make a visualization tool so flexible that it can easily be adapted to many different presentation styles and course resources (for example, textbooks). This is a worthy goal but, of course, more difficult to achieve. For example, there are many different implementation nuances that affect how a particular algorithm (for example, a Quicksort algorithm) actually runs. Virtually every textbook provides a different version. Adapting a given visualization of an algorithm to the precise way in which the algorithm is presented in the textbook may be difficult or time-consuming. The discrepancy in content may lead to confusion on the part of students. Making adaptation easy due to system flexibility can therefore play a key role in a successful adoption of a visualization system.

**Provide comprehensive, integrated support.** To eliminate the frustrating time impediments of learning a visualization resource and teaching students to use it, a comprehensive support structure should be part of the tool. This support should include a very carefully designed GUI that is novice-friendly and easily learned. Documentation on the use of the tool as well as tutorials illustrating its use should also be part of the software. The entire support structure should be refined based on feedback from the user community, both learners and instructors.

**Develop a supporting Web site.** A carefully designed Web site for a visualization system can do much to address the time impediments that frustrate instructors. Choose a clever name for the visualization tool that is catchy, informative, and will be easily found in a Web search. A Web site that provides a place to download the visualization software should include many other things as well, such as sample lectures, exercises, and PDF documents for hard-copy instructions that can be used by students who are learning to use the system. Community forums and FAQs that allow users of the tool to interact with other instructors who are using the system also make adoption of the system more likely. A good Web site is, in fact, such an important aspect of this discussion that we elaborate on it in sections 2.3 and 2.4.

**Register the tool in repositories.** To help overcome the impediment of time to find visualizations on the Web, the systems should be registered in relevant repositories. Unfortunately, there is currently no single authoritative repository for registering visualization tools. However, there are various competing repositories or link collections that can be accessed when searching for visualization tools or content. These include the *Complete Collection of Algorithm Animations* [7], the forthcoming *Algorithm Animation Repository* [10], *CITIDEL* [8], a prototypical repository of visualization resources [42] as well as the "SIGCSE Education Links" on the ACM SIGCSE Web site [1].

**Publicize.** It is probably safe to say that most instructors do not actively seek visualization systems to use in their courses. This could be because they are satisfied with their way of teaching, they are unaware of visualizations in general, or they have tried visualizations in their courses before without success. It is mandatory that, in addition to simply registering their work in a repository, visualization tool designers publicize their work in venues such as the annual SIGCSE symposium (through papers or posters), in general educational journals such as *Inroads* and in area-specific journals (for example, if the tool visualizes aspects of the theory of computing, in relevant theory journals), and any other appropriate media.

The above advice is not meant to be comprehensive, but rather illustrative. Once the impediments are known and some examples of ways to surmount these impediments have been discussed, we are certain that visualization designers will become more adept at providing the community with systems that address the issues that have slowed the widespread use of the tools and thus promote the use of visualizations in a positive way throughout the curriculum.

## 2.3 Disseminating Visualization Tools

How a visualization tool designer disseminates a system plays an important role in how widely the system will be used. In this section, we present a suggested outline of a standard Web site for this purpose. The site should make it easy for Web surfers to find the tool, learn about it, download it, and install it. The site should further provide a mechanism for obtaining feedback from those who choose to download the system. Feedback is used to measure the level of satisfaction of those who use it. In what follows, the word "tool" may mean either software for designing visualizations (examples: ANIMAL, Alice, Matrix, ...), or a collection of pre-prepared visualizations (examples: QuickTime movies or Java applets).

**The portal.** Acknowledging principles of good Web page design (see, for example, [36, 37]), we recommend that the entry page, or *portal*, to the Web site be attractively designed and that it provide clear information describing:

- The name of the tool
- Author contact information
  - Names
  - E-mail addresses
  - Institutions
- A short, clear description of the tool that will let visitors know whether the system is of interest (so that they can decide whether to investigate further or abandon this particular search)
- Other pages that provide in-depth details for interested visitors, including:

- A *detailed description* of the tool
- *Documentation* on the use of the tool
- *Supporting materials* for the instructor
- *Evaluation instruments* and results of prior evaluations
- *Download* information

We elaborate further on the links listed above.

**The description page** The detailed description page should provide:

- A comprehensive description of the tool and its use
- The levels of targeted learners
- References to the algorithms or concepts being visualized so that instructors can determine whether the visualization integrates with their way of teaching
- Further links to any existing publications describing the tool and its use

**The documentation page.** The documentation page should provide:

- Documentation on how to use and install the tool
- A statement about whether the tool is still maintained
- A printable tutorial for students to use when learning to use the tool

**The support page.** The support page should provide, where possible and appropriate:

- Suggestions on the use of the tool
- Lecture support material (such as PowerPoint slides)
- Sample exams and quizzes
- A set of exercises for use with the tool

**The evaluation page.** The purpose of the evaluation page is twofold: (1) to provide feedback to the tool designer on the level of instructor and student satisfaction with regard to use of the tool, and (2) to provide visitors with results of earlier evaluations. As the tool matures, it may even be possible to include formal studies on the effects of the tool on student learning, but in this section we just provide suggestions for measuring instructor and student satisfaction with an eye towards making the tool more enticing.

