Exploring Cognitive, Social, and Cultural Dimensions of Visualization in Computer Science Education

Computer-based visualization technology graphically represents scientific processes and concepts. A key application area of the technology is science education, where it has been used to show students how abstract processes and physical systems work. While science educators were initially enthusiastic about the potential of computer-based visualization to enhance learning, experimental studies of its educational effectiveness have yielded widely mixed results, and it has failed to see the kind of widespread use in science education that its developers envisioned.

It is possible that the failure of computer-based visualization to realize its potential in science education can be traced to its grounding in a learning theory that is fundamentally deficient. Focusing on knowledge at the level of the individual, that theory holds that visualization technology is pedagogically effective because it provides a faithful account of an expert's mental model of a process or concept to be learned, thus enabling the robust and efficient transfer of that mental model to the learner. In contrast, a promising alternative, Sociocultural Constructivism, views learning at the level of the community. On this alternative view, visualization is seen to be pedagogically effective insofar as it enables learners to participate more centrally in the practices of a community.

How might visualization technology facilitate such participation? Within the context of a third-year, undergraduate computer science course on computer algorithms, I have been exploring an approach in which students become teachers by using algorithm visualization technology not only to construct their own visualizations, but also to present those visualizations to their instructor and peers for feedback and discussion. Inspired by the instructional model used in architecture, I propose to take this approach further by developing a *studio-based* algorithms course in which the construction and discussion of visualizations are the central activities of the course. Specifically, students will use algorithm visualization technology to construct their own visual solutions to algorithm design and analysis problems. In a variety of regularly scheduled review sessions, they will present their solutions to instructors and peers for feedback, discussion, and evaluation.

This research will yield several products and results from which science educators, educational researchers, visualization technologists, and even cognitive anthropologists stand to benefit. Specific outcomes and benefits of the proposed research include the following:

- *Technology delivery*. I shall develop and publicly disseminate novel software, specifically designed for electronic whiteboards, that supports the interactive construction and discussion of visualizations in a studio environment. By employing a sound, user-centered design process, and by collaborating with a community of partners within an open source environment, I will produce a robust, easy-to-use, well-documented software system that other educators can adapt for use in their own courses.
- *Curriculum development*. I shall develop and disseminate a detailed syllabus and curricular materials to support a studio-based algorithms course. These materials will be iteratively tested and refined through actual pilot courses. The studio-based curriculum will make a notoriously difficult course more accessible to students, thus enabling more students to learn the material and ultimately complete their computer science degrees.
- *Empirical results*. I shall publish the results of a diverse program of experimental, observational, and ethnographic studies that explore the value and role of visualization and communication in learning complex scientific processes at the cognitive, social, and cultural levels. These studies will build on a rich tradition of empirical work into both computer-based instructional visualization, and the use of representations in computer-supported collaborative learning.
- *Novel research methodology*. I shall develop a novel methodology for using visualization construction and interpretation tasks to gauge one's level of membership in a community of practice. This methodology, which builds on Cultural Consensus Theory (a product of cognitive anthropology), will be used as the basis for "consensus studies" that evaluate learning with respect to the Sociocultural Constructivist definition of increasingly central community membership.
- *Graduate and undergraduate apprenticeship*. I shall apprentice two graduate students and at least one undergraduate student per year in the methods of human-computer interaction, including user-centered software design, video analysis, ethnographic field techniques, and experimental methods. Dozens of other students will benefit through their participation in the proposed studio-based algorithms courses and empirical studies. In providing such training and experience, I will be furthering the educational mission of my department.

Exploring Cognitive, Social, and Cultural Dimensions of Visualization in Computer Science Education

Introduction

Computer-based visualization technology graphically represents scientific processes and concepts. A key application area of the technology is science education, where it has been used, for example, to illustrate embryonic cell processes (Pane, Corbett, & John, 1996); macroeconomic policy (Grimes & Willey, 1990); data structure operations (Stasko, Badre, & Lewis, 1993); Newton's laws of motion (Rieber, 1990); and pump dynamics (Mayer & Anderson, 1992). While science educators were initially enthusiastic about the potential of computer-based visualization to enhance learning, it has failed to see the kind of widespread use in science education that its developers originally envisioned (see, e.g., Iding, Crosby, & Speitel, 2001).

In light of the strong intuitive basis for computer-based visualization, a puzzling research question arises:

Research Question:	*Why has computer-based visualization failed to catch on?*

While no extant empirical research specifically addresses that question, educational researchers have cited four key obstacles to visualization technology's widespread adoption: (a) the technological infrastructure required to deploy visualization technology may not be readily available (see, e.g. Gurka & Citrin, 1996); (b) the quality and usability of visualization technology may not be sufficiently high; (c) creating visualizations for classroom use, or integrating existing visualizations into a course, requires substantial time and effort, (see, e.g., Bazik, Tamassia, Reiss, & van Dam, 1998); and (d) the pedagogical value of using computer-based visualization has not been empirically substantiated (see, e.g., Byrne, Catrambone, & Stasko, 1999; Pane et al., 1996).

Of course, speculating about these obstacles is all in the interest of addressing a second, closely-related research question:

Research Question:	*Can we develop visualization software and pedagogical approaches that overcome these obstacles, allowing visualization technology to become an effective pedagogical tool in mainstream science education?*

Over the past decade, educators and technologists have taken various approaches to addressing this question, such as developing improved visualization technology (e.g., Stasko, 1997), novel pedagogical methods (e.g., Knox, 1996), and more sensitive evaluation methods (e.g., Gurka, 1996)But it is possible that the underlying educational theory that has traditionally guided these approaches is fundamentally deficient. That theory, which I call Epistemic Fidelity Theory (after Roschelle, 1994) holds that visualization technology is pedagogically effective because it provides a faithful account (i.e., one with high epistemic fidelity) of an expert's mental model of a scientific process or concept to be learned, thus enabling the robust and efficient transfer of that mental model to the learner.

