# Exploring the Role of Visualization and Engagement in Computer Science Education

**Report of the Working Group on "Improving the Educational Impact of Algorithm Visualization"**


**Thomas L. Naps (co-chair)**
**U Wisconsin Oshkosh, USA**
**naps@uwosh.edu**

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

**Myles McNally**
**Alma College, USA**
**mcnally@alma.edu**

**Guido Rößling (co-chair)**
**Darmstadt U Techn., Germany**
**roessling@acm.org**

**Chris Hundhausen**
**U of Hawaii, USA**
**hundhaus@hawaii.edu**

**Susan Rodger**
**Duke University, USA**
**rodger@cs.duke.edu**

**Vicki Almstrum**
**U Texas Austin, USA**
**almstrum@acm.org**

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

**J. Ángel Velázquez-Iturbide**
**U Rey Juan Carlos, Spain**
**a.velazquez@escet.urjc.es**

**Wanda Dann**
**Ithaca College, USA**
**wpdann@ithaca.edu**

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



## Abstract

Visualization technology can be used to graphically illustrate various concepts in computer science. We argue that such technology, no matter how well it is designed, is of little educational value unless it engages learners in an active learning activity. Drawing on a review of experimental studies of visualization effectiveness, we motivate this position against the backdrop of current attitudes and best practices with respect to visualization use. We suggest a new taxonomy of learner engagement with visualization technology. Grounded in Bloom's well-recognized taxonomy of understanding, we suggest metrics for assessing the learning outcomes to which such engagement may lead. Based on these taxonomies of engagement and effectiveness metrics, we present a framework for experimental studies of visualization effectiveness. Interested computer science educators are invited to collaborate with us by carrying out studies within this framework.


## 1 Introduction

This report is the culmination of efforts by the Working Group on Improving the Educational Impact of Algorithm Visualization. The group, which was convened by Tom Naps and Guido Rößling, began work during spring 2002. Using a groupware tool and a listserv mailing list, the group discussed a number of issues, prepared and conducted an online survey, and developed a rough draft of the report before meeting in person during the ITiCSE conference in Århus, Denmark. Throughout the remainder of this report, "we" refers to the Working Group, which was composed of the individuals listed at the beginning of the report. We also had three remote members, who are acknowledged in a section at the end of this report.

The impetus for visualization in computing comes from the inherent abstractness of the basic building blocks of the field. Intuition suggests that, by making these building blocks more concrete, graphical representations would help one to better understand how they work. Visualization software emerged in the late 1980's for the purpose of creating and interactively exploring graphical representations of computer science concepts [8, 54].

Our recent surveys of computer science educators suggest a widespread belief that visualization technology positively impacts learning. However, experimental studies designed to substantiate the educational effectiveness of such visualization technology simply do not bear this out [28]. On top of this, a major deterrent to adopting visualization that emerged from our pre-conference survey is the time and effort required for instructors to integrate the technology into their curricula. These findings point to two key obstacles to visualization technology's widespread adoption:

- From the learner's perspective, the visualization technology may not be educationally beneficial.

- From the instructor's perspective, the visualization technology may simply incur too much overhead to make it worthwhile.

Given the belief of computer science educators that visualization technology, under the right conditions, can greatly benefit learners and instructors alike, what can be done to overcome these obstacles? With respect to educational effectiveness, it makes sense to look more closely at past experimental studies of effectiveness. Indeed, closer inspection reveals an important trend in those studies: that learners who are actively engaged with the visualization technology have consistently outperformed learners who passively view visualizations [28]. For example, visualization technology has been successfully used to actively engage learners in such activities as:

- Constructing their own input data sets [40, Chapter 9],
- Making predictions regarding future visualization states [11],
- Programming the target algorithm [31],
- Answering strategic questions about the visualization [23, 44],
- Constructing their own visualizations [26].

Given this trend, it makes sense to design further experimental studies that more closely examine the educational benefits of various forms of active engagement. In this report, we present a framework for this type of empirical study. The framework is flexible enough to allow such studies to take place both in controlled situations and in classrooms. Our thesis is that *visualization technology, no matter how well it is designed, is of little educational value unless it engages learners in an active learning activity*. If this is true, then the key question to consider is what, if any, forms of active engagement with visualization technology can have a positive impact on how much a learner learns.

The second major obstacle to visualization technology identified in our survey, instructor overhead, is not directly addressed in this report. However, if experimental results show that some forms of active engagement with visualization lead to very positive educational outcomes, we are hopeful that a variety of educators will begin the development of instructional materials that take advantage of those forms of engagement.

Drawing on a review of experimental studies of visualization effectiveness and well-known best practices for algorithm visualization, Section 2 motivates our thesis against the backdrop of current attitudes and best practices with respect to algorithm visualization use. Section 3 presents a taxonomy of learner engagement with visualization technology, while Section 4 identifies metrics for assessing the learning outcomes to which such engagement may lead. Based on our taxonomy of engagement and the effectiveness metrics we propose, we present a framework for empirical studies of algorithm visualization effectiveness in Section 5. Section 6 concludes by discussing the future of this framework: studies we are planning, and opportunities for interested computer science educators to collaborate with us.

## 2 Background

Interactive visualization has been employed in computer science education since the 1980s (see, for example, [8, 10]). During that time, a set of "best practices" has evolved through instructors' experiences with the technology. We summarize the most important of these in Section 2.1. To determine current practice and instructors' attitudes toward the efficacy and impact of these practices, we recently conducted a survey of computing educators; we summarize the key results in Section 2.2. In Section 2.3, we summarize the widely mixed results of past experimental studies of visualization effectiveness and motivate the need for a new set of experiments.

### 2.1 Overview of Best Practices

Pedagogical visualization draws on many related disciplines, including typography, psychology, and algorithms. This makes it difficult to summarize the lessons learned, although some general recommendations about typography and layout apply. Khuri [35] summarizes recommendations on display layout, use of color and sound, and interactivity issues. While there is no agreed-upon standard for the "commandments of algorithm animation" [18], the following eleven points are commonly accepted suggestions drawn from experience:

1. *Provide resources that help learners interpret the graphical representation.* As concrete representations, visualizations may assist learners in understanding algorithms. However, visualizations may also be difficult to interpret; learners may find it difficult to map a visualization to the underlying algorithm it is designed to represent. The meaning of the graphical representations and their relation to program elements can be clarified for learners in one of two ways: explain the relationship by embedding the representations in the system using text or narration, or reinforce the relationship by allocating instructional time to the topic during the course.
2. *Adapt to the knowledge level of the user.* Novice learners can become quickly overwhelmed by too many details or windows, and they usually prefer to test an animation with predefined input data. In contrast, advanced learners may benefit from additional facilities for controlling complexity and for navigation, or from the capability to

invent input data to more fully exercise algorithms. In addition, novices may more easily understand the structure of animations that are based on well-known metaphors, for example, comic strips [4], theater performances [21], electronic books [9] or slide presentations [43]. In contrast, advanced learners may benefit from facilities for large data sets and multiple views, such as those provided by systems like BALSA [8].

3. *Provide multiple views.* An algorithm can be watched in many different ways, for example, control flow in source code or state of data structures. Providing the learner with multiple views can facilitate a better understanding of the algorithm. Windows displaying different views should be coordinated to show consistent information. In particular, it is very useful to provide a program animation view (where code is shown and highlighted as the program executes) simultaneously with more abstract algorithm animation views. In this way, the learner can relate algorithm actions to program code. An alternative approach is providing pseudo-code instead of raw code [57]. If pseudo-code nodes are enhanced with expand/contract facilities (i.e. simulating stepwise refinement), animations should be coordinated accordingly to ensure an adequate level of granularity. Finally, it can be advantageous from an educational point of view to offer different views sequentially. For instance, the HalVis system [22, 23] was designed to show animation in three steps: a familiar metaphor that could help learners understand the problem, a detailed animation with synchronous multiple views and explanations, and a large example of an application.

4. *Include performance information.* Efficiency analysis is an important part of algorithmic understanding. Thus, including data collected about the algorithm execution can enhance understanding of the algorithm's efficiency. Another way of reinforcing performance information is by animating several algorithms simultaneously, as in the Sorting Out Sorting videotape [2]. Different rates for solving the same problem are visually deduced by the user.

5. *Include execution history.* After several steps in the algorithm animation, it is common for the learner to forget previous steps, to have misunderstood some previous step in the algorithm, or simply to want to have a global view of the history. Making historical information available to the learner can help overcome these problems. History can be explicitly provided or can be implicitly integrated into some of the algorithm views. In JFLAP [29] when stepping through a nondeterministic example, one can select any configuration and see the history of the path from the start state to the chosen configuration.

6. *Support flexible execution control.* Flexible control of the visualization should be possible, including the ability to execute the visualization both forwards and backwards (see, for example, [6, 53, 56]). A simple but effective user interface for visualization control mirrors a video player, with buttons for the following functions: stop, pause, one step forward, continuous advance, advance to the end, one step backwards and backtrack to the beginning [43].

7. *Support learner-built visualizations.* Stasko [55] advocates that learners build their own visualizations. Such construction enables learners to gain insights into what is important about an algorithm under study. At the same time, it creates for learners a greater sense of responsibility through the construction of their own artifacts [24].

8. *Support custom input data sets.* Allowing learners to specify their own input data sets (for example, [8, 38]) engages them more actively in the visualization process. It allows the learner to explore the animation freely in order to discover how the algorithm executes on a range of data.

9. *Support dynamic questions.* To encourage learners to reflect on a visualization, visualization systems can use a "pop quiz" approach by periodically presenting short questions requiring a response from the learner [22, 44]. It is often useful to provide two kinds of questions. Some questions can pop up in random order, but in an appropriate context. Such questions focus the learner's attention on specific issues and promote self-evaluation as a means of improving comprehension. Other questions may be placed at critical points beyond which learners cannot proceed until they correctly answer the questions.

10. *Support dynamic feedback.* Learners should be provided with dynamic feedback on their activities within a visualization system. For example, Korhonen and Malmi [37] describe a visualization system that presents learners with graphical representations of algorithms and requires the learners to manipulate these representations in order to simulate the algorithm. The system then provides learners with automatic, dynamic feedback about the correctness of such simulations.

11. *Complement visualizations with explanations.* Educational research rooted in dual-coding theory suggests that visualizations may be better understood if they are accompanied by explanations [41]. Such integration can be made in a number of different ways, such as writing an accompanying explanation in a coordinated graphical window or providing a coordinated audio track for the visualization. A more traditional approach to explanations, based on paper books, is also possible. In particular, Bazik et al. [3] emphasize the need for animations to be tightly integrated with textbooks if such animations are to be integrated naturally into a course.

In considering the above recommendations, an educator must weigh carefully how to adapt and apply them, since there is no single visualization system or activity that is best for all learners. In fact, the design of an animation system

and its animations should be as carefully planned as any other design activity (see for example, [34]).