Thus, what we suggest is that this page include links to online statistics-gathering instruments for:

- Obtaining feedback from instructors who download and use the system
- Obtaining feedback from students who use the system

This information is so vital to the ongoing success of the tool that we devote Section 2.4 to an elaboration of possible instruments for these items.

**The download page.** This is a crucial page. This page should not only supply an easy way to download the tool, but it should be designed to elicit information from those who download. The first part of this page should be a genuine plea to the person downloading the tool that asks for help with the project. It should be clearly explained that this is an academic (rather than commercial) project and that the continued success and improvement of the tool depends crucially on voluntary feedback from the user community. The download process should thus have a mandatory registration procedure that requests information about:

- E-mail and other contact information
- The background of the person downloading the software (instructor, student, or other)
- The purpose of the download (for use in a course, independent learning, simple inquisitiveness)
- The person's willingness to receive future e-mails about the use of the tool (along with a clear statement that contact information will not be used for other purposes)

The information on this page should point to the evaluation instruments (see the next section) to provide the person with a clear idea of the kinds of information that might be requested in the future. It is also always good to provide a free-response box to allow the person to provide additional comments as well.

## 2.4 Evaluation

In this section, we propose sample items to include in evaluation instruments intended to measure instructor and student satisfaction with the tool. The purpose is to provide feedback to the tool designer that will allow modification of the tool to improve instructor and student satisfaction. These instruments will be filled out by instructors and students after the tool has been used. Thus, the evaluator needs to maintain contact with instructors after they have first downloaded the system.

### *2.4.1 Evaluation instruments*

The evaluation instruments should be extremely easy to fill out so that as great a return as possible is obtained. They should be administered online and be automatically accumulated in a database belonging to the tool designer. Requests for written answers should be carefully thought out and not used excessively so as not to be too time-consuming for those filling out the form.

The evaluation instrument for instructors should include questions that obtain:

- The instructor name and contact information
- The content and level of the course in which the tool was used
- Course enrollment
- Assumed prerequisites

Scaled questions using the traditional Likert scale with values such as *strongly disagree, disagree, neutral, agree, strongly agree* include:

- The tool is easy to obtain
- The tool is easy to install
- The tool is easy for an instructor to use
- The tool is easy to show and teach to students
- The tool is easy for students to learn
- The tool works reliably
- The tool contributes to good learning outcomes

Multiple-choice questions which don't fit on a scale like the one above include:

- How did you learn about the tool (private communication, from a conference, from a Web site, from a Web search, in a book, ...)?
- How often did you use the tool in the course (one or two times, regularly for part of the course, regularly throughout the course, ...)?
- In what context did you use the tool (classroom presentation, closed lab exercises, open lab assignments, ...)?
- In this context, was the use of the tool (required, optional)?
- How did students interact with the tool (watched, answered questions, provided input or parameters, designed their own visualizations, gave a presentation in which the tool played a role, ...)?

A student evaluation instrument would ask different questions. It is hoped that the tool designer, upon contacting an instructor who downloaded the tool, could encourage the instructor to require students to fill out this separate student evaluation instrument on the tool Web site. Scaled questions for measuring student satisfaction include:

- I enjoy using the tool
- I feel I understand the concept better when using the tool
- The tool is easy to use
- The tool works reliably for me

Multiple-choice questions for students include:

- For any given assignment, how much time did you spend with the tool on average (about 5 minutes, about 10 minutes, about 15 minutes, about 30 minutes, about an hour, more than an hour, ...)?
- How many exercises or assignments did you do with this tool?
- How did you use the tool (watched in class, used in lab, used in university work area, used on my own computer, ...)

The evaluation instrument, whether for instructor or student, should always provide an open field for additional comments at the end. Further, since an instructor may use visualizations in different ways in different courses, the instructor should be encouraged to complete an evaluation for each course. We assume that instructors who were willing to fill out the instruments would not mind being contacted again if clarification of responses is needed.

# 3. EVALUATION OF LEARNER OUTCOMES

Ultimately, the application of visualization techniques in the teaching and learning process will become widespread only if instructors have concrete evidence that student performance improves and/or that student interest and motivation in the subject are enhanced when visualizations are used.

Since we cannot directly measure the learning process, the focus of this section is on measuring learning outcomes and how student attitudes are affected by the use of visualization techniques. We provide suggestions for evaluating student learning outcomes and attitudes when visualization tools are employed. We also offer guidance for the visualization tool designer, the visualization designer, and instructors who desire to study the effects of using visualization tools in their courses. Throughout, we attempt to describe experiments that support the selection of tools for everyday teaching situations (hereafter TS) as well as experiments that are designed for more formal education research purposes (hereafter RP).

## 3.1 Different forms of evaluation

*Summative evaluations* are those that occur after students' use of the tool is completed for the study in question. *Formative evaluations* occur during the study and are meant to determine whether project-related objectives are being met as the study progresses.