An alternative approach is to address the problem not at the surface, but at its roots. In other words, instead of refining our current design, pedagogy, and evaluation methods, we rethink the theory of effectiveness in which they are rooted. A promising alternative theoretical position is Sociocultural Constructivism (see esp. Lave & Wenger, 1991). Rather than viewing knowledge and learning at the level of the individual, as Epistemic Fidelity Theory does, Sociocultural Constructivism views knowledge and learning at the level of the *community of practice* (Wenger, 1998). On this alternative view, learning is seen not as acquiring target knowledge structures, but rather as participating more centrally in the practices of a community.

In prior work within the context of computer science education, I have explored this theoretical position through ethnographic studies of an undergraduate, junior-level algorithms course in which algorithm visualization (AV) technology was used to provide students with access to more central forms of participation. Specifically, for an assignment in the course, students were required to construct and present their own visualizations of the algorithms under study. Thus, students participated in the course in ways that instructors typically participate. A key observation in these studies was that having students construct and present their own visualizations was educationally beneficial, not only because it increased their motivation and level of interest in algorithms, but also because it stimulated meaningful discussions about algorithms in which students and their instructor could bridge the gap in their understandings.

While these ethnographic studies illustrate the potential for this alternative teaching approach, they also raise key research questions that I propose to pursue here. The first question has to do with pushing the approach further: What if we fundamentally changed computer science education's traditional, lecture-based instructional model by making visualization construction and presentation the central activities of a computer algorithms course? Inspired by the instructional model presently used to teach architectural design (see, e.g., Boyer & Mitgang, 1996), I propose to explore this question by developing a curriculum and supporting technology for a *studio-based* algorithms course. Architectural students spend most of their time with their peers in an architecture studio, where they collaborate on design projects. In scheduled review sessions, they present their projects their instructor and peers for feedback and discussion. So, too, will it be in a studio-based algorithms course. In the "algorithms studio," students will engage in collaborative algorithm design and analysis projects at their "drafting tables"—electronic whiteboards running an algorithm visualization tool specifically designed to support the student construction and presentation of visualizations. Students will present their solutions (i.e., their visualizations) to their peers and instructors for feedback and discussion during scheduled review sessions.

Development of the studio-based algorithms curriculum and supporting visualization technology constitutes the first key component of the research I am proposing here. It gives rise to fundamental research questions concerning the role and value of visualization construction and discussion in learning abstract processes and concepts. In articulating these questions, I find it useful to distinguish three conceptual levels of learning:

1. *Cognitive*. At the cognitive level, learning is an individual phenomenon in which students' knowledge undergoes conceptual change. This is the level at which learning outcomes are traditionally measured through some sort of knowledge test. The key question at this level is, "Do students who participate in visualization construction and presentation exercises actually learn algorithms better than students who participate in more traditional learning activities (e.g., lectures, individual study)?"
2. *Social*. At the social level, learning is the collaborative achievement of students; visualizations serve as *mediational resources* (Roschelle, 1994) that help students to build a mutual understanding of the phenomenon under study. The key question at this level is, "In what ways, and to what extent, do visualizations mediate conversations that (a) are relevant to algorithms, and (b) enable students and instructors to negotiate shared understandings of algorithms?"
3. *Cultural*. At the cultural level, learning fundamentally entails participating more centrally in a community of practice (Lave & Wenger, 1991); it is the process by which a community reproduces itself. At this level, the key question is, "Do visualization construction and presentation exercises enable students to become fuller members of the community being reproduced through an undergraduate algorithms course, and do they facilitate learning in this sense better than more traditional learning activities?"

The second key component of this work, then, is a comprehensive research program to explore these questions.

This research will yield several products and results from which computer science educators, educational researchers, visualization technologists, and even cognitive anthropologists stand to benefit. Specific outcomes and benefits of the proposed research include the following:

- *Technology delivery*. I shall develop and publicly disseminate novel AV software, specifically designed for electronic whiteboards, that supports the interactive construction and discussion of visualizations in a studio environment. By employing a sound, user-centered design process, and by collaborating with a community of partners within an open source environment, I shall produce a robust, well-documented system that other educators can adapt for use in their own courses.
- *Curriculum development*. I shall develop and disseminate a detailed syllabus and curricular materials to support a studio-based algorithms course. These materials will be iteratively tested and refined through actual pilot courses. The studio-based curriculum will make a notoriously difficult course more accessible to students, thus enabling more students to learn the material and ultimately complete their computer science degrees.
- *Empirical results*. I shall publish the results of a diverse program of experimental, observational, and ethnographic studies that explore the value and role of visualization and communication in learning complex scientific processes at the cognitive, social, and cultural levels. These studies will build and expand on a rich tradition of empirical work on both computer-based instructional visualization, and the use of representations in computer-supported collaborative learning.

- *Novel research methodology*. I shall develop a novel methodology for using visualization construction and interpretation tasks to gauge one's level of membership in a community of practice. This methodology, which builds on Cultural Consensus Theory (a product of cognitive anthropology), will be used as the basis for "consensus studies" that evaluate learning with respect to the Sociocultural Constructivist definition of increasingly central community membership.
- *Graduate and undergraduate apprenticeship*. I shall apprentice two graduate students and at least one undergraduate student per year in the methods of human-computer interaction, including user-centered software design, video analysis, ethnographic field techniques, and experimental methods. Dozens of other students will benefit through their participation in the proposed studio-based algorithms courses and empirical studies. In providing such training and experience, I will be furthering the educational mission of my department.

Background Research

The research program proposed here draws on over a decade of past research into the design, evaluation, and pedagogical use of AV technology. In this section, I develop a logical progression from this existing research to the research I am proposing. I begin with a brief history of AV technology that illustrates the ways in which its design and pedagogical use have been beholden to Epistemic Fidelity Theory. Next, I review a legacy of experimental studies that have aimed to substantiate its educational effectiveness. This review makes a strong case for the inadequacy of Epistemic Fidelity as a guiding theory of effectiveness, and suggests the plausibility of Sociocultural Constructivism as an alternative guiding theory. Finally, I describe my own prior research that explores a Sociocultural Constructivist approach to integrating AV technology into an algorithms course (Hundhausen, 1999).