### 2.2 Survey of Current Practice

In this section, we describe the process and results from three surveys, two of which were designed and carried out by our Working Group. We first describe the design and content of all three surveys. Next, we profile the respondents and describe their teaching contexts. We go on to explore the responses related to visualization and its use in teaching. We conclude by discussing our impressions from the survey results and implications for further work.

#### 2.2.1. Description of the surveys

As the Working Group began to collaborate a few weeks before the ITiCSE 2002 conference in Århus, we considered a number of resources. One item that was circulated was a summary of a pencil-and-paper survey conducted by Scott Grissom during the ITiCSE 2000 conference in Helsinki, Finland [19]. The results from Grissom's survey motivated us to develop a new, more detailed on-line survey that we conducted prior to the conference. Using the items from Grissom's survey as a starting point, we successively refined the items to provide a basis that would guide the working group in carrying out its agenda.

The working group's on-line survey was designed using the tool *SurveySuite* (`http://intercom.virginia.edu/SurveySuite`), which allows a researcher to compose surveys from a pre-defined menu of item types. To complete the survey, a respondent receives a direct URL, fills in the form with their responses to the various items, and clicks a button to submit their responses. Once submitted, responses are accumulated in an underlying database, from which the researcher can obtain profiles of the responses-to-date and download for local analysis spreadsheets containing the responses.

The final pre-conference survey contained four sections. The first section polled respondents for their experiences with visualizations. We included items designed to gauge both the attitude toward and experience with various types of visualization. Some items were designed to detect the use of *mediated transfer* (that is, how instructors help learners relate their visual experience to the concepts being taught in the course). The second section of the pre-conference survey was designed to provide a profile of each respondent's teaching situation. Items addressed the availability and physical set-up of equipment, the type and size of classes, the number of learners and faculty members at the institution, and how extensively various types of visualization are used. The third section of the pre-conference survey sought information about the individual respondents, including how long they have taught, location of their institutions, and sources they use for information about visualization. We also requested contact information as well as information about how the respondents learned about the pre-conference survey. The final section, Closing Thoughts, provided respondents with an opportunity to share any other thoughts or information.

By the time we arrived in Århus for the ITiCSE 2002 conference, we had collected 29 responses. As we discussed the results (which are presented shortly), we decided that it would be useful to conduct a quick and very informal index card survey of conference attendees. Our index card survey had only two items: 1) Using visualizations can help learners learn computing science [5-point scale: *Strongly agree*; *Agree*; *Neutral/No opinion*; *Disagree*; *Strongly disagree*], and 2) How often do you use algorithm visualizations in your teaching? [4-point scale: *Extensively (every week)*; *Frequently (every other week)*; *Occasionally (once or twice per term)*; *Never*]. To conduct the survey, we distributed index cards at the beginning of the conference session during which each Working Group was to present information about the group's work. We requested that audience members should quickly answer the two items on the front of their index cards; we also invited respondents to add any comments or questions on the back of the index card. We collected the completed index cards before we presented results from our pre-conference survey.

#### 2.2.2 Background of respondents and their institutions

In this section, we summarize the results from three surveys: Grissom's survey from ITiCSE 2000, our pre-conference online survey, and our ITiCSE 2002 index card survey. Table 1 summarizes the number of respondents and gives some background information about the respondents and their institutions. Full results of the pre-conference survey, including the full set of items, are available on the working group's web site at `http://www.animal.ahrgr.de/iticseWG.html`.

In terms of the size of the respondents' institutions, for the Grissom survey the average enrollment was 7893 students. For the pre-conference survey, the responses to item 2.1 showed that the average enrollment across the 29 respondents' institutions was 7402 students, with the distribution of size representing the full range of possibilities: Two respondents (7%) came from institutions with 500 or fewer students; three respondents (10%) had 501 to 1000 students at their institution; four institutions (14%) had from 1001 to 2000 students; six institutions (21%) had 2001 to 4000 students; five respondents (17%) each reported the ranges 4001 to 7500 students and 7501 to 15000 students enrolled; and four institutions (14%) had at least 15001 students enrolled.

Item 2.2 from the pre-conference survey asked respondents approximately how many students are studying computing as a major, minor, or service course at their home institutions. The form of the response for each area was open-ended, so respondents entered their own numeric responses rather than being constrained to a menu of choices. The number of ma-

| | **Grissom Survey** (July 2000) | **Pre-conference Survey** (April-June 2002) | **Informal Index Card Survey** (June 2002) |
|---|---|---|---|
| Number of respondents | 91 | 29 | 66 |
| Number of countries | 21 | 11 | — |
| Avg. teaching experience [years] | 17 | 14 | — |
| Avg. number CS faculty | 17 | 23 | — |
| Population surveyed | ITiCSE 2000 Attendees | Various listserv lists | ITiCSE 2002 Attendees |

Table 1: Summary information for the three surveys

jors reported varied from 50 to 3000, with nine (31%) reporting 50 to 100 majors, sixteen (55%) reporting 101 to 1000 majors, and five (17%) reporting 1001 to 3000 majors. There was no apparent relationship between size of institution and number of computing majors reported. For two respondents (7%), the size of the institution matched the number of majors (1500 and 2000, respectively); as expected, neither of these respondents reported any students with computing as a minor or service course. Two respondents did not respond to this item. As a point of comparison, the Grissom survey resulted in an average of 559 computing majors, while the average for the pre-conference survey was 561 computing majors.

For the number of students with computing as a minor and in computing service courses, nine (31%) and ten (38%) respondents either reported *none* in the respective category or gave no response. The number of minors reported ranged from 5 to 3900, with twelve (41%) reporting 10 or fewer minors, six (21%) reporting 11 to 50 minors, four (14%) reporting 51 to 600 minors, and five (17%) reporting 1000 to 3900 minors. The number of students reported in service courses ranged from 10 to 5000, with fourteen (48%) reporting 100 or fewer students, seven (24%) reporting 101 to 900 students, and six (21%) reporting 1000 to 5000 students in service courses. Once again, there was no apparent relationship between number of students studying computing as a minor or service course and the total number of students enrolled in the institution. The Grissom survey did not consider the number enrolled in minor or service courses. For the pre-conference survey, averaging only over the non-zero/non-missing responses, there were an average of 495 minors and an average of 818 students in service courses.

Items 2.3 and 2.4 from the pre-conference survey asked for the number of faculty responsible for teaching computing courses and the number of faculty, including the respondent, who use dynamic visualizations to support learning. The number of computing faculty ranged from 2 to 100, with eight (28%) reporting 2 to 8 faculty members, twelve (41%) reporting 11 to 30 faculty members, five (17%) reporting 40 to 100 faculty members, and two (7%) omitting this item. The number of faculty using dynamic visualizations ranged from 1 to 20, with three respondents (10%) each reporting they had 1, 2, 4, and 5 faculty members using dynamic visualization, eight (28%) reporting 3 faculty members, two reporting 10 faculty members, three (10%) reporting 15 to 20 faculty members, two (7%) reporting that no faculty are using dynamic visualization, and two (7%) who omitted this item. For the pre-conference survey, the average number of computing faculty was approximately 24; the average number of faculty members using dynamic visualizations to support learning was slightly less than 5. On the Grissom survey, an average of 17 faculty members were responsible for teaching computing courses at the respondents' institutions; no results were available for the number of computing faculty who use computer-based visualizations to support student learning.

Comparing the responses to items 2.3 and 2.4 from the pre-conference survey, the ratio of faculty using dynamic visualizations to the number of computing faculty at an institution ranges from 4% to 75%. Three respondents (10%) reported a ratio in the range 4% to 10%, nine (31%) reported a ratio in the range 11% to 20%, ten (34%) reported a ratio in the range 31% to 50%, and two (7%) reported a ratio of 75%. One response was hard to interpret because the respondent said that the number of faculty using dynamic visualizations was greater than the number of faculty responsible for teaching computing courses (17 to 2).

Items 2.6 and 2.7 on the pre-conference survey asked the respondent to report the most common classroom set-ups at their institution and the classroom set-up they personally used most often. Table 2 summarizes the results of these two items. The most common set-up, reported by about half of the respondents, is that the computer and a ceiling-mounted projection system remain in the room. Nearly as common (reported by 41%) are classrooms where the projector is permanently mounted and the instructor brings their own computer. It is equally common (reported by 41%) to have the students sitting at terminals during class but with no central coordination with the instructor's equipment. The type of classroom that respondents normally teach in reflects the same trends as observed in the common classroom set-ups.

The Grissom survey included an item similar to item 2.7 from the pre-conference survey: "Which describes the classroom you teach in the most often?"; It is interesting to note

| | Common classroom set-ups | Classroom taught in most often |
|---|---|---|
| Computer and ceiling-mounted projection system remain in the room | 14 (48%) | 11 (38%) |
| Projector remains in classroom and bring own computer | 12 (41%) | 5 (17%) |
| Students sit at terminals, but there is no central coordination with instructor's equipment | 12 (41%) | 4 (14%) |
| Computer(s) and projector(s) are delivered to the classroom as needed | 8 (28%) | 3 (10%) |
| Computers and projectors are not available | 5 (17%) | 1 (3%) |
| Projector remains in classroom and computer is delivered | 4 (14%) | 1 (3%) |
| Students sit at terminals that are coordinated with instructor's equipment (e.g. instructor can see what student is doing) | 4 (14%) | 1 (3%) |
| Computer remains in classroom and projector is delivered as needed | 1 (3%) | 3 (10%) |

Table 2: Common classroom set-ups and classroom set-up used most often

that the percentage of respondents with classroom set-ups with permanently installed computer and ceiling-mounted projection systems was much higher among the 91 respondents for the Grissom survey (62%) than for the 29 respondents for the pre-conference survey (48%). The number of respondents reporting that a computer and projector are delivered to the classroom as needed was similar: 21% for the Grissom survey and 28% for the pre-conference survey. On the Grissom survey, 10% of the respondents reported that computers and projectors were not available, compared to 17% from the pre-conference survey. The only other category reported on the Grissom survey, that the rooms had permanently installed computers and the projector was delivered as needed, included 7% of the respondents, compared to 3% of the respondents for the pre-conference survey.

With respect to the teaching experience of respondents, the respondents for the Grissom survey reported an average of 17 years. For the pre-conference survey, responses on item 3.1 resulted in an average of 14 years of teaching experience, with two respondents (7%) reporting 1-4 years, nine respondents (31%) 5-10 years, five respondents (17%) 11-14 years, seven respondents (24%) 15-22 years, and six respondents (21%) 24 or more years of experience.

#### 2.2.3 Results related to using visualization in teaching

The initial item on the pre-conference survey, item 1.1, asked respondents to indicate their strength of agreement with the statement "Using visualizations can help students learn computing concepts." All 29 respondents either strongly agreed or agreed with this statement (59% and 41%, respectively). The Grissom survey did not include such an item. On the index card survey, twenty nine of 67 respondents (43%) strongly agreed with this statement, thirty three (49%) agreed, and the remaining five (8%) indicated that they were neutral or had no opinion. Out of the 93 computer science educators who responded to this question in the surveys, *none* disagreed at all with the premise that visualizations can help students as they learn computing concepts. This strong perception among educators that visualization can help makes the lack of direct supporting evidence (see Section 2.3) particularly intriguing.