### *3.1.1 Formative evaluations*

Formative evaluations typically involve qualitative assessment. For more details about formative evaluations, see [17, 16]. This section discusses formative evaluations in general, but focuses on those formative evaluations we believe are particularly well-suited for visualization.

1. Student attention to a visualization

   By studying in depth how the student uses the specific visualization, we can determine if the student is using it in ways the instructor intended. Possible implementation ideas include observations (where the student is watched, either directly or through a one-way mirror, performing a specified task) and eye-tracking cameras (where it can be determined on which parts of the visualization tool the student is focusing).

2. Time-on-task

   This evaluation may be thought of as both formative and summative. The purpose is to keep track of how long the student spends working with the visualization tool in an assignment. The simplest approach is to have the tool record the time at startup and when it is shut down. A more detailed implementation involves the generation of a log by the visualization tool of all student interactions with the tool. While generating a log is more difficult to implement (and analyze), it does allow for a more detailed analysis of interaction of the student with the tool. This approach may be used in a formative manner. For example, a log would allow the instructor to see if the student is having difficulties using the tool. Such a formative evaluation allows the instructor to adjust lecture materials, or perhaps provide a modified tutorial on the visualization tool's use.

This is especially true in the case of virtual learning environments, where the learner may lack direct feedback. Here monitoring the overall student performance in a time-on-task sense is a bit more easily adopted. The feature was recently incorporated, for example, into the electronic text book illustrated in [27].

Software that allows an instructor to watch the screen of a student at a remote workstation may also be used when the instructor wants to focus on one particular student's interaction with the visualization tool.

3. Intermediate student feedback

   Anonymous surveys may be employed during the actual usage of the visualization tool in a class to get students' impressions of the tool. Students may be asked to solve a problem to gauge their comprehension, how much time they are spending using the tool, how much time they are spending on the course without involving the tool, their opinion of the materials, the lecture, the visualization tool itself, and so forth. One way to improve the outcome of the survey is to compare the survey data to actual facts known from the tool (from a log automatically generated by the visualization tool, as discussed in item 2 above). Thus, the impression expressed by the students can be "verified" by comparing their opinions with actual test results of the performance. These are not absolute values but merely comparable with each other, for example to determine the learner's opinion on the tools versus other resources used in the course.

   Another approach of interest in obtaining intermediate student feedback is the use of Classroom Assessment Techniques (CATs) [45]. While not directly related to the use of visualization tools in class per se, CATs appear to be a promising mechanism for obtaining ongoing student feedback, as well as helping to contextualize course content for the students.

4. Peer reviews of curricular materials

   Peer review is a widely accepted technique for examining the content, construct, and criterion validity of instructional materials [3]. This is particularly valuable when the instructors' materials are evaluated by the visualization tool creator, and by other visualization experts in the field, as well as by "expert teachers" in the specific content area where the visualization is being used. The instructor may receive valuable feedback ensuring that proposed use of the visualization tool in class is pedagogically sound.

5. Student interviews

   Interviewing a random subset of the students who have used the specific visualization tool and its associated materials can provide valuable feedback. By focusing on the students' experiences with the tool, the instructor can gain a detailed understanding of the students' comprehension, student attitudes towards the visualization tool and the subject being studied, students' comfort with using the tool, students' suggestion for tool/lecture improvements, and so forth. Interviewing may be done in an individual and/or a small group format.

### *3.1.2 Summative evaluations*

Summative evaluations summarize the effectiveness and/or results of the study after students have completed their use of the tool. Generally quantitative methods are used in summative evaluations. More details about summative evaluations may be found in [16]. Like the formative evaluation subsection, the focus here is on those summative evaluations most applicable to visualization.

1. Analysis of learner understanding using *mental models*

   Evaluation of student outcomes using mental models is an analysis technique used primarily in formal education research studies. Since we have no analysis method based on mental models for visualizations, we describe an analysis method that has been developed for mental models used by programmers. It would be important to try to find out what is good comprehension of algorithms (instead of programming in general). By knowing this we would have a solid basis for the evaluation of learning outcomes. One possibility for determining the quality of algorithm comprehension is to use similar study techniques as Pennington [38] did with programming.

   Pennington [38] studied comprehension strategies in programming. Good [18] continued Pennington's work and developed a classification for analyzing novice program comprehension. Her classification has been applied by Good and Brna [19] and Sajaniemi and Kuittinen [44]. These studies describe the method and the procedure for analyzing the mental models of novice programmers. Thus, they can be used as examples of how to use mental models for evaluating learning outcomes.

   Pennington had 40 expert programmers that were the top and bottom quartile comprehenders (20 each) from her previous study (see the details in [38]). The programmers had to make a modification to a program normally maintained by another programmer. After studying the program they were asked to write a summary of the program, respond to a set of comprehension questions and explain their answers. Then they were asked to make the modification and, after that, to summarize the program and respond to another set of comprehension questions. Answers to the questions as well as the program summaries were the basis for the measurement of programmers' comprehension which then could be classified as good or poor comprehension based on the subjects' quartile.