Algorithm Visualization Technology

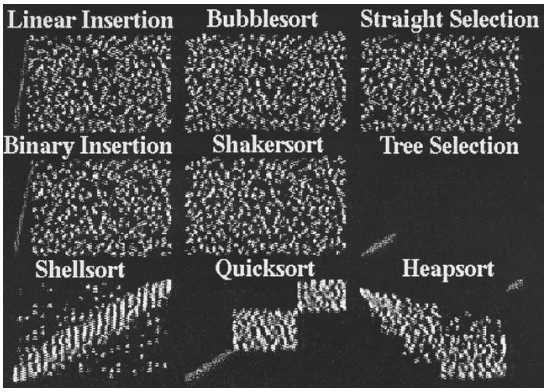
Ever since the publication of the first volumes of Knuth's seminal series *The Art of Computer Programming* (Knuth, 1973), graphical illustrations of algorithms have been commonplace as visual aids in computer science literature. With the advent of computer graphics technology in the late 1970s and early 1980s illustrations moved from paper to computer screen, as computer scientists began to develop computer software to facilitate the creation of *algorithm visualizations*—animated illustrations of algorithms in action. The first software aided in the production of algorithm movies (Baecker, 1975)—most notably, *Sorting Out Sorting* (Baecker, 1981), a legendary instructional film on sorting algorithms (see Figure 1a).

By utilizing emerging graphical workstation technology, later systems supported *interactive* environments for exploring algorithm pictures and animations. For example, the seminal interactive algorithm animation system, BALSA (Brown, 1988), defined an environment in which users could (a) select sets of input data on which to view an algorithm animation; (b) choose an arrangement of alternative views defined for the animation; (c) start and stop the animation, and control the animation's execution speed; (d) zoom and pan an animation view; and (e) write an animation viewing session to a script for later use. Figure 1b presents a snapshot from an interactive session with BALSA.

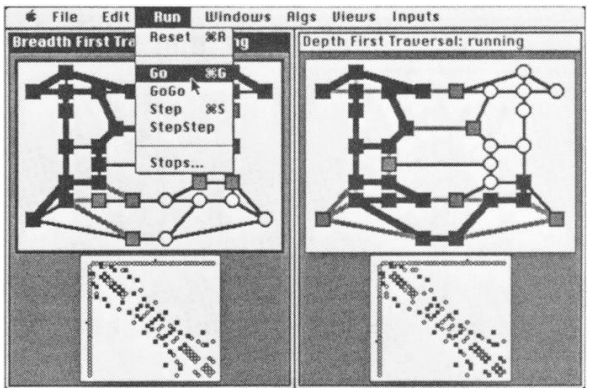
According to the conservative estimate of Price, Baecker, and Small (1993), over 100 algorithm visualization systems have been developed (see, e.g., Duisberg, 1986; Helttula, Hyrskykari, & Raiha, 1989; Roman, Cox, Wilcox, & Plun, 1992). While, on the surface, these systems appear diverse, their design has been guided by the same underlying theory: Epistemic Fidelity. In particular, Epistemic Fidelity Theory's view of visualization as an effective mechanism for knowledge transfer is embodied in two specific features pioneered in Brown's BALSA system, and widely-embraced by AV technology ever since. First, observe that the user model on which AV technology is based consists of two distinct conceptual actors: *client programmers* (experts), who, through sophisticated graphics programming, create the visualizations; and end-users (learners), who view and interact with those visualizations. This user model is clearly beholden to Epistemic Fidelity Theory because it sets up a situation in which experts *encode* algorithm knowledge, which learners subsequently decode. Thus, under this model, visualizations are seen to play the role of *knowledge conveyors* by transferring knowledge from experts to learners.

Second, existing AV technology goes to great lengths to support the production of visualizations with *algorithmic fidelity*—that is, visualizations that faithfully depict the execution of the underlying algorithm. AV technology commonly supports algorithmic fidelity through three specific design features:

- *Direct generation*. With most extant AV technology, one creates a visualization by mapping an algorithm to a graphical representation. One does this by, for example, annotating its code with calls to graphics routines that



(a) The "Grand Race" in *Sorting Out Sorting*[1]



(b) An session with BALSA[2]

Figure 1. Snapshots of Representative AV Technology

produce and update the visualization (see, e.g., Brown, 1988; see Roman & Cox, 1993 for a review of mapping techniques). Thus, the visualization "is constructed automatically as a by-product of [the algorithm's] execution", and is therefore "guaranteed to portray [its] execution faithfully" (Baecker, 1998, p. 369).

- *Input generality*. The generation of visualizations directly from implemented computer algorithms implies that visualizations, just like the algorithms on which they are based, *must operate on arbitrary input*. AV technology's support for input generality is widely regarded as a key feature of the technology; several systems support graphical user interfaces that make it easy for end-users to specify input data interactively (see, e.g., Brown, 1988; Brown & Hershberger, 1991; Duisberg, 1987)
- *Typeset fidelity*. Nearly all extant AV technology requires that visualizations be programmed in terms of Cartesian coordinates. This leads to the kinds of high-quality drawings that one would expect to find in an algorithms textbook. Notice that such visualizations, which have high typeset fidelity, are "always accurate, even the ones which would tax the best of draftsmen" (Brown, 1988, p. 5).

In sum, high-powered computer graphics have paved the way for a legacy of over 100 AV systems, which allow one to construct and interactively explore visualizations of computer algorithms. Through their commitments to (a) a user model with two distinct conceptual actors, (b) direct generation, (c) input generality, and (d) typeset fidelity, these systems are loyal to a knowledge transfer model of effectiveness that I am calling Epistemic Fidelity Theory.