In polling respondents for how they use visualization, the Grissom survey addressed the use of both static and dynamic visualizations in the classroom. As shown in Table 3, all respondents indicated that they used static visualizations such as slides, images, and hand-drawn figures on the board at least sometimes. For dynamic visualizations such as computer software, animations, and interactive demonstrations, over half of the respondents used dynamic visualizations in the classroom only a few times per term, with 13% never making use of dynamic visualizations. Only a quarter of the respondents used dynamic visualizations at least once per week. The Grissom survey also looked at how students used dynamic visualizations outside of class. Nearly one quarter of the respondents said that students never used dynamic visualizations outside of class; slightly more than half reported that students used dynamic visualizations outside of class at least a few times per term.

| **Type of usage** | almost every day | once per week | a few times per term | never |
|---|---|---|---|---|
| Static | 72% | 20% | 8% | 0% |
| Dynamic | 10% | 23% | 54% | 13% |
| Dynamic outside class | 5% | 19% | 53% | 23% |

Table 3: Frequency of Visualization Use in Grissom's Survey

For the pre-conference survey, we chose not to ask respondents about their use of static visualization, since we believe that virtually all computing instructors use some form of static visualization frequently. Thus, the pre-conference survey focused solely on the use of dynamic visualization. When asked how they have used dynamic visualizations in item 1.2, 97% (all but one respondent) replied that they at least occasionally demonstrate visualizations during classroom lectures. [These percentages pool the responses for the three positive options *used extensively (about every week)*, *used frequently (about every other week)*, and *used on occasion (at least once or twice per term)*.] About two-thirds of the respondents reported that they make visualizations available for student use outside of closed laboratories, nineteen (66%) with prior guidance or instruction and seventeen (59%) without prior guidance or instruction. Seventeen of the respondents (52%) indicated that they required student use in a closed, supervised lab, and the same number indicated that visualizations are available for optional student use in a closed, supervised lab. Only twelve of the respondents (41%) require student use during classroom lectures.

The second item on the index card survey asked ITiCSE 2002 participants to indicate how frequently they use algorithm visualizations in their teaching. Two out of 64 respondents who answered this item (3%) replied that they use visualizations nearly every week, fifteen (23%) that they use them every other week, and twenty nine (45%) that they use visualizations only once or twice per term. Eighteen respondents (28%) indicated that they never use such visualizations.

All of these responses about frequency of use point to the fact that, even among educators who use visualization, few tightly integrate it with other aspects of their courses. This contrasts sharply with the experience of educators at Brown University, where they found early success with Marc Brown's BALSA system [8]. They claimed:

> Much of the success of the BALSA system at Brown is due to the tight integration of its development with the development of a textbook and curriculum for a particular course. BALSA was more than a resource for that course - the course was rendered in software in the BALSA system [3].

We will return to the issue of whether students are sufficiently familiar with the visualization tools they use when we discuss covariant factors in Section 4.4.

In interpreting the pre-conference survey, we were also curious about certain combinations of ways in which the respondents use dynamic visualization. For example, six respondents (21%) require that students use visualizations during classroom lectures as well as for both required and optional use in a closed, supervised lab. Five additional respondents (17%) who require use during classroom lectures also have some sort of use in a closed, supervised lab; two of these (7%) require use in the lab setting, while for three (10%) such use is optional. In another comparison, we discovered that ten respondents (34%) make visualizations available for student use outside of a closed lab setting both with and without prior guidance and instruction. An additional fourteen respondents make visualizations available for student use outside of a closed lab setting, with a split of eight (28%) giving prior guidance or instruction and six (21%) without prior guidance.

Item 1.3 of the pre-conference survey asked respondents to indicate all ways in which learners use visualization. Three of the respondents (10%) indicated that students do not directly use visualizations. Nine of the respondents (31%) indicated that students must construct their own visualizations, for example, as homework. Seven of the respondents (24%) indicated that their students have tools available so they can construct their own visualizations. Six of the respondents (21%) indicated that *students are encouraged to construct their own visualizations*. None of these respondents chose the option students present their own visualizations. Four respondents (14%) filled in the *other* option: one indicated that students have the tools they make available helps student to visualize program structure; another indicated "passive use"; a third said the visualizations were pre-built for the students; and the fourth said that students use but do not construct visualizations.

Item 1.4 of the pre-conference survey asked what learners do along with the visualization. This question was designed in part to elicit information about the use of mediated transfer. [These percentages pool the responses for the three positive options *used extensively (about every week)*, *used frequently (about every other week)*, and *used on occasion (at least once or twice per term)*.] Twenty six of the respondents (90%) indicated that students watch the visualization in class. Twenty one (72%) said that students use the visualization in lab exercises. Twenty one (72%) replied that students must construct the visualization, while twenty (69%) have students experiment with different data sets. Fifteen (52%) ask students to give brief oral responses during class to questions about the visualization. Thirteen of the respondents (45%) say students must give written answers to describe their understanding of the visualization in an open-ended format such as an essay question. Eleven (38%) ask students to give written answers to closed-form questions about the visualization, such as fill-in-the-blank, multiple-choice, or short answer. Ten (34%) ask students to give longer, more open-ended oral responses to describe their understanding of the visualization. In the final comments, one respondent explained that one way they use visualizations is to ask students "what if" questions; the student must then develop a hypothesis and test it.

When asked on item 1.5 of the pre-conference survey to describe any ways in which they had used visualizations be-

sides the options mentioned in items 1.2 - 1.4, eleven respondents provided comments. One respondent described the use of stacking cups to demonstrate sorting algorithms; students then study Java applets demonstrating the sorts, and finally must transfer that experience to sorting the cups, including a physical demonstration of the sorting algorithms with the stacking cups as part of their exam. Another respondent mentioned using conceptual maps for working with object-oriented programming concepts. A third respondent clarified that students construct visualizations, show how given data structures are modified by given algorithms, and then must present the state of the data structure in a visual form that is submitted to a server that assesses it and gives feedback for the student. Yet another respondent said that students learn to express data models using entity relation diagrams and relational schema, and work extensively with data-structure diagrams such as indexes, B-trees, and B+ trees to study search operations as well as the effects of operations such as insert and delete. This respondent also has students use many graph and tree models to represent different problems and to design solutions using graph algorithms. Another respondent said that students select to work with a visualization package to produce a non-scientific data visualization using atypical visualization paradigms, for example, in genealogy. Another respondent values visualization to help make issues tangible and uses visualization for simple examples such as how compilation or event composition works. Another respondent described a system that has been used world-wide and provides students with an environment where they can test virtually every important concept in geometric modeling. This software is used in a classroom that merges the ideas of lab and lecture to do demonstrations, design, and exercises. Students are also required to use this system to generate test data for their programming assignments. This respondent also discusses another system, designed to teach multi-threaded programming, with an automatic visualization component that can be activated automatically by a user program. Students are required to use this system for their programming assignments, which allows them to see the concurrently running threads, the dynamic behavior of the threads, and all activities of synchronization primitives.

Item 1.6 of the pre-conference survey was a free-response item that asked respondents to explain their pedagogical aim in using visualizations. Twenty-two respondents (76%) provided comments. Several respondents mentioned that dynamic visualization is useful in explaining dynamic behavior or state changes. Others mentioned that the use of dynamic visualization helps improve comprehension of both theoretical and practical issues because it makes the ideas more tangible. One respondent observed that dynamic visualization gives students immediate feedback and also helps students understand patterns that abound in the structures. Another respondent has observed in twenty years of teaching that *everyone* learns better through visualization. This respondent believes that visualization and kinetic experience better instill learning in long-term memory than do other sensory and cognitive processes and concluded with the observation that learners better manipulate symbols when they have visual representations of process. A second respondent explained that visualizations have progressed from a series of PowerPoint slides to GIF animations to flash "movies". This respondent's students claim that visualization helps them understand how the algorithms work. Another respondent uses visualizations to convey more difficult or visually oriented material, explaining that dynamic visualizations express such ideas more clearly and succinctly than words do. Another respondent expressed a similar notion, observing that when students exercise on a conceptual level how algorithms work, they understand the algorithms better. This respondent went on to say that visual representation of the data structure supports this conceptual understanding, because implementation details no longer blur the students' view. Another respondent has used visualizations to help students achieve a deeper understanding of class concepts. This respondent has found that dynamic visualizations reach a broader cross-section of students than do some other pedagogical techniques. Another respondent commented that visualizations are fun for students and instructors alike. Yet another respondent reinforced this view by observing that visualization provides better motivation for students as well as better illustration of the dynamic nature of algorithms and data structures. This respondent values the use of thought-provoking questions combined with visualization, since this tends to encourage interesting discussions. Another respondent described the use of dynamic visualizations that are created on the fly from source code; paradoxically, these visualizations are also fairly static, in that there is no animation. This respondent feels that these visualizations express relationships between classes in a much more obvious way than is possible with a textual description. In general, this respondent feels that by presenting the same concepts both verbally and graphically, students seem to profit from experiencing two approaches to the same material.

Item 1.7 asked respondents to list courses at their institutions where dynamic visualization is used regularly. The most frequently named course was data structures, both with and without mention of an algorithms component. Other courses mentioned by more than one respondent were CS1, introductory programming (perhaps the same as CS1?), computer networks, artificial intelligence, software engineering, databases, and survey of CS/CS0. Courses that were mentioned once include compilers, computer architecture, CS theory, data mining, data visualization, image processing, office technologies, multimedia and Internet technologies, scientific visualization, and simulation and modeling.

Item 1.8 asked respondents to indicate benefits that they had experienced from using visualization; the form of the item allowed respondents to select any number of the options. In designing the items, we were unable to use results from the

Grissom survey; while that survey included a similar item, the results were unavailable. Instead, we generated the options through an informal process of polling working group members and colleagues. In decreasing order of frequency based on the pooled responses for the two positive options *a major benefit* and *a minor benefit*, the benefits were:

- 90%: the teaching experience is more enjoyable
- 86%: improved level of student participation
- 83%: anecdotal evidence that the class was more fun for students
- 76%: anecdotal evidence of improved student motivation
- 76%: visualization provides a powerful basis for discussing conceptual foundations of algorithms
- 76%: visualization allows meaningful use of the available technology
- 72%: anecdotal evidence of improved student learning
- 62%: (mis)understandings become apparent when using visualization
- 52%: objective evidence of improved student learning
- 48%: interaction with colleagues as a benefit

Item 1.9 was a free-response item that allowed respondents to list additional benefits they had experienced from using visualization. Eight respondents (28%) provided comments. One respondent mentioned the advantage of anonymous help-seeking. Another mentioned that with visualization, big-O measures of efficiency have been transformed from one of the most difficult concepts into a straightforward one. This respondent also observed that students spend much more time when visualization is involved. Another respondent remarked that data visualization is a good way to get students thinking about space, time, processes, and communication. Yet another respondent felt that dynamic visualizations capture students' attention, particularly when color and sound are used effectively. Another respondent had no evidence to support this observation, but felt that students grasp the main concepts quicker and more easily (and with less effort from the instructor). Another respondent observed that algorithm simulation exercises promote learning but at the same time was uncertain whether this benefit comes from compulsory exercises and automatic assessment of the submitted solutions, or from the visual form used when constructing the solutions.