   The basic idea of Good's analysis method is that comprehension can be studied using program summaries which are analyzed using two different methods: *information type analysis* and *object description analysis*. According to Good and Brna [19], the information types classification is used to code summary statements on the basis of the information types they contain. The types include eleven categories: function, actions, operations, state-high, state-low, data, control, elaborate, meta, unclear, and incomplete. For a definition of these terms, refer to [19]. The object classification looks at the way in which objects are described. This classification has seven categories: pro-

gram only, program, program – real-world, program – domain, domain, indirect reference, and unclear.

The information types described above help to divide information into *low level information* and *high level information*. High level types are function, action, state-high, and data, while low level types include operation, state-low, and control. Elaborate, meta, and unclear cannot be classified as either high or low level types.

There are two ways to use these analytical methods: students can be asked to write program summaries or they can be asked to answer very carefully designed questions. *Program summaries* are difficult to analyze and therefore these kinds of tasks may not be suitable in TS. For RP, the analysis method is very useful for finding what kind of effect different teaching methods may have on students' mental models. *Asking questions* is more suitable for TS because the questions can be designed to make the analysis of the answers easy.

Next we will discuss the design of the questions. Both high and low levels of information types should be used when asking questions, although the high level questions measure deeper understanding and therefore should be more valuable. It is possible that a student whose comprehension is on a high level cannot correctly answer the questions on a low level. This is not unusual and not necessarily bad, since high level comprehension replaces low level comprehension when learning is deep enough [38].

Object descriptions enable us to ask three kinds of questions: questions about the *program code* itself, questions about the *domain* of the program, and *cross-reference* questions in which the student needs to combine both the program code and the domain. Those students who can answer the cross-reference questions have a deep comprehension of the program while the others have only a surface level comprehension (that is, they probably recognize the code and/or the domain but may not understand the connection between them). From the evaluation point of view the best situation is when a student is able to answer all three kinds of questions. The next best is that a student can answer cross-reference questions (deep level) even though program or domain questions are not answered.

2. Analysis of learner understanding using levels in *Bloom's taxonomy*

   Bloom's taxonomy [5] is another common tool used in formal education research studies. The taxonomy offers a way to systematically design test tasks for analyzing learner understanding. Six comprehension levels are provided, each of which characterizes an aspect of the learning process. The idea is to reduce the qualitative improvement in learning to discrete levels of comprehension and moreover, to measurable scores. The six comprehension levels are *knowledge*, *comprehension*, *application*, *analysis*, *synthesis*, and *evaluation*. The experiment can be designed to measure the learner outcome in each of these levels starting from the knowledge and ending up with evaluation.

   The 2002 Working Group report [35] provides more detailed examples of the different Bloom levels. The context used in that report is a course in data structures and algorithms.

3. Pre- and post-content tests

   To help determine content mastery where a visualization tool is used, pre- and post-tests are particularly effective. Typically, the same test is used for both tests. The purpose of the pre-test is to determine the level of prior knowledge the student has. It is of utmost importance to ask the "right" questions. If in the case of a study designed for TS it is not possible to use identical exams for the pre- and post-tests, the questions on the pre-test may be a proper subset of those on the post-test.

4. Attitude survey

   Attitude surveys are generally given before and after students' use of a visualization tool. They are used to determine changes in students' attitudes, confidence, motivation, and so forth as a result of their experience with the visualization. The two most widely accepted surveys for determining student attitudes towards computers and computer science, Francis [15], and Loyd and Gressard [33], are somewhat dated. These survey instruments are most appropriate for use in introductory classes. There is certainly a need for the development of a newer, more relevant, survey instrument to be tested and validated. Instructors of upper-level courses will need to create their own survey instruments. The difficulty is that the survey instrument itself will not be experimentally validated. A good attitude survey provides valuable feedback about the students' impression of the visualization tool, and is typically administered as a pre- and post-test.

5. Retention and attraction

   Particularly for visualization tools used in lower-level courses, it is interesting to monitor the change in student retention in the computer science major and minor. For those tools that impact courses for non-majors, it is important to examine whether or not the use of the visualization helps to encourage students to become computer science majors or minors. Retention and attraction statistics are often combined with student attitude surveys to help gauge student reaction to their experiences with the visualization tool.

6. Grades

   Student grades are a measure of student success in a course. While they are generally not as useful as pre- and post-tests to gauge student content mastery, grades are easy to collect and analyze.

7. Time-on-task

   While described in the formative subsection, time-on-task may also be used as a summative measure. As an example, consider the following. A student's use of a visualization tool as part of an assignment can be timed. For example, if a student does an assignment in an environment where no clock is available, the student can be asked how much time elapsed while working on

the assignment. If the student thinks that a smaller amount of time passed than actually did, this indicates that student's interest in using the visualization tool. See [21] or [4] for more details.