Experimental Studies of Effectiveness

Given the influence of Epistemic Fidelity Theory on AV technology design, an important question arises: To what extent is this theory supported by empirical evidence? To answer that question, one need only examine a substantial corpus of 22 experimental studies of AV technology.[3] All of these studies employed a similar between-subjects design. A sample of computer science students was divided into two treatment groups, both of which were charged with the task of learning the same target algorithm. At least one of the treatment groups used some configuration of AV technology to learn the target algorithm; the other group either used an alternative configuration of AV technology, or alternative materials (e.g., a textbook). At the end of the learning session, the two groups were given an identical post-test of algorithm knowledge. Finally, scores of the opposing treatments were tested for statistically significant differences, in order to determine whether one treatment was more effective than the other.

[1]From *Software visualization* (p. 379), edited by J. Stasko, J. Domingue, M. Brown, & B. Price, Cambridge, MA: The MIT Press. ©1998 The MIT Press. Reprinted with permission. The grand race graphically compares nine alternative sorting algorithms operating on an identical 2500-element data set. Data elements to be sorted are represented as scatterplots of dots—one scatterplot for each sorting algorithm. As each respective sorting technique places elements in place, the corresponding scatterplot gradually transforms into an upward-sloping line

[2]From *Algorithm animation* (p. 65), by M. Brown, Cambridge, MA: The MIT Press. ©1988 The MIT Press. Reprinted with permission. In this snapshot, the user is running a side-by-side comparison of breadth-first and depth-first search executing on an identical graph. The user has chosen to view each algorithm in terms of two identical views. The larger, top views depict the graph being traversed; vertices that have been visited are shaded black, vertices whose descendents have not been fully explored are shaded gray, and vertices that have not yet been visited are shaded white. The animations are presently paused; the user is about to restart them by choosing "Go" from the "Run" menu.

[3] For a comprehensive review of this corpus of studies, see (Hundhausen, Douglas, & Stasko, 2001).

Thirteen of the 22 studies found a statistically significant difference between the performance of a group of students using some configuration of AV technology, and another group of students using either an alternative configuration of AV technology, or no AV technology at all. For example, Crosby and Stelovsky (1995) found that students who interacted with an algorithm animation performed significantly better than students who listened to a lecture. Likewise, Hansen, Schrimpsher, and Narayanan (2000) found that students who learned an algorithm using their HalVis hypermedia environment, which provides its viewers with multiple views and engages them in input data set design, interactive prediction, and question-answering, significantly outperformed students who learned the same algorithm using (a) textual materials (Study I), (b) a lecture (Study II), or (c) a conventional, one-view algorithm animation environment with no facilities for interactive prediction or question-answering (Study V).

In contrast, nine studies did not have a significant result. For instance, Stasko, Badre, and Lewis (1993) had two groups of students learn the pairing heap data structure (a) by interacting with an algorithm animation, and (b) by studying textual materials. Although the animation group performed better on a post-test, the difference was not statistically significant. Likewise, Lawrence (1993, Ch. 4 & 5) compared the post-test performance of students who learned algorithms using animations with alternative representational characteristics: sticks versus dots; 9 vs. 24 vs. 41 data elements; and labels vs. no labels. She found no significant differences among the groups.

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 AV. Lawrence (1993, Ch. 4, 5, 7, 8), Stasko, Badre, and Lewis (1993), Gurka (1996), and Hansen et al. (2000, VI – VIII) varied *representational characteristics* of the learning materials, including (a) text versus animation, (b) text-first or animation-first, and (c) various graphical attributes of animations. By contrast, a second group of studies (Byrne et al., 1999, §2 and 3; Hansen et al., 2000, I – V; Jarc, Feldman, & Heller, 2000; Kann, Lindeman, & Heller, 1997; Lawrence, 1993, ch. 6 and 9) varied level of *learner involvement*. In addition to having learners passively view an animation, these studies involved learners more actively by having them (a) construct their own data sets, (b) answer strategically-chosen questions about the algorithm; (c) make predictions regarding future algorithm behavior, or (d) program the algorithm.

If one considers the results of the experiments vis-à-vis these two categories (see Figure 2), a notable trend emerges: experiments that manipulate learners' level of involvement have consistently yielded significant results, whereas experiments that manipulate representation have not. This suggests that what learners do, not what they see, may have the greatest impact on learning—an observation that casts doubt on Epistemic Fidelity Theory's knowledge transfer view of effectiveness.

Low Fidelity Algorithm Visualization

In the previous two subsections, I reviewed two lines of work that motivate the research program I have been pursuing over the past five years under the general title of "Low Fidelity Algorithm Visualization." (Hundhausen, 1999; see also Hundhausen, 2001; Hundhausen & Douglas, 2001). I now turn to a review of that research.

Given the importance of student involvement in AV-based learning, it makes sense to explore learning theories that account for the benefits of AV technology in terms of active learning. An obvious candidate is Constructivist learning theory (see, e.g., Fosnot, 1996), which views learning as a process in which one constructs one's own understandings through active engagement with one's environment. In the case of AV-based learning, promoting such engagement means getting students maximally involved in the process of visualizing an algorithm.

To that end, Stasko (1997) advocates the use of "visualization assignments" in which students use AV software to construct their own visualizations of the algorithms under study. While such assignments enable students to actively construct their own understandings of algorithms, I believe that they could go further. To be sure, the individualistic understandings they foster are part of the competence that an algorithms student would ideally develop. However, on closer inspection of the learning objectives of an algorithms course, I believe that "knowing" algorithms encompasses much more than cognitive skills. Rather, I suggest that the "schoolbook algorithms" curriculum taught in an undergraduate algorithms course is a fundamentally *social* endeavor; at its heart, it demands participation in the social world of schooled (expert) algorithmaticians.