For the Grissom survey, respondents were asked to describe up to two reasons they were reluctant or unable to use visualizations. The reasons listed included time (for developing visualizations, installing software, transitioning into the classroom, learning new tools, and preparing courses); equipment (including concerns about availability and reliability); the lack of effective and reliable software; platform dependence and the lack of effective development tools; the difficulty of adapting existing materials to teach what and how the educator wants to teach; the fact that students are too passive if they simply watch demonstrations in a darkened room; a concern that AV may hide important details and concepts; and the lack of evidence of effectiveness.

Item 1.10 for the pre-conference survey asked about factors that make the respondent or the respondent's colleagues reluctant or unable to use dynamic visualizations. To develop the options for this item, we began from the list of factors from the Grissom survey presented in the previous paragraph. Respondents could select any combination of options that reflect factors they believe discourage the use of dynamic visualization. In decreasing order of frequency based on pooling the responses for the two positive options *a major factor* and *a minor factor*, the impediments were:

- 93%: time required to search for good examples (on the Web, in the literature)
- 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
- 69%: lack of effective and reliable software
- 69%: time it takes to transition them into the classroom
- 66%: unsure of how to integrate the technology successfully into a course
- 66%: time it takes to install the software
- 59%: lack of evidence of effectiveness
- 55%: concerns about the equipment or presentation location (e.g. darkened room)
- 48%: unsure of how algorithm visualization technology will benefit students
- 38%: students are too passive
- 31%: AV may hide important details and concepts

Item 1.11 of the pre-conference survey was a free-response item that asked respondents for reasons other than those listed in item 1.10 that make the respondent or colleagues reluctant to use dynamic visualizations. Seven respondents (24%) provided such comments. One respondent was unable to share colleagues' ideas about visualization, having never discussed the topic with them in any detail. Another

respondent, a self-described developer of visualization tools, observed "the above questions are not my concerns". Another comment was a reaction to the option in item 1.10 that students are too passive when viewing visualizations; this respondent observed that, to the contrary, students become more active when they are dealing with visualizations. Another respondent remarked that the impediment is being too busy with other work to make big changes in course content and presentations. Yet another respondent mentioned cost and resource issues. Another respondent is convinced that the most profound reason colleagues do not use dynamic visualization is a personality characteristic that permits one teacher to engage in high risk behavior such as using children's toys as instructional props. This respondent observes that some educators feel it is very high risk to engage in this sort of extravert activity.

In item 1.12 for the pre-conference survey, we provided several options for techniques respondents might use to evaluate the effectiveness of visualization. Respondents could select any number of the options. In decreasing order of frequency based on pooling the responses for the two positive options *major source of information* and *minor source of information*, the most often used evaluation techniques were:

- 83%: informal feedback during class
- 83%: informal feedback outside of class
- 52%: brief paper-and-pencil questionnaire at the end of the term
- 48%: brief paper-and-pencil questionnaire during class
- 45%: informal comparison of results between two or more groups
- 34%: on-line questionnaire at the end of the term
- 21%: on-line questionnaire during class

Respondents were invited to elaborate on their evaluation approach(es) and results in item 1.13, a free-response item. Five respondents (17%) provided such comments. One respondent mentioned assessing student work using predefined objectives and rubrics. Another respondent had used a pre-test, a post-test, and an anonymous attitudinal survey that includes more than 40 questions for each course. The attitudinal survey permitted students to rate each topic and feature of the visualization tool. Another respondent has compared student behaviors from before beginning to use visualization and simulations to behaviors after using these techniques. This respondent has observed a profound difference between the student's being able to reproduce code and being able to physically manipulate a group of physical objects such as people or stacking cups. The fourth respondent described evaluation work that concentrates on the effect of automatic assessment of algorithm simulation exercises. The fifth respondent had just started using a formal approach to the evaluation of visualizations.

#### 2.2.4 Discussion of results

The respondents to the pre-conference survey seemed generally knowledgeable about visualization and convinced that visualization can make a difference in helping learners better learn concepts. As evidence of the overall knowledge of the respondent pool, eleven respondents from the pre-conference survey (38%) indicated that they had created their own visualization tools from scratch.

When asked to indicate their level of agreement with the statement *Using visualizations can help learners learn computing concepts*, all respondents for the pre-conference survey and 93% of the respondents for the index card survey agreed or strongly agreed. That is, among these 98 computing educators, only five responded that they were neutral or had no opinion; no one disagreed with the statement to any degree. This overwhelming belief that visualization can improve learning is somewhat surprising given the lack of empirical evidence that visualization really does help. Our concern that the results from the pre-conference survey were biased toward individuals who were favorably inclined toward the use of visualization led us to design the index card survey. We believed that folks attending ITiCSE 2002 were less likely to be biased favorably toward visualization than the individuals who chose to complete the pre-conference survey. Thus, the index card survey strengthened our understanding that computing educators are convinced that visualization can make a difference in helping learners better learn concepts.

While none of the surveys addressed the effectiveness of visualizations that produce a sequence of discrete snapshots versus those that portray an algorithm's execution via a smoothly continuous animation, two respondents volunteered knowledgeable commentary. They observed that continuous animation offers no real advantage in the learning that occurs. One of these respondents indicated that smooth animations "are not as useful since there is little time for the learner to notice patterns. It is better to lay out the frames in sequence or in other arrangements to allow viewers to look for patterns without time pressure". The other response came from a psychologist studying the effectiveness of visualizations in a variety of learning contexts. This respondent indicated "As for animations, there are no well-controlled studies showing any advantages of animated over still graphics in teaching. ... We give a cognitive analysis of why animations aren't more successful. In addition, we have some half dozen studies of our own where visualizations helped over equivalent text, but animated visualizations were no better than static ones." These observations lead us to wonder about the learning pay-off, if any, for smooth animations. While many individuals involved in visualization research have worked long and hard to achieve smooth animations, this type of

comment suggests that such efforts may have little or no influence in improving learning.

A dilemma that may be inescapable when developing animations is the very personal nature of the visual mapping in the viewer's mind and what the graphics can depict. One respondent observed "There are too many journals and conferences. I rarely used web repositories because the tools that I want are rarely available or not up to my and my students' expectation." This same respondent added in another response "Most visualization tools do not address the real needs of students. People designed so many visualization tools for *visualization* and/or *animation* only and forgot what these tools are for. As a result, on the surface, many of these tools have a pretty face without content. Many designers only consider their wants and never put themselves into the shoes of the users." Despite the negative tone of this response, it does indicate that, even among experts who understand algorithms, there is often little agreement about what the visual display should be. Perhaps designers of visualizations must do a better job of analysis before plunging into the design and implementation of visualization tools. In this case, the question is how to proceed with such an analysis.

Item 1.2 from the pre-conference survey indicated that the most frequent use of visualizations is for demonstrations during classroom lectures. The next most frequent uses were to have the visualization available for student use outside of a closed lab setting, both with and without prior guidance or instruction. All three of these strategies involve learners in relatively passive interactions with the visualizations. In response to item 1.4 from the pre-conference survey, 17% of the respondents indicated that they never ask learners to construct the visualization. The responses to item 1.4 also reveal that only about half of the respondents ask their students to respond to written or oral questions about the visualizations that are used in the course. We must conclude that, for many computing educators, even though they use visualizations, the visualizations are not really woven into the instructional fabric of the course. If an educator does not expect learners to be able to converse and answer questions about activities they carry out as part of the learning process, why require the activity? One respondent emphasizes the value of having learners do exercises directly related to the visualizations by pointing out that the exercises are more important than the visualizations themselves. This respondent states "We know that algorithm simulation exercises promote learning. However, it is not clear whether this benefit comes from compulsory exercises and automatic assessment of the submitted solutions, or from the visual form used when constructing the solutions." If this is indeed the case, then the logical conclusion is that educators who are engaged in visualization need to devote more time to the accompanying instructional materials, rather than having a single-minded focus on the graphics of visualization.

An interesting issue is whether there are any advantages to having learners actively engage in some form of viewing expert-constructed visualizations over having learners construct their own visualizations as part of programming or other exercises. One respondent is very convinced that the latter is of greater value. This respondent observed: "I have also assigned students to prepare animations using some home grown software ... . The students implement standard graph algorithms and can see their code animated. ... It seems to be a much more engaging activity than working through animations exercises that I have packaged together." A related issue is the amount of time that students spend working with the material when visualization is involved. One respondent observed that students spend much more time when visualization is involved. If learners are spending more time, does this mean they are learning more material, getting a deeper understanding of concepts, or simply being distracted by the features of the visualization?

An underlying and important question when considering the use of visualization is whether some kinds of learners are better served by visualization. One respondent pondered that the computing curriculum is attracting more and more right-brain learners, in spite of being a left-brain curriculum. It makes sense to consider learning characteristics in studies designed to determine the effectiveness of visualization, since this may reveal information that will assist researchers in customizing tools to better serve different learning styles.

Item 1.10 addressed the issue of what makes educators reluctant to use visualization. We offered fourteen possible reasons an instructor might feel reluctant to use visualization. The option that was cited as a major impediment by two-thirds of the respondents was the time it takes to develop visualizations. Four other options were listed as major impediments by 45%-48% of the respondents. These were the time required to search for good examples, the time it takes to learn the new tools, the time it takes to adapt visualizations to teaching approach and/or course content, and the lack of effective development tools. It is telling that four of these five factors are related to the precious commodity of time. This hints that the most effective way to enable more instructors to use visualization will be to make it less time-consuming and more convenient to do so. Although we feel that this conclusion is a key result that can be gleaned from our survey, it is not a topic that our report will address in depth.

The responses to item 1.12, which asked respondents about the techniques they have used for evaluating the effectiveness of visualization efforts, showed that, with few exceptions, evaluation was based largely on informal feedback or brief pencil-and-paper questionnaires. Thus, the evidence these respondents have collected is primarily anecdotal. One respondent observed "One of the areas I would like to see your group address is how to measure student engagement and student learning for programming exercises involving visualization that take place outside of class over a period of

a week or two. I believe that students find these very engaging, but I have no notion of how to collect data to illustrate this."

In the next section, we look at experimental results from prior studies of visualization effectiveness. These studies shed some light on what has been done in this area in the past, as well as providing an impetus for the work we propose in Sections 3, 4, and 5 of our report.

### 2.3 Review of Experimental Studies of Visualization Effectiveness

In [28], Hundhausen reports a meta-analysis of twenty-one experiential evaluations. The striking result from this meta-analysis is that the studies cast doubt on the pedagogical benefits of visualization technology. Indeed only 13 of those experiments ([11, §2 and 3]; [13]; [23, I, II, IV, V, VII, VIII]; [31]; [40, Chapters 6, 7, 9]) showed that some aspect of visualization technology or its pedagogical application significantly impacted learning outcomes.