#### 3.1.3 Which of these evaluation methods should be used?

This is, in general, a difficult question to answer. The first question to answer is whether the particular evaluation of a study is primarily for teaching or research purposes. A research study typically requires significantly more assessment than a study primarily interested in improving a teaching situation. Additionally, the specific visualization tool and its associated visualizations are key determinants of the specific evaluation strategies to be used. For several sample studies, see the case studies described in Section 4 of this paper.

### 3.2 Covariant factors

If we are going to set up a study where we compare different tools and assignments, we should recognize that learning style may affect the results. Consider as a trivial example a study with only two students, A and B, in which the results of two assignments given to both are compared. Suppose that in the first assignment, A gets grade of 1 out of 5 and B gets grade of 5 out of 5. In the second assignment, suppose A gets 5 out of 5 and B gets 1 out of 5. Obviously, the average results are the same, and we could claim that we found no difference in results for the assignments. Suppose, however, that A is a highly verbal and B a highly visual learner and that the assignments were visual and verbal, respectively. By considering the learning style as a covariant factor in the study, we would observe a major change in students' performance.

Several covariant factors are listed in last year's working group report [35]. Here we discuss in detail some factors as they pertain to evaluation.

1. Learning styles

   Students have different styles of taking in and processing information. Thus, some students are more comfortable with reading text while others prefer grasping information from figures and visualizations. Other students like to process information actively through experimentation and observations while others prefer processing information in their minds through intuition and reflection.

   It is important that we as teachers recognize these differences, since they very much affect how our students feel about studying and how well they succeed at various activities and tasks we design. A conflict often occurring in undergraduate education is that teachers explain the topic deductively, first concentrating on principles and theories, and then proceed to explain how these theories and principles are applied to analyzing phenomena and solving practical problems [12]. Most undergraduate students, however, seem to process information inductively, learning details first and then proceeding to understand principles. As an example, in introductory programming courses, most students have difficulties with syntax and semantic details and find it hard to understand the relevant principles in the background, even if they are explicitly explained.

   Differences in learning styles can be expressed with *learning models*, which are general frameworks for characterizing different aspects of learners' activities. Examples of such models include Kolb's experiential learning [23] and the Felder-Silverman learning model [13]. In the following, we discuss the latter model in some more detail since it seems particularly suited to science education.

   Felder and Silverman identify four dimensions of student behaviors, each of which has two extremes.

   *Sensory vs. Intuitive* – what type of information does the student preferentially perceive. *Sensing learners* prefer collecting information by observation. They often like facts, working with details, and memorizing data. They also prefer a practical approach, experimentation and solving problem using standard methods. *Intuitive learners* like conceptual thinking, such as theories and principles, and grasping new concepts.

   *Visual vs. Verbal* – how is the sensory information most effectively perceived. *Visual learners* better adopt visual information and *verbal learners* prefer information in written or spoken form.

   *Active vs. Reflective* – how does the student prefer to process information. *Active learners* learn by trying things out and prefer interaction. *Reflective learners* prefer examining and manipulating information introspectively.

   *Sequential vs. Global* – how does the student progress towards understanding. *Sequential learners* like to proceed linearly with small steps using the given instructions. *Global learners* like to get a holistic view of acquired knowledge.

   Initially, Felder and Silverman also had a dimension *inductive* vs. *deductive* learners, but they decided to drop it since the model should not promote the existing conflict between deductive teachers and inductive learners [12].

   Obviously, all these axes are continuous, that is, each student lies somewhere between the extremes. Their orientation can be evaluated by exposing the student to a simple questionnaire, such as presented in [22]. However, we note that orientation may change over time and may depend on the context in which students are working.

   The general goal is that our students should extend their skills of adopting and processing information within all four axes. To accomplish this goal, the teacher may push the change by setting up activities that train the weaker side of student's behavior.

   Next we present a few examples of assignments that demonstrate how to request different types of activities in this context. The examples are presented in more detail in [30].

   Consider an assignment that requests the user to trace how various binary tree traversal algorithms work, that is, list the order in which the nodes are traversed when a specific traversal algorithm is applied. A visual form of this exercise could include a picture of a binary tree, allowing the student to click the nodes on the screen in the appropriate order. Alternatively, the assignment

could be given in verbal order by giving the tree as an adjacency list and asking the student to write the keys in the nodes as a list according to their traversal order.

A sensing learner could solve the exercise by applying a simple mnemonic such that a line is drawn around the tree starting from the left side of the root and listing each key when the corresponding node is passed from the left (preorder), below (inorder) or right (postorder). An intuitive form of the exercise is that we give the tree and the pseudo code of the traversal algorithm and ask the student to list the nodes in the order the given algorithm visits them.

2. Student background surveys

   Introductory student surveys are useful for obtaining background data (such as previous classes taken, previous experience, mathematics background, and general entrance exam scores) as well as information such as why they registered for this section of the course. This data is particularly useful in helping to determine whether the student's background is a factor in the student's success either in the particular course, or with the particular visualization tool.

3. Time-on-task

   Time-on-task has already been discussed in the formative evaluation subsection. We have also included it as a covariant factor because it might be expected that increased time spent using visualization may be a factor in performance.