To see this, let us consider the learning objectives of a typical undergraduate algorithms course. Upon completing an algorithms course, students would ideally be able to (a) present an understandable description of an algorithm; (b) prove an algorithm correct; (c) present a convincing efficiency analysis of an algorithm; and ultimately (d) convince others that a given algorithm is the appropriate one to use in a given situation. While all of these skills certainly have

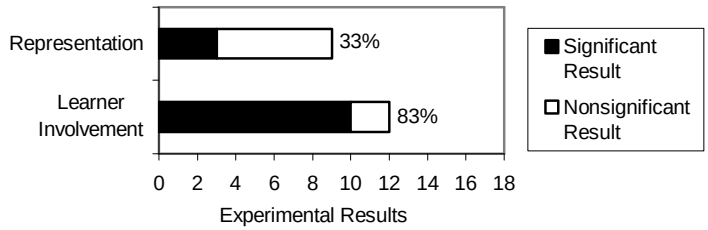


Figure 2. Results of AV Effectiveness Experiments Broadly Classified by Their Independent Variables

key cognitive components, I believe that they are fundamentally social. Indeed, proof, like the presentation of any analysis or procedure, is a "social interaction" that relies on one's ability to communicate using the tools (e.g., computers), conventions (e.g., first describe algorithm, then present proof), and notations (e.g., visual representations) of the community of practice to which one is presenting.

Recognizing the social nature of the skills and competence that students develop in an undergraduate algorithms course, I have become interested in Sociocultural Constructivsm (Lave & Wenger, 1991). This version of Constructivism views knowledge and learning at the level of the community of practice (Wenger, 1998). AV software is seen as pedagogically valuable insofar as it enables students to participate in a course in ways that increasingly resemble the ways in which teachers participate. Inspired by this perspective, I have explored a teaching approach that takes visualization assignments one step further by having students not only construct their own visualizations, but also *present* their visualizations to their instructor and peers for feedback and discussion. In this approach, students assume two tasks that closely resemble those typically performed only by teachers: (1) they construct their own visualizations; and (2) they use their visualizations as a basis for talking about algorithms with others.

Ethnographic Studies

To explore the practical costs and benefits of this approach, I conducted a pair of ethnographic field studies in consecutive offerings of a junior-level algorithms course at the University of Oregon. For an assignment in these courses, students were required to construct their own visualizations of one of the divide-and-conquer, greedy, dynamic programming, or graph algorithms they had studied, and then to present their visualizations to their classmates and instructor during specially-scheduled presentation sessions.

To study students' use of algorithm visualization technology in these courses, I employed a variety of *ethnographic field techniques*, including participant observation, semi-structured interviews, videotape analysis, diary collection, and artifact analysis. Below, I briefly summarize the two studies' key findings. For a comprehensive treatment of these studies, see (Hundhausen, 1999 ch. 4 & app. A–B; Hundhausen, 2001).

Study I. In the first of our ethnographic studies, students used the *SAMBA* algorithm animation package (Stasko, 1997) to construct "high fidelity" visualizations that (a) were capable of illustrating the algorithm for arbitrary input, and (b) tended to have the polished appearance and precision of textbook figures, owing to the fact that they were generated automatically as a byproduct of algorithm execution.

In this first study, three key findings were noteworthy. First, students spent 33.2 hours on average (n = 20) constructing and refining a single visualization. They spent most of that time steeped in *low-level graphics programming*—for example, writing general-purpose graphics routines capable of laying out and updating their visualizations for any reasonable set of input data. Second, in students' subsequent presentations, their visualizations tended to stimulate discussions about *implementation details*—for example, how a particular aspect of a visualization was implemented. Third, in response to questions and feedback from the audience, students often wanted to back up and re-present parts of their visualizations, or to dynamically mark-up and modify them. However, conventional AV software like SAMBA is not designed to support interactive presentations in this way.

Study II. These observations led us to change the visualization assignments significantly for the subsequent offering of the course. In particular, students were required to use simple art supplies (e.g., pens, paper, scissors, transparencies) to construct and present "low fidelity" visualizations that (a) illustrated the target algorithm for a few, carefully-selected input data sets, and (b) tended to have an unpolished, sketched appearance, owing to the fact

that they were generated by hand. In prior work (Douglas, Hundhausen, & McKeown, 1995), I called such visualizations *storyboards*.

In this second study, three key findings stood out. First, students spent 6.2 hours on average (n = 20) constructing and refining a single storyboard. For most of that time, students focused on understanding the target algorithm's procedural behavior, and how they might best communicate it through a visualization. Second, rather than stimulating discussions about implementation details, their storyboards tended to mediate discussions about the *underlying algorithm*, and about how the visualizations might bring out its behavior more clearly. Third, students could readily go back and re-present sections of their visualizations, as well as mark-up and dynamically modify them, in response to audience questions and feedback. As a result, presentations tended to engage the audience more actively in interactive discussions.

In summary, these findings not only suggest that this teaching approach is an effective way of getting students involved in and excited about algorithms, but also furnish three reasons why low fidelity visualizations are superior to high fidelity visualizations within the context of this approach. First, low fidelity visualizations not only take far less time to construct, but also keep students more focused on topics relevant to an undergraduate algorithms course: algorithm concepts, rather than low-level implementation details. Second, in student presentations, low fidelity visualizations stimulate more relevant discussions that focus on algorithms, rather than on implementation details. Finally, low fidelity visualizations are superior in interactive presentations, since they can be backed up and replayed at any point, and even marked-up and modified on the spot, thus enabling presenters to respond more dynamically to their audience.