Further analysis of these experiments suggests that they fall into two broad categories based on the factors they identify as critical to the experimental evaluation of visualization. The first category is represented by the work of Gurka [20], Hansen et al. [23, VI-VIII], Lawrence ([40], Chapters 4, 5, 7, 8), and Stasko, Badre, and Lewis [56]. These studies varied representational characteristics of the learning materials, including:

1. text versus animation,
2. text-first or animation-first, and
3. various graphical attributes of animations.

By contrast, a second group of studies ([11, §2 and 3], [13], [23, I - V], [30], [31], and [40, Chapters 6 and 9]) varied the level of learner involvement. In addition to having learners passively view an animation, these studies involved learners more actively by having them do one or more of the following:

1. construct their own data sets,
2. answer strategically-chosen questions about the algorithm,
3. make predictions regarding future algorithm behavior, or
4. program the algorithm.

Figure 1 summarizes these experiments, contrasting those with the more passive representational focus with those that had a more active focus on learner engagement. As shown, ten out of twelve (83%) experiments using learner engagement produced a significant result. Out of nine experiments that manipulated representations, only three (33%) showed a significant result. This suggests that what learners do, not what they see, may have the greatest impact on learning – an observation that is consistent with constructivist learning theory (see, for example, [47]).

For the full meta-analysis of the twenty-one experiments see [28]. This expanded analysis includes an assessment of the extent to which the experiments support alternative learning theories.

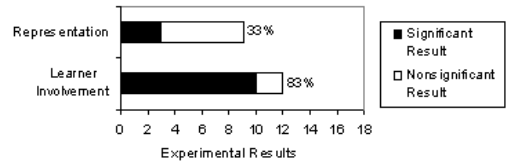


Figure 1: Results of visualization effectiveness experiments broadly classified by their independent variables

Given the trend that active engagement with visualizations is far more important to learners than the graphics that they see, and given the fact that these previous studies do not consider the impact of visualization technology on instructors' time and effort, it makes sense to design a new series of experiments that systematically explore the educational impact of alternative forms of active engagement with visualization technology. In the next section, we begin this design by classifying forms of learner engagement.

## 3 Engagement Taxonomy

In order to better communicate learners' involvement in an education situation that includes visualization, we broadly define six different forms of learner engagement with visualization technology. Since it is certainly possible to learn an algorithm without the use of visualization technology, the first category is "No viewing," which indicates that no visualization technology is used at all.

1. No viewing
2. Viewing
3. Responding
4. Changing
5. Constructing
6. Presenting

While this taxonomy does in some sense reflect increasing levels of learner engagement, we do not consider this to be an ordinal scale. The relationships among these six forms of engagement do not form a simple hierarchical relationship. Figure 2 illustrates, in the form of a Venn Diagram, the overlapping possibilities among the last five of these engagement

categories, with "Viewing" forming the universe on which all of the last four forms of engagement must occur. The first category in our list, "No viewing," does not appear in the Venn diagram.

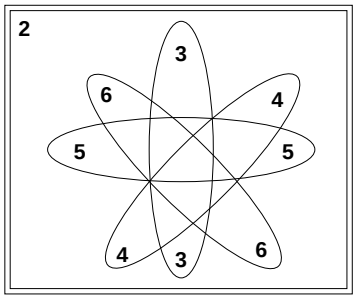


Figure 2: Possible Overlaps in the Engagement Taxonomy. Basic regions are 2 (Viewing), 3 (Responding), 4 (Changing), 5 (Constructing), 6 (Presenting)

### 3.1 Viewing

"Viewing" can be considered the core form of engagement, since all other forms of engagement with visualization technology fundamentally entail some kind of viewing. The Venn diagram of Figure 2 indicates this by providing "Viewing" as the universe in which all other forms of engagement exist. Viewing is also probably the form of engagement where the largest number of variations can be found. For example, a learner can view an animation passively, but can also exercise control over the direction and pace of the animation, use different windows (each presenting a different view), or use accompanying textual or aural explanations.

Viewing by itself is the most passive of the forms of engagement; indeed, aside from controlling a visualization's execution and changing views, viewing does not entail active involvement with a visualization. Note that, in its broadest sense [49, 50], visualization includes auralization; thus, we include "hearing" within this category.

The remaining four categories all include viewing. They do not, however, create a strict hierarchy, even though they can be done in concert with each other (see intersections in Figure 2).

### 3.2 Responding

Category 3 in the engagement taxonomy is "Responding". The key activity in this category is answering questions concerning the visualization presented by the system. For example, learners might be asked such questions as

- "What will the next frame in this visualization look like?" (prediction)
- "What source code does this visualization represent?" (coding)
- "What is the best- and worst-case efficiency of the algorithm represented by this visualization?" (efficiency analysis)
- "Is the algorithm represented by this visualization free of bugs?" (debugging)

In the responding form of engagement, the learner uses the visualization as a resource for answering questions. As such, the engagement involves only limited interaction with the visualization. However, responding to a question may involve activities that ultimately culminate in further viewing activities. For example, in a debugging activity, a valid response to the question, "Is there a bug in this program?" may be to alter source code and regenerate the visualization.

### 3.3 Changing

Category 4 in the engagement taxonomy, "Changing," entails modifying the visualization. The most common example of such modification is allowing the learner to change the input of the algorithm under study in order to explore the algorithm's behavior in different cases (for example, [40, Chapter 9]). Asking the learner questions, as in the previous category, can further enhance this form of engagement. For example, the system of Korhonen, Sutinen and Tarhio [39], based on the Matrix framework [38], prompts learners to provide input data sets that cover all possible execution paths.

### 3.4 Constructing

Category 5 in the engagement taxonomy is "Constructing". In this form of engagement, learners construct their own visualizations of the algorithms under study. Hundhausen and Douglas [27] have identified two main ways in which learners may construct visualizations: direct generation and hand construction.

- In direct generation, the most common construction technique, learners map a program or an algorithm to a visualization. This is done automatically for program visualizations that visualize the code or state of the data structure. Algorithm animation is a more abstract view that is usually achieved by annotating algorithm source code of algorithms under study with animation commands, so that animations are generated as a byproduct of algorithm execution [48, 55]. An intermediate technique is producing automatically program visualizations that let the user exert some control over the contents of the final animation, so that they can be tailored to a specific educational goal [42]. Alternatively, learners might be given a predefined visual representation, which they manipulate so as to simulate the algorithm [15, 38]. In this case, learners' simulations may be automatically checked against the actual execution of the algorithm.

- Using hand construction, learners might use a drawing or animation editor (for example, [12]) or even simple art supplies (pen, paper, scissors, etc.) [25], to hand-construct their own visualizations. Lacking a formal execution model of the algorithm, this technique casts learners in the role of a virtual machine that executes the visualization of the algorithm. However, in some construction-type editors such as JFLAP [29], the model created can then be executed.

It is important to note that the "Constructing" form of engagement does not necessarily entail coding the algorithm. Indeed, while learners may need to implement the algorithm to be visualized if they are using direct generation, the implementation of the algorithm could certainly be supplied to them in advance. In the case of hand construction, there is no notion of an underlying "driver" algorithm's implementation; hence, there is no need to code.

### 3.5 Presenting

Level 6 in the engagement taxonomy, "Presenting", entails presenting a visualization to an audience for feedback and discussion. The visualizations to be presented may or may not have been created by the learners themselves. For example, Hundhausen [25] describes an undergraduate algorithms course in which learners were required to present visualizations of their own creation. Another possibility is a presentation exercise in which learners are required to present to their instructor and peers a visualization they found through a web search.

These six categories of engagement provide a basis for the metrics we will propose in the next section.

## 4 Metrics for Determining Effectiveness of Visualization

In this section, we develop a basis for defining metrics for determining the effectiveness of visualization. We begin by discussing how Bloom's taxonomy can be used to give a concrete definition of expectations for a learner's understanding. Next, we provide a specific example of applying Bloom's taxonomy in the context of algorithmics and data structures. With the examples providing context, we then explore factors that could be measured in order to demonstrate learning improvement. Finally, we explore additional factors that can be collected to help profile the learners and provide a better context for data analysis.

### 4.1 Learner Understanding and Bloom's Taxonomy

In order to design studies to investigate the effectiveness of various strategies for engaging learners in visualization, we must first break down exactly what we expect of learners studying a particular topic, a very difficult undertaking. Rather than attempting to provide an all-encompassing breakdown for all of computer science, we use a general taxonomy developed by Bloom in 1956 [5]. It then becomes incumbent upon any particular study of visualization effectiveness to define understanding within the particular area of CS in which that study is being conducted.

Bloom's taxonomy structures a learner's depth of understanding along a linear progression of six increasingly sophisticated levels:

**Level 1:** *The knowledge level*. It is characterized by mere factual recall with no real understanding of the deeper meaning behind these facts.

**Level 2:** *The comprehension level*. At this level, the learner is able to discern the meaning behind the facts.

**Level 3:** *The application level*. Now the learner can apply the learned material in specifically described new situations.

**Level 4:** *The analysis level*. The learner can identify the components of a complex problem and break the problem down into smaller parts.

**Level 5:** *The synthesis level*. The learner is able to generalize and draw new conclusions from the facts learned at prior levels.

**Level 6:** *The evaluation level*. The learner is able to compare and discriminate among different ideas and methods. By assessing the value of these ideas and methods, the learner is able to make choices based on reasoned arguments.

### 4.2 Algorithmics and Data Structures in the Context of Bloom's Taxonomy

As an example of how a researcher conducting an effectiveness study could map a particular area to Bloom's breakdown, we develop some sample tasks in the area of algorithms and data structures. We recognize that creating such a mapping is not a trivial task and the following classification may raise objections. Many activities in algorithmics, for example implementation and analysis, include tasks of hugely varying complexity. Such tasks cover a wide range of the levels in Bloom's taxonomy. In the remainder of this section, we consider some of the problematic issues before we present our detailed list of knowledge levels.

The first issue is the differing complexity of basic concepts within the field of algorithmics. As in any field, algorithmics includes a substantial number of terms and concepts that build the vocabulary of the field but will be new to the learner. For example, basic terms of graph theory, like nodes, edges, paths, and cycles are easy to understand. In contrast, concepts such as the depth-first search (DFS) algorithm for traversing graphs are much more difficult. Knowledge about

names of data structures and algorithms belongs to the lowest level in the taxonomy (knowledge level) whereas being able to explain their working belongs to the second level (comprehension).

The second problematic issue is that algorithmics includes knowledge that can be viewed as both conceptual and related to implementation. It seems reasonable to assume that to be able to implement an algorithm a learner must understand its working on a conceptual level. However, it is unclear whether implementing an algorithm in a programming language belongs to level 2 (comprehension) or to level 3 (application). If a learner implements an algorithm that is understood on a conceptual level, the learner certainly applies conceptual knowledge. However, implementation is just another, more detailed form of representing the algorithm. Thus, we propose that a test item to "write the code for implementing Quicksort" belongs to level 2 (comprehension) whereas a programming assignment where the learner must apply a sorting algorithm to sort an array of records can be classified as belonging to level 3 (application).