## 3.3 Testing with human subjects

We wish to issue a note of warning for those instructors planning to run studies in their classes. In many countries, approval of "human subjects review boards" is required for all studies involving students. Written consent from students involved in studies will also often be required. The case studies in Section 4 provide examples of successful applications that others have used in broaching this sensitive area within their institutions. Such applications may serve as useful templates for instructors to use for gaining "human subjects" approval at their own schools.

# 4. CASE STUDIES

Many members of the working group have been and are involved in studies that demonstrate the efficacy of the guidance given in Sections 2 and 3. These are collected online and provide concrete realizations of the principles that are described in this report. The letter legends used indicate whether the study is completed and published (CP), completed but not yet published (CNP), in progress (IP), oriented toward measuring instructor satisfaction (IS), oriented toward measuring student outcomes with respect to everyday teaching situations (SO-TS) or more formal education research (SO-RP).

Presently, this list includes the following:

- Stephen Cooper and Wanda Dann – This study examines the use of program visualization for introducing objects and their behaviors using Alice [11], a 3D program visualization environment. Statistical data is collected to show evidence of student performance and retention in CS1 for test and control groups. Early summaries may be found in [9]. CNP, SO-TS, SO-RP.

- Rudolf Fleischer – This study will measure the effectiveness of visualizations in the context of a second-year course on the theory of computation (finite automata, context-free grammars, Turing machines) [14]. The study will be done in Spring 2004 at the Hong Kong University of Science and Technology. IP, SO-TS.

- Boris Koldehofe – By using the framework presented in this working group report and considering the results of a previous study [25], a new study of instructor's satisfaction with LYDIAN [24], a simulation-visualization environment for learning and teaching distributed algorithms, is planned. The purpose is to evaluate which features of LYDIAN contribute to instructor satisfaction and the teachers' performance when integrating LYDIAN in their course. The study also tries to determine the factors which may prevent instructors from actually using LYDIAN after they download the tool. IP, IS.

- Ari Korhonen and Lauri Malmi – Intervention study with automatic assessment – the paper [28] presents the results of the large scale (N=550 students) intervention study carried out in a virtual learning environment based on the TRAKLA system [26]. The system provides individually tailored exercises that the learner solves on the Web using an algorithm simulation [29]. The learner receives immediate feedback around the clock on his or her performance. The system has features that traditional instruction cannot provide. Thus, the study was pursued to research the quality, advantages and limitations of this novel approach. One of the conclusions was that the learning environment was as good as if they were solving the same exercises in class room with human tutors giving the feedback. CP, SO-RP.

- Ari Korhonen and Lauri Malmi – This study [34] presents some experiences and observations made during 10 years of using the TRAKLA system [26]. It is inevitable that learners perform better when they are allowed to resubmit their work. However, this is not the whole story; it is also important to organize the course so that the skills and challenges of the learner coincide. Moreover, the grading policy seems to have a major impact on performance. The study summarizes the results and points out changes in learner performance under the several changes the course has gone through during the time period. CP, SO-TS.

- Ari Korhonen and Lauri Malmi – This study focuses on the effect of different learner engagement levels on learning. The ITiCSE 2002 working group report [35] on visualization prepared a taxonomy of engagement in order to identify and differentiate various types of actions the learners may perform while using visualizations. The plan for this study will appear at the Web site of the Computer Science Education Research Group (`http://www.cs.hut.fi/Research/COMPSER/`) at Helsinki University of Technology. IP, SO-TS.

- Marja Kuittinen and Jorma Sajaniemi – This study [31, 44] evaluates the use of variable roles and role-based animation in teaching introductory programming.

The role concept captures tacit expert knowledge in a form that can be taught to novices. Details about the project and evaluation studies can be found at [47]. CP, SO-RP.

- Charles Leska – This study examines the impact on student performance of using visualization on a unit in a computer literacy course. The unit focuses on the binary representation of data and the interplay of the processor components during the fetch-execution cycle. Details about the study can be found at [48]. IP, SO-TS.

- Scott Grissom, Myles McNally, and Tom Naps – This study compares the effect of three different learner engagement levels with visualization in a course module on introductory sorting algorithms. Results of the study have been published in [20]. CP, SO-TS.

- Jarmo Rantakokko – This study evaluates the effects of using algorithm visualization in parallel computing. The study focuses on student learning outcomes. Details about the project and this particular study can be found at [49]. IP, SO-RP.

The working group's URL [46] contains updated information on these studies and others that have been added since the preparation of this report.

# 5. CONCLUSION

This report has been based on the premise that visualization can significantly impact CS education only if two goals are met – widespread use and positive student outcomes. Visualization designers have typically not researched the factors that would influence instructor satisfaction, and Section 2 of this report has offered guidance for beginning this process. Section 3 has provided an overview of student outcome measurement techniques. This overview includes approaches that can be used for informal everyday teaching situations and for more formal education research studies. We hope this will be of help to both visualization tool developers and visualization designers who may not be specialists in doing such evaluation but who nonetheless are required to validate their product by demonstrating its effectiveness.