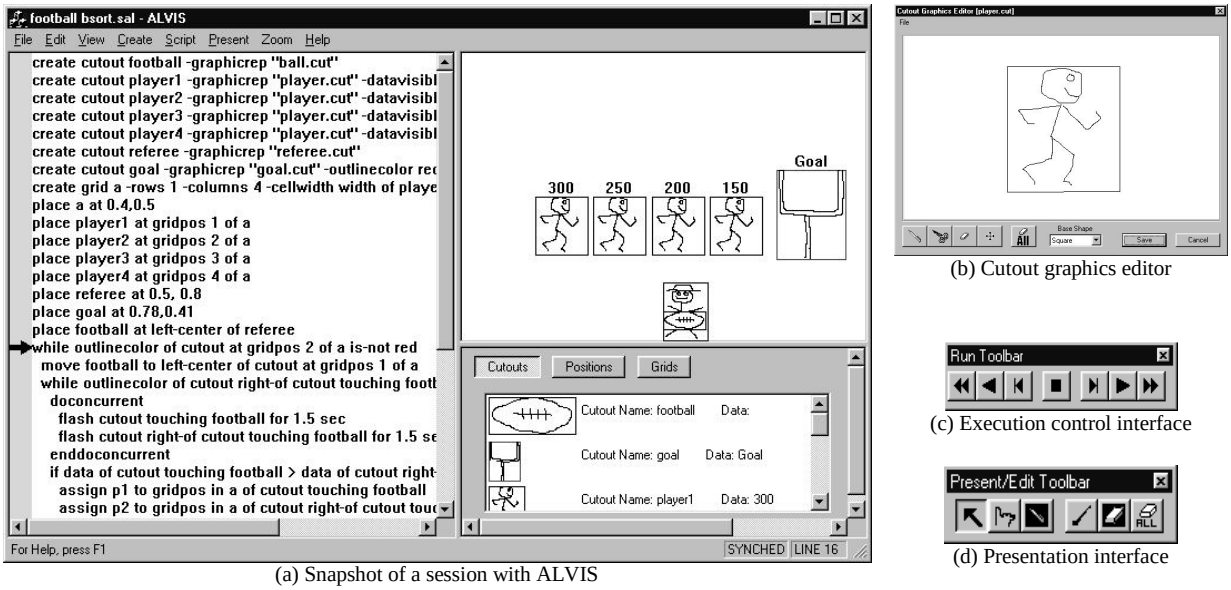
SALSA and ALVIS

In a follow-up research project, I have explored the design implications of my ethnographic study findings through the development of prototype "low fidelity" AV software. The prototype consists of two key components: SALSA (Spatial Algorithmic Language for StoryboArding), a high-level, interpreted language for specifying low-fidelity visualizations, and ALVIS (ALgorithm VIsualization Storyboarder), a direct-manipulation environment for programming in SALSA . For a fuller treatment of this prototype, see (Hundhausen & Douglas, 2001).

Figure 3a presents a snapshot of a sample session with the ALVIS environment. In the *script* region (left), the user is editing a SALSA script that generates a "football" visualization (see Douglas et al., 1995) of the well-known bubblesort algorithm. Unlike "high fidelity" visualization software, which requires users to create objects in terms of Cartesian coordinates, ALVIS enables users to create objects simply by sketching them in a graphics editor (Figure 3b), and dragging-and-dropping them from the *created objects* region (bottom-right) to the *animation* region (top-right). For example, ALVIS generated the statement *place referee at 0.5, 0.8* in response to the user's dragging-and-dropping the referee to its current position.

In my ethnographic studies, as well as in prior empirical studies (Douglas et al., 1995; Douglas, Hundhausen, & McKeown, 1996), I observed that authors of low fidelity visualizations tend to execute their visualizations by maintaining important *spatial relations* among visualization objects. The hallmark of SALSA is its explicit support for referencing visualization objects and programming visualization logic in terms of such spatial relations. For example, the script depicted in Figure 3a flashes the two players currently being compared. Those two players are referenced solely in terms of their spatial relationships to the football (*flash cutout touching football* and *flash cutout right-of cutout touching football*). Likewise, spatial relations can be used to drive the internal logic of a SALSA script. For example, the inner *while* loop of the script depicted in Figure 1a executes as long as the following spatial condition holds: *outlinecolor of cutout right-of cutout touching football is-not red*.

Finally, ALVIS supports three key features specifically designed to mediate communication about algorithms. These features are taken for granted in the art supply presentations I observed in Study II, but are notably absent in conventional "high fidelity" AV technology. First, ALVIS's execution interface (Figure 3c) supports both forwards and *backwards* execution, enabling the user to unwind and re-present portions of the animation at the request of the audience. Second, ALVIS's presentation interface (Figure 3d) provides several tools that support interactive presentation, including a pen and eraser for dynamically marking up a visualization as it is being presented. Third, in response to audience feedback, the presenter may dynamically modify the visualization simply by inserting or removing SALSA code at the point at which the script is presently halted (denoted by the black arrow to the left of the script in Figure 3a).



(a) Snapshot of a session with ALVIS

(b) Cutout graphics editor

(c) Execution control interface

(d) Presentation interface

Figure 3. The ALVIS Interactive Environment

Research Activities

The preceding review both motivates a shift from Epistemic Fidelity to Sociocultural Constructivism as our guiding theory for AV technology, and demonstrates the implications of such a shift for the design and pedagogical use of AV technology. The next step in the research is to make visualization construction and discussion the *central* activities of an algorithms course, and to explore further the impact of visualization construction and discussion on learning. In this section, I describe plans for a studio-based algorithms course, and I outline an accompanying program of empirical research into the value and role of visualizations in learning abstract processes and concepts.

Studio-Based Algorithms Course

The studio-based instructional model owes its origins to the master-apprentice educational system used in the guilds of the Middle Ages (Lackney, 1999). The architectural schools of Europe and North America adopted this instructional model in the form of the "design studio": a place where students set up their own workspaces—drafting tables, books, drawing and modeling materials—and spend much of their lives working individually on common design tasks (Schon, 1983). As students spend long hours working on these tasks, they build camaraderie, looking to each other for support and feedback as they work toward a common purpose. As one architecture student put it, "The long hours of work in a common studio space forged us into a close-knit group. We shared the excitement of learning to see the world in a new way" (Cuff, 1992; cited in Lackney, 1999).

The design studio curriculum encompasses a series of design problems, which may be either progressively more challenging, or different components of the same overall design project. The cornerstones of the curriculum are the "design crits" (design critiques): review sessions at which students present their evolving solutions to these problems for feedback and discussion. Design crits take four different forms. At the most informal of these, the weekly "desk crit," the studio instructor meets with each student one-on-one to review the student's design sketches and give critical feedback. Based on that feedback, the student refines her design, presenting an updated design at the next desk crit. This design process may go through several iterations, ending when the instructor and student are satisfied

Three other types of "design crits" are more formal. In the "pin-up," students present their designs and design rationales to their peers and instructor. In addition to producing specific comments about design, pin-up sessions typically generate higher-level discussions about the general principles and methods being explored in the course. The "midterm crit" and "final crit" are the most formal reviews. At these, students formally present their work to a jury of architects. A popular version of the formal crit is the "Oregon Crit" (so named because it was developed at the University of Oregon). In the style of a science fair, students set up booths displaying their work, and jurors visit each booth for a one-on-one critique and discussion.