In general, applying algorithms to solve real-world problems almost always requires some modifications to the textbook code examples learners have studied. Often solutions require the combination of various algorithms and data structures. This requires skills like analyzing the problem area, identifying objects and their structures, choosing appropriate representations for the structures, and deciding which algorithms and structures best apply to solving the problems. Problem analysis is a task belonging to level 4 (analysis). Constructing the solution belongs to level 3 (application) if learners can manage with algorithms that they know, or to level 5 (synthesis) if learners need to produce something that is new to them. Of course, the boundary between different problems can often be subtle, but assignments that require doing research to discover new algorithms must belong to at least level 5 (synthesis). Finally, when learners must evaluate their own solutions based on some criteria, they are working at level 6 (evaluation), the highest level in Bloom's taxonomy.

The third issue we must consider is that algorithm analysis is a complicated undertaking that has parts belonging to several levels in Bloom's taxonomy. Knowing the basic concepts, like big-O notation or worst-case complexity, belongs to level 1 (knowledge), whereas following and repeating analysis of some algorithm requires more understanding (level 2, comprehension). Carrying out an analysis requires learners to apply their knowledge of algorithm analysis to a specific problem. However, the range of difficulty of such analysis problems is large and hence difficult to pigeon-hole into just one of Bloom's levels. We suggest splitting this topic across level 3 (application), level 4 (analysis), and level 5 (synthesis). In writing an exam, the instructor could ask learners to analyze a basic algorithm covered in the course or to produce a simple modified version. When analyzing more challenging and complex solutions, learners must split the problem into simpler tasks (a level 4 analysis activity). Given these simpler tasks, learners can then analyze each independently. In complicated problems, learners may have to use and combine several different techniques to develop a solution, which employs skills needed in doing algorithm research (level 5, synthesis). Finally, when learners are charged with critiquing their own analysis methods and results, they are performing level 6 (evaluation) activities.

This initial discussion lays the groundwork for our example of how algorithmic knowledge can be matched to Bloom's taxonomy. Tables 4 and 5 illustrate for each level of Bloom's taxonomy what learners can be expected to do. The tables also provide concrete example tasks and assignments for each level.

### 4.3 Other Factors to be Measured

In the previous section, we presented the levels of knowledge the learner can achieve. In this section, we consider more closely those topics that can be measured as learner improvement, as well as factors that can have a confounding effect on results.

1. Learner's progress

   During a course of study, learners should make progress so that their knowledge deepens along Bloom's taxonomy hierarchy, that is, so they can begin to work at more advanced levels. However, this progress is not discrete. On each level, learners can perform either poorly or well, although the deeper their knowledge, the better they should perform at the lower levels. When assessing learners' knowledge, the hierarchical nature of knowledge must be considered. Suppose the instructor sets up an assignment that tests learners' knowledge on some level and grade the assignment in a traditional way using some grading scale, for instance a point scale from 0 to 6. Then the evaluation study could produce results like "We observed that on an assignment testing level 2 knowledge, learners using method A gained an average 4.6 out of 6 points whereas learners using method B gained only 3.2 out of 6 points." A t-test could be used to determine whether this difference is statistically significant.

2. Drop-out rate

   For some learners, the difficulties they encounter in their studies cause them to decide to drop a course. There can be many reasons for such a decision. The reasons that are most relevant in the context of this report are the learners' motivation for studying the topic and their attitude toward different learning methods and tools used in the course. Measuring drop-out is not always straightforward, since some learners may register for the course before they make their final decision to take it. Thus, the number of learners who remain in the course long enough to submit

| Level in Bloom's Taxonomy | What Can The Learner Do At This Level | Sample Tasks And Assignments |
|---|---|---|
| 1 - Knowledge | Recognize and informally define specific concepts in algorithmics, like stacks, trees, graphs, Quicksort, AVL-tree, linear probing, or basic analysis concepts like Big-O notation and worst-case complexity. | • Define the following concepts: directed graph, binary tree, array.<br>• List three different sorting algorithms. |
| 2 - Comprehension | • Understand the general principle behind an algorithm and explain how it works using words and figures.<br>• Define concepts formally, that is, recognize their essential properties and present them in an exact way.<br>• Understand the key concepts involved in an algorithm and their role in the algorithm.<br>• Implement the algorithm using some programming language and test that the implementation works.<br>• Understand the behavior of the algorithm in the worst case and in the best case.<br>• Be able to follow and repeat the worst-case and best-case analysis of the algorithm. | • Explain how the Boyer-Moore algorithm for string searching works.<br>• Summarize the key properties of balanced trees.<br>• Illustrate how rotations in an AVL tree maintain the tree's balance.<br>• Explain the difference between a binary tree and a binary search tree.<br>• Formally define a red-black tree.<br>• Write the pseudocode for inserting items into a 2-3-4 tree.<br>• Write a program that sorts an array of 100 integers using shell sort.<br>• Explain why Quicksort is, in its worst case, a quadratic time algorithm. |
| 3 - Application | • Adapt a previously studied algorithm for some specific application, environment or specific representation of data.<br>• Construct the best-case and worst-case analysis of basic algorithms. | • Implement a program that sorts a linked list of strings using insertion sort and demonstrate that it works.<br>• Apply the DFS algorithm to check whether a graph is connected and analyze the complexity of the algorithm.<br>• Demonstrate the worst-case form of an AVL-tree, and calculate its height. |

Table 4: Sample Tasks for Bloom's Comprehension Levels 1-3

| Level in Bloom's Taxonomy | What Can The Learner Do At This Level | Sample Tasks And Assignments |
|---|---|---|
| 4 - Analysis | • Understands the relation of the algorithm with other algorithms solving the same or related problems.<br>• Understands the invariants in the algorithm code.<br>• Be able to reason, argue about and/or prove the correctness of the algorithm.<br>• Be able to analyze a complicated problem, identify essential objects in it and split the problem into manageable smaller problems. | • Categorize various tree structures.<br>• Compare the performance of Quicksort and Heapsort.<br>• Argue why Dijkstra's algorithm works.<br>• Explain why Prim's algorithm works for graphs containing edges with a negative weight but Dijkstra's algorithm does not.<br>• Analyze what kind of data structures and algorithms are needed in building a search engine. |
| 5 - Synthesis | • Design solutions to complex problems where several different data structures, algorithms and techniques are needed.<br>• Analyze the efficiency of complex combined structures.<br>• Set up criteria for comparing various solutions. | • Design a search engine and analyze its efficiency of space and time.<br>• Design the data structures and algorithms needed by a car navigation system.<br>• Create a test environment for assessing how various search structures perform in a hierarchical memory. |
| 6 - Evaluation | • Argue how and why some algorithm should be modified or combined with other algorithms to solve efficiently a new, more complex problem.<br>• Discuss the pros and cons of different algorithms that solve the same or similar problems.<br>• Carry out an evaluation of a design or analysis. | • Define appropriate criteria for assessing the applicability of search algorithms and argue why these criteria are important.<br>• Compare balanced trees and hashing as methods of implementing a dictionary.<br>• Discuss the design of a solution, and argue why it is better or worse than a different solution.<br>• Discuss how the analyses presented in the source text could be refined. |

Table 5: Sample Tasks for Bloom's Comprehension Levels 4-6

at least the first assignment or exam for evaluation may be a better indicator of initial participants. Another issue to consider in measuring drop-out rate is related to the institutional rules for taking the final exam. Depending on the rules of a given university, learners may have one or several options for taking the final exam. Viable checkpoints for measuring drop-out rate could be immediately after the first exam or after all exams have been completed. The definition of drop-out to be used in a specific instance must be determined as the experiment is designed.

3. Learning time

   Different learners need varying amounts of time to gain the same level of knowledge. Different learning methods can also affect the learning time. This can be important if the instructor wishes to cover more topics in the course. Thus, instead of setting up assignments with a set time limit and assessing the learner's improvement, the instructor could set up assignments with unlimited time and measure the time required for students to complete all assignments. If using visualization motivated learners to spend more time on task, this could be viewed as a positive result.

4. Learner satisfaction

   Learners have different motivations for taking a course. Additionally, their motivation can change during the course. It is therefore reasonable to ask for feedback to inform the educator what the learners think about the course. These questions may cover attitudes toward the subject itself, as well as the learners' opinions of various learning methods and tools applied during the course.

### 4.4 Covariant factors

There are additional factors that can affect the results of an experimental study. These factors describe the nature of the population under study. Such information is not a direct focus of the experiment, but can be gathered separately as background information that can help in analyzing the measured data.

1. Learning style

   Learners exhibit different learning styles. Several learning style models have been presented in the context of computing education, for example, the Felder-Silverman learning model [16, 17] and Kolb's learning model [36]. These models classify learners in different categories and dimensions. For example, the Felder-Silverman model defines four different dimensions of learning: visual/verbal, active/reflective, sensing/intuitive, and sequential/global. Each learner falls somewhere on the continuum in each dimension, and their position can considerably affect their learning when different learning methods and tools are used. In an evaluation study, learning style can blur the results. For example, visual learners could perform better using visualization tools whereas verbal learners could perform better without them. If both learning types are equally represented in the population that is being observed, the overall result may show no improvement on average.

   Learning style can be determined using simple questionnaires (for example, the Keirsey instrument [33]). However, learning style is not completely static, since properties that are less well developed in a given learner can be trained.

2. Learner's familiarity with using visualization technology

   If an experiment is conducted on how students learn one particular topic using visualization technology, the researcher should consider whether some students participating in the experiment have also used the visualization tool to explore other topics. The researcher would expect that previous familiarity with the tool would improve students' ability to learn with it. Hence the effectiveness of visualization in learning may well be highly dependent on how deeply the instructor has integrated the use of the visualization tool into a variety of activities in the course. Indeed, Ross [52] goes so far as to state that "even very good, active learning visualization software will be seldom used if it is a standalone system that is not integrated as part of the teaching and learning resources of a class."

3. Learning orientation

   Learners have different goals in their learning. Niemivirta [46] mentions several different learning orientations, including those with respect to achievement, performance-approach, performance-avoidance, and avoidance. Thus, some learners may have a goal of learning the topic, while the goal of others may be to just pass the course, or to achieve good grades or better grades than their fellow learners. Some learners, on the other hand, avoid situations that they feel are hard. All of these attitudes can considerably affect performance when different learning methods are being used.

   Learning orientation can be tested with simple questionnaires, but again the researcher must recognize that this factor need not be static. Learning orientation could change over the duration of a course if the learner's motivation for the topic changes considerably. In general, however, change of learning orientation occurs much more slowly than this.

4. Other background information

   There are several other factors that can be of interest in testing learners' improvement. These include learner's background, gender, age, curriculum, and so forth.