# 6. REFERENCES


[1] ACM Special Interest Group on Computer Science Education. SIGCSE Education Links. Available online at `http://www.sigcse.org/topics`, 2003.

[2] Akingbade, A., Finley, T., Jackson, D., Patel, P., and Rodger, S. H. JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses. In *Proceedings of the* $34^{th}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada* (2003), ACM Press, New York, pp. 162–166.

[3] American Educational Research Association, American Psychological Association, and National Council on Measurement in Education. *Standards for Educational and Psychological Testing*. American Educational Research Association, Washington, D.C., USA, 1999.

[4] Bederson, B. Interfaces for Staying in the Flow. Keynote at IEEE Human-Centric Computing Languages and Environments. Available online at `http://www.cs.umd.edu/~bederson/talks/HCCKeynote-Sept2002.ppt` (seen July 14, 2003).

[5] Bloom, B. S., and Krathwohl, D. R. *Taxonomy of Educational Objectives; the Classification of Educational Goals, Handbook I: Cognitive Domain*. Addison-Wesley, 1956.

[6] Boroni, C. M., Goosey, F. W., Grinder, M. T., and Ross, R. J. Engaging Students with Active Learning Resources: Hypertextbooks for the Web. In *Proceedings of the* $32^{nd}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2001), Charlotte, North Carolina* (2001), ACM Press, New York, pp. 65–69.

[7] Brummund, P. The Complete Collection of Algorithm Animations, 1998. Available online at `http://cs.hope.edu/~alganim/ccaa/` (see July 14, 2003).

[8] CITIDEL - Computing and Information Technology Interactive Digital Educational Library. Available online at `www.citidel.org` (seen July 14, 2003).

[9] Cooper, S., Dann, W., and Pausch, R. Introduction to OO: Teaching Objects-First in Introductory Computer Science. In *Proceedings of the* $34^{th}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada* (2003), ACM Press, New York, pp. 191–195.

[10] Crescenzi, P., Faltin, N., Fleischer, R., Hundhausen, C., Näher, S., Rößling, G., Stasko, J., and Sutinen, E. The Algorithm Animation Repository. *Proceedings of the Second International Program Visualization Workshop, HornstrupCentret, Denmark* (2002), 14–16. Available online at `http://www.daimi.au.dk/PB/567/PB-567.pdf`.

[11] Dann, W., Cooper, S., and Pausch, R. Making the Connection: Programming With Animated Small World. In *Proceedings of the* $5^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000), Helsinki, Finland* (2000), ACM Press, New York, pp. 41–44.

[12] Felder, R. M. Author's preface – June 2002. `http://www2.ncsu.edu/unity/lockers/users/f/felder/public/Papers/LS-1988%.pdf` (seen July 14, 2003), 2002.

[13] Felder, R. M., and Silverman, L. K. Learning Styles and Teaching Styles in Engineering Education. *Engineering Education 78*, 7 (1988), 674–681.

[14] Fleischer, R. COMP 272: Theory of Computation — A proposed study on the learning effectiveness of visualizations, 2003. Manuscript.

[15] Francis, L. Attitude towards Computers Scale. *Computers in Education 20*, 3 (1993), 251–255.

[16] Frechtling, J. *The 2002 User Friendly Handbook for Project Evaluation*. National Science Foundation, 2002.

[17] Frechtling, J., and Sharp, L. *User-Friendly Handbook for Project Evaluation*. National Science Foundation, 1997.

[18] Good, J. *Programming Paradigms, Information Types and Graphical Representations: Empirical Investigations of Novice Program Comprehension*.

PhD thesis, University of Edinburgh, 1999.

[19] Good, J., and Brna, P. Toward Authentic Measures of Program Comprehension. In *Proceedings of the Fifteenth Annual Workshop of the Psychology of Programming Interest Group (PPIG 2003)* (2003), pp. 29–49.

[20] Grissom, S., McNally, M., and Naps, T. L. Algorithm Visualization in Computer Science Education: Comparing Levels of Student Engagement. In *Proceedings of the First ACM Symposium on Software Visualization, San Diego, California* (2003), ACM Press, New York, pp. 87–94.

[21] Jenkins, J., and Visser, G. Making Learning Fun. Available online at `http://www.imaginal.nl/learningFun.htm` (seen July 14, 2003).

[22] Keirsey, D. M. Keirsey Temperament and Character Web Site. Available online at WWW: `http://www.keirsey.com` (seen July 14, 2003), 2002.

[23] Kolb, D. A., Ed. *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall Inc, New Jersey, USA, 1984.

[24] Koldehofe, B., Papatriantafilou, M., and Tsigas, P. LYDIAN, An Extensible Educational Animation Environment for Distributed Algorithms. In *Proceedings of the* $4^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000)* (July 2000), ACM Press, New York, p. 189.

[25] Koldehofe, B., Papatriantafilou, M., and Tsigas, P. Integrating a Simulation-Visualization Environment in a Basic Distributed Systems Course: A Case Study Using LYDIAN. In *Proceedings of the* $8^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2003)* (June 2002), p. 226.