In their comprehensive review of architecture education, Boyer and Mitgang (1996) concluded that "the core elements of architectural education—learning to design within constraints, collaborative learning, and the refining of knowledge through the reflective act of design—have relevance and power far beyond the training of future architects" (p. iv). Given the progression of research presented in the previous section, this form of instruction has tremendous potential to enhance algorithms education; it is, in fact, the next logical step in my exploration of a Sociocultural Constructivist approach to using AV technology.

Curriculum Development

I will develop and pilot a one-semester studio-based algorithms course. The key learning objectives of the course will mirror those of a typical junior-level algorithms course (Tucker et al., 1990): (a) to develop proficiency at designing efficient algorithmic solutions; (b) to develop proficiency at analyzing the efficiency of algorithms; and (c) to develop proficiency at proving the correctness of algorithms.

Just like concrete buildings, abstract computer algorithms are designed according to conventional design strategies and principles. For example, students typically learn four general problem-solving strategies in an algorithms course (see, e.g., Cormen, Leiserson, & Rivest, 1990): divide-and-conquer, greedy, dynamic programming, and graph algorithms. In the application of these strategies, a key principle is to design an algorithm that solves the problem as efficiently as possible. Following this principle, one quickly discovers the classic tradeoff between space and time: the fastest algorithms often require the most computer memory. Therefore, designing efficient algorithms requires finessing this tradeoff based on the specific constraints of the problem.

Students will learn how to apply such principles and strategies for "good" algorithmic design in the "algorithms studio," where they will use SALSA and ALVIS to construct visualizations that explore the design space of solutions to strategically-selected algorithm design problems. These problems will cover the four fundamental problem-solving strategies listed above. The end goal will be to "actively" learn algorithm design by finding, with the help of one's instructor and peers, the "best" solutions, thereby developing a feel for just what "best" means in each specific situation.

In order to develop competence at time complexity and correctness proving, students will perform proofs and efficiency analyses in the studio orally, and using visualization as appropriate (see, e.g., Goodrich & Tamassia, 1998). At regularly-scheduled desk crits, students will present their intermediate design solutions, efficiency analyses, and proofs to their instructor in one-on-one sessions. Four pin-up sessions per semester (one for each of the four problem solving strategies) will give students the opportunity to present their work to the entire studio. Finally, instead of midterm and final exams, students will present an "algorithms portfolio"— their solutions, proofs, and efficiency analyses for an extra set of design problems—to a jury of instructors at science-fair style "Oregon Crits."

A detailed syllabus, which will include a collection of studio design problems, concrete learning objectives will be developed during the first year of the funded research, and pilot tested in an actual offering of ICS 311 ("Algorithms and Data Structures") in the information and computer science department at the University of Hawai'i during the second year. Based on the pilot offering, I will refine the course during the third year of the funded research, and offer it a second time during the fourth year of the funded research.

Technology Development

In the "algorithms studio," electronic whiteboards ("SmartBoards"; see http://www.smarttech.com), which are especially well-suited to collaborative discussions, will serve as students' drafting tables. Their drawing and modeling materials will be "low fidelity" AV technology: ALVIS and SALSA. In the first year of the funded research, I will establish this technological infrastructure by purchasing four rear-projection SmartBoards (enough for a pilot course of 20 students who rotate their access to the studio), and by developing a preliminary version of ALVIS and SALSA suitable for use in the algorithms studio. Funded by an Educational Improvement Grant I received from the University of Hawai'i, an undergraduate student (Chad Takahashi) began improving my dissertation prototype of SALSA and ALVIS during the summer of 2001. At the time of this writing, key updates to be made include: (a) fully implementing the SALSA language (see app. F of (Hundhausen, 1999) for a complete specification); (b) adding full "undo" support for graphics editing and direct manipulation command specification; and (c) customizing ALVIS so that it works fluidly with SmartBoards, which entails supporting script writing through gesture recognition in the script window (see Figure 3a).

Using ALVIS and SALSA for the studio-based course raises a potential concern: namely, that students will have to take precious class time to learn an unfamiliar programming language and system. Indeed, why not just use art supplies, or a language that looks more like one that students already know (e.g., Java)? I can cite two reasons. First, because it is technology-based, SALSA has a distinct advantage over art supplies: SALSA scripts may be modified and reused more easily than art supplies. For example, rather than having to create a new visualization storyboard by hand for each input data set of interest, a SALSA programmer can create one visualization storyboard, and simply tweak the data values of the objects. Second, based on extensive empirical studies, I have designed (and will continue to refine through iterative usability testing) SALSA and ALVIS such that they are as learnable, natural, and easy to use as art supplies. Thus, the overhead of learning ALVIS and SALSA will be minimal.

Formation of Community and Dissemination

During the first year of the funded research, I will establish a "studio-based visualization" web site, which I will update throughout the funded period with the latest versions of the curricular materials, ALVIS and SALSA software, C++ source code (the system will be "open source"; see, e.g., Pavlicek & Miller, 2000), user's manual, and related publications and reports. The web site will be designed to encourage collaboration with other computer science educators and technologists, who will be invited to use the materials and software, modify them as needed, and post their experiences and suggestions to others in the community. In this way, I hope gradually to build a community of educators and technologists (a) to support each other in the implementation of studio-based courses, and (b) to continue the active development of the curriculum and supporting technology.

Empirical Research

The instructional approach described above aims to help students learn algorithm design and analysis by *doing* algorithm design and analysis with the support and guidance of a community of algorithms practitioners. In this process, the visual representations of the community—algorithm visualizations—play a central role, both in getting students more involved (i.e., facilitating access to the community), and in mediating meaningful discussions about algorithms. The second key component of this research is a comprehensive program of empirical research into the role and value of visualization and communication in facilitating learning at three conceptual levels: *cognitive*, *social*, and *cultural*. Below, I describe planned empirical studies at each of these levels.