## 5 An Experimental Framework

This section describes a large general study on forms of engagement and their learning outcomes to be carried out over the next year. The study will involve the members of the Working Group and any other educators who would like to join in. In this section, we describe our hypotheses, propose a general framework for performing experiments, and provide several examples of experiments that would fit within our vision.

### 5.1 General Hypotheses

We make the following hypotheses based on the six forms of engagement described in Section 3. We have represented the forms of engagements and their possible overlaps in the Venn diagram of Figure 2. This diagram showed that viewing is included in the latter four categories and these latter four can overlap in many ways. The hypotheses can be related to the fundamental principles set forth in constructivist learning theory, which was mentioned in Section 2.3.

The hypotheses are the following:

**I. Viewing vs. No Viewing:** *Viewing* results in equivalent learning outcomes to no visualization (and thus *no viewing*).

Several studies [1, 11, 40] have shown that mere passive viewing provides no significant improvement over no visualization, but these studies were based on small sample populations. Our study may be able to verify this with larger numbers.

**II. Responding vs. Viewing:** *Responding* results in significantly better learning outcomes than *viewing* [7, 11, 15, 24, 30, 44].

**III. Changing vs. Responding:** *Changing* results in significantly better learning outcomes than *responding* [1, 18, 35, 45].

**IV. Constructing vs. Changing:** *Constructing* results in significantly better learning outcomes than *changing* [1, 24, 51, 55].

**V. Presenting vs. Constructing:** *Presenting* results in significantly better learning outcomes than *constructing*.

**VI. Multiple Engagements:** A mix of several forms of engagement is natural and we expect this to occur in experiments, especially in the latter types of engagement. This sixth hypothesis merely states "More is better." That is, the higher the level or the more forms of engagement that occur when using visualization, the better the learning becomes [1]. Figure 2 provided an indication of the variety of combinations that could occur in this regard.

### 5.2 A General Framework

This general framework is provided as a means of encouraging a level of consistency across all experiments conducted as a part of this study. These guidelines assume that the experimenter has selected a hypothesis for testing, an algorithm as the focal point, and a visualization tool that will support the form of engagement. The framework includes selection of participants, preparation of materials to use in performance of tasks, a procedure that defines the organization of the experiment, and a description of the evaluation instruments for data collection [32].

**Participants** Typically, participants will be drawn from a particular course. An instructor might consider the use of volunteers outside the classroom setting. On the one hand, a volunteer group is less desirable as it introduces a factor of self-selection. On the other hand, learners in a course may view participation in the study as affecting their grade in the course. The ethics of studies that involve human subjects is a matter of concern in most campus communities; participating researchers must consult the guidelines of the local Human Subjects Research Board and, if necessary, seek approval for the study.

Background information about the participants should be collected for data analysis purposes. (See the evaluation instruments section below.) The identity of individual learners may be made anonymous by use of ID-numbers.

**Materials and Tasks** The researcher must generate a list of learning objectives, keyed to Bloom's taxonomy (see Section 4.1), and prepare the instructional materials that will be used in the tasks. This comprises the pre-test and post-test, laboratory materials that include the visualization, and possible plans for classroom lectures, presentations, and discussions. The instructional materials and tasks carried out by the instructor and the learners will vary depending on the hypothesis being tested. This is necessarily the case due to the demands of varying levels of engagement.

As an example, suppose the topic area of the visualization experiment is Quicksort. In section 4.2, we suggested tasks at level 2 (comprehension) and level 3 (application) in Bloom's taxonomy. If the level of engagement is *Responding*, the materials can include the Quicksort visualization and a worksheet handout. The learner's task is to view the Quicksort visualization and then answer questions on the worksheet. Questions on the worksheet at level 2 (comprehension) of Bloom's taxonomy would be of the form "Given data set X, which item will be chosen as the pivot?" or "How many items are in a particular partition?"

As a second example, consider the Quicksort algorithm with combined levels of engagement, *Constructing* and *Changing*. The materials include a tool that supports learner construction of visualizations. The tasks could be: "(1) Con-

struct a visualization of the Quicksort algorithm so the resulting list of items is in ascending order; and (2) Change the data set for the Quicksort visualization to force worst case behavior."

**Procedure** The purpose of the procedure is to organize the experiment. The overall experimental procedure is rather straightforward: pre-test, use materials to perform tasks, post-test. This three-step sequence does not cover all aspects of using the visualization materials and collecting data. We leave the details of the procedure to the instructor to plan a best fit for their schedule and their learners.

To construct such an experiment, the best method for obtaining reliable results is to split the learners into two or more randomized groups, in which the distribution of the population under study is similar, according to the chosen covariants presented in Section 4.4. However, since such groupings are often not feasible, some other approach may be used, for example:

- If an institution has two sections of a course, each section can use one type of engagement and then the two can be compared.
- At some institutions where learners should have the same type of learning experience, one method can be used during the first half of the course in one section and not in the other section. Then the post-test is administered in both sections. After the post-test, when the experimental data has been collected, the section without the engagement can be given the same treatment to ensure equal opportunity for learning.
- If the same course is taught in two different semesters, then one engagement method could be used one semester and another engagement method could be used another semester.

**Evaluation Instruments** Several instruments are needed in order to carry out the experiment, including pre- and post-tests, a background questionnaire, task-specific logs, and learner feedback forms.

*Pre-Test and Post-test*: Care must be taken to ensure that the pre-test and post-test are isomorphic. This is most easily ensured by using the same questions in a different order and/or with different, but comparable, data sets. Items in the pre-test and post-test should be keyed to Bloom's taxonomy and the learner objectives already defined during the "Materials and Tasks" stage of planning. However, learners who are aware of such a pre- and post-test may prepare for the post-test by practicing the questions that were part of the pre-test, hence tainting the experiment.

*Background*: Expected items include: gender, year in school, major, standardized test scores. Other items are at the discretion of the instructor.

*Task-specific*: The instructor will manage the collection of data intended to measure the following:

- Time on task – categories could include the time learners spend on task in the classroom, in a closed lab, in an open lab, and on their own.
- Learner progress – scores and an item analysis of the pre-test and post-test. For example, in a *Constructing* vs. *Changing* experiment, three items on a pre-test and post-test might be keyed to a learner objective at level 3 (application) of Bloom's taxonomy. In order to assess the learners' ability to apply the material they have learned to new situations, there can be an item analysis of the pre- and post-tests. This will give an indication of the contribution of each level of engagement to whatever gain (or no gain) occurred.
- Drop-out rate – this may be more significant in the lower-level courses.

*Learner feedback*: A form will be used to collect information on the covariant factors defined in Section 4.4. Sample questions include the following.

1. How effective was the visualization in illustrating the concept?
2. What is the contribution of visualization to your understanding?
3. In what way has your attitude changed on this topic?

### 5.3 Example Experiments

In this section we suggest examples of possible experiments and how they could be interpreted with different forms of engagement.

We describe six topics from different areas of computer science and how they could be defined using different forms of engagement. Each example describes the forms of engagement that would be the independent or treatment variables of the experiments. The dependent or criterion variable in each example will be the pre-test and post-test that will be given to each group.

1. **Area:** Programming Languages
   **Topic:** Data Types in Functional Programming
   **Hypothesis I:** Viewing vs. No Viewing

   In this example, a tool is needed that allows one to run a program with functional data structures both in a textual and in a graphical display. For instance, breadth traversal involves lists and trees.

   *No viewing* breadth traversal could mean looking at the states of the algorithm on given data sets, where the states are given in textual format.

*Viewing* breadth traversal could mean looking at an animation of the algorithm on the same data sets, where the states of the animation are given in graphical format.

2. **Area:** CS-1

   **Topic:** Quicksort

   **Hypothesis II:** Responding vs. Viewing

   In this example, a tool is needed that allows one to view the animation of Quicksort.

   *Viewing* could mean looking at an animation of the algorithm on given data sets. The animation may or may not have controls associated with it such as pausing and stepping through the phases. The animation could be viewed with given data sets that illustrate the worst case and average case.

   *Responding* could mean viewing Quicksort with either prediction built into the software or a worksheet containing questions that learners must answer while stepping through the animation. Learners must answer questions such as "Which element will be chosen as the next pivot? What will the array look like after the call to find the next pivot? Which section of the array will be sent in the next call of recursion? Which section of the array at this point in time is guaranteed to be in sorted order?"

   Concepts to focus on are the understanding of the overall algorithm, understanding the recursion part of the algorithm, and understanding the choice of pivot and the algorithm for the rearrangement of the data around the pivot.

3. **Area:** CS-2

   **Topic:** Tree Traversal

   **Hypothesis II:** Changing vs. Responding

   In this example, a tool is needed that allows one to load a tree and see the animations of the tree traversals pre-order, in-order and post-order.

   *Responding* could mean watching animations of tree traversals pre-order, in-order, and post-order on given trees with either prediction built into the software or a worksheet that learners must answer during the stepping-through of the animation. Learners must answer questions such as "Which node is printed out next? When will this node be visited? How many times is this node visited before it is printed?"

   *Changing* could mean changing the underlying data, which in this case is the tree. Learners can change the tree and then see what the different results are for different trees.

4. **Area:** Automata Theory

   **Topic:** Converting an NFA into a DFA

   **Hypothesis IV:** Constructing vs. Changing

   In this example, a tool is needed that allows one to load or construct a nondeterministic finite automaton (NFA) and animate the process of converting the NFA into a deterministic finite automaton (DFA).

   *Changing* could mean changing the underlying NFA and then following steps to see how the different NFA's are converted into DFA's.

   *Constructing* could mean starting with an NFA and then constructing the equivalent DFA using software that will give helpful error messages if mistakes are made.

5. **Area:** Algorithmics and Data Structures

   **Topic:** Shortest Paths

   **Hypothesis V:** Presenting vs. Constructing

   In this example, a compiler is needed, along with a tool for producing an animation, such as a scripting language.

   *Constructing* could mean writing program code and the animation for the shortest paths algorithm. Both of these are time-consuming. The learner is likely given pseudocode or general code for the algorithm and must first adapt it, and then add the components to produce the animation using a scripting language.

   *Presenting* could mean the learner would present the shortest path algorithm in detail in front of the class, possibly using a presentation software.

   Note: the *Presenting* example above for the shortest paths algorithm is likely to include additional forms of engagement such as *Constructing*, *Changing*, or *Viewing* in order to learn the algorithm before presenting it.

6. **Area:** Introduction to Programming

   **Topic:** Recursion

   **Hypothesis VI:** Changing vs Viewing

   In this example, a program visualization tool is needed. A program visualization tool allows the learner to construct a program and then automatically generates an animation. For instance, the *Alice* 3D animation tool (`http://www.alice.org`) provides a programming language environment where learners can immediately see an animation of how their program executes [14].

   *Viewing* could mean watching an animation where a skater (or some figure) is skating to a cone on the ice, avoiding collision. The condition of nearness to the cone is used to control a recursive call to the animation method (tail recursion).

   *Constructing and Changing* could mean the learner constructs the animation using recursive calls to glide a skater to the cone without colliding. Then, the learner is asked to change the animation to make the skater respond to a mouse click on different cones, skating to the selected cone.