[26] Korhonen, A., and Malmi, L. Algorithm Simulation with Automatic Assessment. In *Proceedings of the* $5^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000)* (Helsinki, Finland, 2000), ACM, pp. 160–163.

[27] Korhonen, A., Malmi, L., Mård, P., Salonen, H., and Silvasti, P. Electronic course material on Data structures and Algorithms. In *Proceedings of the Second Annual Finnish / Baltic Sea Conference on Computer Science Education* (October 2002), pp. 16–20.

[28] Korhonen, A., Malmi, L., Myllyselkä, P., and Scheinin, P. Does it Make a Difference if Students Exercise on the Web or in the Classroom? In *Proceedings of the* $7^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2002)* (Århus, Denmark, 2002), ACM Press, New York, pp. 121–124.

[29] Korhonen, A., Malmi, L., Nikander, J., and Silvasti, P. Algorithm Simulation – A Novel Way to Specify Algorithm Animations. In *Proceedings of the Second International Program Visualization Workshop, HornstrupCentret, Denmark* (2002), pp. 28–36. Available online at `http://www.daimi.au.dk/PB/567/PB-567.pdf`.

[30] Korhonen, A., Malmi, L., Nikander, J., and Tenhunen, P. Interaction and Feedback in Automatically Assessed Algorithm Simulation Exercises. *Accepted for publication in Journal of Information Technology Education* (2003).

[31] Kuittinen, M., and Sajaniemi, J. First Results of An Experiment on Using Roles of Variables in Teaching. In *EASE & PPIG 2003, Papers of the Joint Conference at Keele University* (2003), pp. 347–357.

[32] Levy, R. B.-B., Ben-Ari, M., and Uronen, P. A. An Extended Experiment with Jeliot 2000. In *Proceedings of the First International Program Visualization Workshop, Porvoo, Finland* (July 2001), University of Joensuu Press, Finland, pp. 131–140.

[33] Loyd, B. H., and Gressard, C. P. Computer Attitude Scale. *Journal of Computing Research 15*, 3 (1996), 241–259.

[34] Malmi, L., Korhonen, A., and Saikkonen, R. Experiences in Automatic Assessment on Mass Courses and Issues for Designing Virtual Courses. In *Proceedings of the* $7^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2002)* (Århus, Denmark, 2002), ACM Press, New York, pp. 55–59.

[35] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. Á. Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin 35*, 2 (June 2003), 131–152.

[36] Nielsen, J. *Designing Web Usability. The Practice of Simplicity*. New Riders Publishing, 1999.

[37] Norman, D. "Top Ten Mistakes" Revisited Three Years Later. Available online at `http://www.useit.com/alertbox/990502.html` (seen July 14, 2003), May 1999.

[38] Pennington, N. Comprehension Strategies in Programming. In *Empirical Studies of Programmers: Second Workshop* (1987), G. M. Olson, S. Sheppard, and E. Soloway, Eds., Ablex Publishing Company, pp. 100–113.

[39] Price, B., Baecker, R., and Small, I. An Introduction to Software Visualization. In *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1998, ch. 1, pp. 3–27.

[40] Ross, R. J. Hypertextbooks for the Web. In *Proceedings of the First International Program Visualization Workshop, Porvoo, Finland* (July 2001), University of Joensuu Press, Finland, pp. 221–233.

[41] Ross, R. J., and Grinder, M. T. Hypertextbooks: Animated, Active Learning, Comprehensive Teaching and Learning Resources for the Web. In *Software Visualization* (2002), S. Diehl, Ed., no. 2269 in Lecture Notes in Computer Science, Springer, pp. 269–284.

[42] Rößling, G. Algorithm Animation Repository. Available online at `http://www.animal.ahrgr.de/` (seen July 14, 2003), 2001.

[43] Rößling, G., and Freisleben, B. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing 13*, 2 (2002), 341–354.

[44] Sajaniemi, J., and Kuittinen, M. An Experiment on Using Roles of Variables in Teaching Introductory Programming. *Submitted to Empirical Software Engineering* (2003).

[45] Schwarm, S., and VanDeGrift, T. Making Connections: Using Classroom Assessment to Elicit Students' Prior Knowledge and Construction of Concepts. In *Proceedings of the* $8^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2003)* (2003), ACM Press, New York, pp. 65–69.

[46] Visualization Working Group Home Page. Available online at `http://www.algoanim.net`, 2003.

[47] Kuittinen, M., and Sajaniemi, J. Home Page of the Study on Variable Roles and Role-Based Animation. Available online at `http://cs.joensuu.fi/~saja/var_roles/`, 2003.

[48] Leska, C. Homepage of the Visualization Study in Computer Literacy. Available online at `http://faculty.rmc.edu/cleska/public_html/algvizstudy.htm`, 2003.

[49] Rantakokko, J. Homepage of the Visualization Study in Parallel Computing. Available online at `http://user.it.uu.se/~jarmo/hgur.html`, 2003.