## 6 Conclusion

In this report, we have set the stage for a wide variety of future studies that will allow computer science educators to

measure the relationship between a learner's form of engagement with a visualization and the types of understanding that are affected by that engagement. We have defined an engagement taxonomy to facilitate a consistent approach toward defining the form of engagement used in such studies. We have also described how Bloom's taxonomy can be used to differentiate among types of understanding in various areas of computer science. Based on these taxonomies, we have specified a framework for conducting experiments that use these two taxonomies to establish the independent and dependent variables respectively.

In the coming year, we intend to design several specific experiments that are based on this methodology. These experiments will be broad enough to allow collaboration between researchers at many institutions. We invite educators who are interested in participating to contact either of the working group co-chairs at `naps@uwosh.edu` or `roessling@acm.org`.

## 7 Acknowledgments

We gratefully acknowledge the contributions of the following colleagues:

- Jay Anderson, Franklin and Marshall, USA
- Scott Grissom, Grand Valley State University, USA
- Rocky Ross, University of Montana, USA

Jay and Rocky had hoped to be part of the group, but events kept them from coming to Denmark. They nonetheless provided substantial input to our deliberations through electronic communication. Scott's survey at ITiCSE 2000 was a key factor in formulating the group's original goals.

We also thank all the CS educators who responded to our pre-conference on-line survey as well as those who took part in the index card survey during ITiCSE 2002.

## References

[1] Anderson, J. M., and Naps, T. L. A Context for the Assessment of Algorithm Visualization System as Pedagogical Tools. *First International Program Visualization Workshop, Porvoo, Finland. University of Joensuu Press* (July 2001), 121–130.

[2] Baecker, R. *Sorting Out Sorting*: A Case Study of Software Visualization for Teaching Computer Science. In *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1998, ch. 24, pp. 369–381.

[3] Bazik, J., Tamassia, R., Reiss, S. P., and van Dam, A. Software Visualization in Teaching at Brown University. In *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1998, ch. 25, pp. 382–398.

[4] Biermann, H., and Cole, R. Comic Strips for Algorithm Visualization. Tech. rep., NYU 1999-778, New York University, Feb. 1999.

[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., Eneboe, T. J., Goosey, F. W., Ross, J. A., and Ross, R. J. Dancing with Dynalab, Endearing the Science of Computing to Students. *Twenty-seventh SIGCSE Technical Symposium on Computer Science Education* (1996), 135–139.

[7] Bridgeman, S., Goodrich, M. T., Kobourov, S. G., and Tamassia, R. PILOT: An Interactive Tool for Learning and Grading. 31$^{st}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2000), Austin, Texas* (Mar. 2000), 139–143.

[8] Brown, M. H. *Algorithm Animation*. MIT Press, Cambridge, Massachussets, 1988.

[9] Brown, M. H., and Raisamo, R. JCAT: Collaborative Active Textbooks Using Java. *Computer Networks and ISDN Systems 29* (1997), 1577–1586.

[10] Brown, M. H., and Sedgewick, R. A System for Algorithm Animation Structures. *ACM SIGGRAPH '84 Proceedings, Minneapolis, Minnesota* (July 1984), 177–186.

[11] Byrne, M. D., Catrambone, R., and Stasko, J. Evaluating Animations as Student Aids in Learning Computer Algorithms. *Computers & Education 33* (1996), 253–278.

[12] Citrin, W., and Gurka, J. A Low-Overhead Technique for Dynamic Blackboarding Using Morphing Technology. *Computers & Education 26*, 4 (1996), 189–196.

[13] Crosby, M. E., and Stelovsky, J. From Multimedia Instruction to Multimedia Evaluation. *Journal of Educational Multimedia and Hypermedia 4* (1995), 147–162.

[14] Dann, W., Cooper, S., and Pausch, R. Using Visualization To Teach Novices Recursion. 6$^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2001), Canterbury, England* (June 2001), 109–112.

[15] Faltin, N. Structure and Constraints in Interactive Exploratory Algorithm Learning. In *Software Visualization* (2002), S. Diehl, Ed., no. 2269 in Lecture Notes in Computer Science, Springer, pp. 213–226.

[16] Felder, R. M. Reaching the second tier. *Journal of College Science Teaching 23*, 5 (1993), 286–290.

[17] Felder, R. M. Matters of style. *ASEE Prism 6*, 4 (1996), 18–23.

[18] Gloor, P. A. User Interface Issues for Algorithm Animation. In *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1998, ch. 11, pp. 145–152.

[19] Grissom, S. Personal communication, 2002.

[20] Gurka, J. S. *Pedagogic Aspects of Algorithm Animation*. PhD thesis, Department of Computer Science, University of Colorado, 1996.

[21] Haajanen, J., Pesonius, M., Sutinen, E., Tarhio, J., Teräsvirta, T., and Vanninen, P. Animation of User Algorithms on the Web. *IEEE Symposium on Visual Languages* (1997), 360–367.

[22] Hansen, S., Schrimpsher, D., and Narayanan, N. H. From Algorithm Animations to Animation-Embedded Hypermedia Visualizations. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 1999), Seattle, Washington* (1999), 1032–1037.

[23] 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).

[24] Hundhausen, C. D. *Toward Effective Algorithm Visualization Artifacts: Designing for Participation and Communication in an Undergraduate Algorithms Course*. PhD thesis, University of Oregon, 1999. Unpublished Doctoral Dissertation, available as technical report CIS-TR-99-07 (June 1999) in Department of Computer and Information Science, University of Oregon, Eugene.

[25] Hundhausen, C. D. Integrating Algorithm Visualization Technology into an Undergraduate Algorithms Course: Ethnographic Studies of a Social Constructivist Approach. *Computers & Education* (2002), (in print).

[26] Hundhausen, C. D., and Douglas, S. Using Visualizations to Learn Algorithms: Should Students Construct Their Own, or View an Expert's? *IEEE Symposium on Visual Languages, Los Alamitos, California* (2000), 21–28.

[27] Hundhausen, C. D., and Douglas, S. A. Low-Fidelity Algorithm Visualization. *Journal of Visual Languages and Computing* (2002), (in print).

[28] Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing* (2002), (in print).

[29] Hung, T., and Rodger, S. H. Increasing Visualization and Interaction in the Automata Theory Course. $31^{st}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2000), Austin, Texas* (Mar. 2000), 6–10.

[30] Jarc, D., Feldman, M. B., and Heller, R. S. Assessing the Benefits of Interactive Prediction Using Web-based Algorithm Animation Courseware. $31^{st}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2000), Austin, Texas* (Mar. 2000), 377–381.

[31] Kann, C., Lindeman, R. W., and Heller, R. Integrating Algorithm Animation into a Learning Environment. *Computers & Education 28* (1997), 223–228.

[32] Katz, B., and Almstrum, V. Collaborative Project Plans, version 1.3. WWW: `http://www.cs.utexas.edu/users/csed/CPP/`, Nov. 3, 1998.

[33] Keirsey, D. M. Keirsey Temperament and Character Web Site. WWW: `http://www.keirsey.com`, 2002.

[34] Khuri, S. Designing Effective Algorithm Visualizations. *First International Program Visualization Workshop, Porvoo, Finland. University of Joensuu Press* (Feb. 2001), 1–12.

[35] Khuri, S. A User-Centered Approach for Designing Algorithm Visualizations. *Informatik / Informatique, Special Issue on Visualization of Software* (Apr. 2001), 12–16.

[36] Kolb, D. *Experiential Learning*. Prentice-Hall, New Jersey, 1984.

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

[38] Korhonen, A., and Malmi, L. Matrix - Concept Animation and Algorithm Simulation System. *Proceedings of the Working Conference on Advanced Visual Interface (AVI 2002), Trento, Italy* (May 2002), 256–262.

[39] Korhonen, A., Sutinen, E., and Tarhio, J. Understanding Algorithms by Means of Visualized Path Testing. In *Software Visualization* (2002), S. Diehl, Ed., no. 2269 in Lecture Notes in Computer Science, Springer, pp. 256–268.

[40] Lawrence, A. W. *Empirical Studies of the Value of Algorithm Animation in Algorithm Understanding*. PhD thesis, Department of Computer Science, Georgia Institute of Technology, 1993.

[41] Mayer, E., and Anderson, R. B. Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of Educational Psychology 83* (1991), 484–490.

[42] Naharro-Berrocal, F., Pareja-Flores, C., Urquiza-Fuentes, J., Velázquez-Iturbide, J. A., and Gortázar-Bellas, F. Redesigning the Animation Capabilities of a Functional Programming Environment under an Educational Framework. *Second International Program Visualization Workshop, Århus, Denmark* (June 2002), (in print).

[43] Naharro-Berrocal, F., Pareja-Flores, C., and Velázquez-Iturbide, J. A. Automatic Generation of Algorithm Animations in a Programming Environment. $30^{th}$ *ASEE/IEEE Frontiers in Education Conference, Kansas City, Missouri* (Oct. 2000), S2C 6–12.

[44] Naps, T., Eagan, J., and Norton, L. JHAVÉ: An Environment to Actively Engage Students in Web-based Algorithm Visualizations. $31^{st}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2000), Austin, Texas* (Mar. 2000), 109–113.

[45] Naps, T. L. Incorporating Algorithm Visualization into Educational Theory: A Challenge for the Future. *Informatik / Informatique, Special Issue on Visualization of Software* (Apr. 2001), 17–21.

[46] Niemivirta, M. Motivation and performance in context - the influence of goal orientation and instructional setting on situational appraisals and task performance. *International Journal of Psychology in the Orient* (2002), (in print).

[47] Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.

[48] Pierson, W., and Rodger, S. H. Web-based Animation of Data Structures Using JAWAA. $29^{th}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '98), Atlanta, Georgia* (1998), 267–271.

[49] 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.

[50] Price, B. A., Baecker, R. M., and Small, I. S. A Principled Taxonomy of Software Visualization. *Journal of Visual Languages and Computing 4*, 3 (1993), 211–264.

[51] Rodger, S. Integrating Animations into Courses. $1^{st}$ *Annual ACM SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education (ITiCSE '96), Barcelona, Spain* (June 1996), 72–74.

[52] Ross, R. J. Personal communication, 2002.

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

[54] Stasko, J. TANGO: A Framework and System for Algorithm Animation. *IEEE Computer 23* (1990), 27–39.

[55] Stasko, J. Using Student-built Algorithm Animations as Learning Aids. $28^{th}$ *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '97), San Jose, California* (Feb. 1997), 25–29.

[56] Stasko, J., Badre, A., and Lewis, C. Do Algorithm Animations Assist Learning? An Empirical Study and Analysis. *Proceedings of ACM INTERCHI 1993 Conference on Human Factors in Computing Systems* (1993), 61–66.

[57] Stern, L., Søndergaard, H., and Naish, L. A Strategy for Managing Content Complexity in Algorithm Animation. $4^{th}$ *Annual ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE'99), Cracow, Poland* (Sept. 1999), 127–130.