Exploring Cognitive Dimensions

At the cognitive level, learning occurs at the level of the individual, in the form of changes in students' conceptual and procedural knowledge of algorithms. The guiding research question is, "Do students who construct their own algorithm visualizations and discuss them with others learn algorithms better than students who study algorithms through more conventional approaches?" Past empirical research into AV effectiveness has focused squarely on learning at the cognitive level of understanding and recall. Indeed, the legacy of controlled experimental studies reviewed in the previous section measured learning exclusively at this level.

I will build on these experiments by conducting a systematic study of the cognitive impact of the studio-based approach. At a high level, the approach manipulates two general factors: *pedagogy* (lecture-based, studio-based) and *visualization technology* (high fidelity, low fidelity). However, as discussed in the previous section, past experiments have demonstrated that *actively* viewing a visualization (by, for example, designing input data or making predications) is more pedagogically effective than viewing a lecture without technology (see, e.g., Byrne et al., 1999; Lawrence, 1993, ch. 9). Given these prior results, it makes sense to explore the impact of various factors within the studio-based approach. Specifically, two factors are integral to the approach: (a) *self-construction*—whether or not students construct their own visualizations; and (b) *instructor communication*—whether or not students discuss the visualizations with an instructor.

The hypothesis to be tested is that students who construct their own visualizations and discuss them with their instructor will learn algorithms better than students who actively view a pre-defined visualization, and who do not interact with their instructor. Here, learning will be operationalized in terms of two dependent variables: (a) *procedural understanding*—how quickly and accurately a student can perform a detailed trace of an algorithm's procedural behavior for a range of best case, worst case, and average case input data; and (b) *recall*—how quickly and accurately a student can write a procedural description of the algorithm in a programming language.

In a prior experiment that tested the self-construction factor alone (Hundhausen & Douglas, 2000), I did not find a significant effect on the above dependent variables; however, the trends were encouraging, and I was able to gain three important insights that will be applied to the experimental design proposed here. First, I found that the effect of

self-construction was not strong enough to make a difference in a single, two-and-a-half hour learning session. I believe, however, that a significant effect would have emerged, had the treatment been applied over a longer period of time. Second, because they did not receive feedback from an expert, participants in the "self-construction" group frequently constructed visualizations that did not accurately portray the algorithm. By contrast, participants in the control group studied visualizations that were guaranteed always to be correct. Thus, "self-construction" participants may have done better, had they also discussed their visualizations with an instructor. Finally, the experiment measured absolute learning in terms of post-test performance. However, a review of past experimental studies of AV (Hundhausen et al., 2001) suggests that pre-test to post-test improvement is a more reliable measure of learning, because it controls for students' background knowledge on the target algorithm.

Building on the above insights, I will conduct a 2×2 mixed factor experiment with four treatment groups: (a) Self-Construction; (b) Self-Construction + Instructor Interaction; (c) Active Viewing; and (d) Active Viewing + Instructor Interaction. This design will allow me to test for interaction effects, which my prior experiment suggests may be significant. In order to observe the effects of the treatments over a longer period of time, I will conduct the study in conjunction with an offering of ICS 311 ("Algorithms and Data Structures"), from which I will recruit participants for three separate experimental sessions coinciding with weeks 5, 10, and 15 of the semester. Treatment group size will range from 15 to 20, depending on the number of students willing to participate.

At each experimental session, participants will use visualization and instructor interaction to learn about a different target algorithm. I will select the target algorithms to represent the major problem-solving strategies covered in the ICS 311 course. Each experimental session will have an identical procedure similar to that used in my prior experiment. First, participants will begin with a knowledge pre-test for the algorithm they will be learning. This pre-test will consist of tracing and programming tasks similar to the ones that they will perform in the post-test. Second, they will be debriefed on how to use the learning materials. All participants will receive a textual description of the target algorithm, excerpted from their textbook. Self-Construction participants will receive an array of simple art supplies (paper, pens, scissors, etc.).[4] They will be instructed to use these materials to construct their own "low fidelity" visualizations of the algorithm operating on a variety of input data sets. In contrast, the Active Viewing participants will receive training on how to use a computer-based, "high fidelity" visualization of the target algorithm. They will be asked to "actively view" the visualization by specifying their own input data sets and observing the visualization operate on those sets. Third, all participants will have two hours and 10 minutes to learn the algorithm. Fourth, participants in Instructor Interaction conditions will be given 20 minutes to discuss their visualizations (or the pre-defined visualization) with their course instructor; the treatments without instructor interaction will be given that time to continue studying. Finally, all participants will take a post-test consisting of a tracing and programming tasks.

To analyze the experiment data, I will average participants' pre-test to post-test improvement scores and time improvements over the three semester sessions; statistical tests will determine whether significant differences exist. I will also videotape participant learning sessions and participant-instructor interaction. Later analysis of these videotapes will assist me in interpreting the quantitative results, and form the basis for further empirical studies of the social dimensions of visualization-mediated learning, as described below.

Exploring Social Dimensions

At the social level, learning takes place through the collaborative negotiation of shared understandings (see. e.g., Suchman, 1987); visualizations serve as *mediational resources* (Roschelle, 1994) in that endeavor. The guiding research question is, "In what ways, and to what extent, do algorithm visualizations mediate conversations that (a) are relevant to algorithms, and (b) enable students and instructors to negotiate shared understandings of algorithms?"

In two of the four treatments of the controlled experiment described above, students will discuss either their own, "low fidelity" visualizations, or a predefined "high fidelity" visualization, with an instructor. These student-instructor interaction sessions will be the focus of a further empirical study. Videotapes of student-instructor interaction will be transcribed, including all gestures and interaction with the visualizations. These transcripts will form the foundation for two detailed analyses of the role and value of visualization in facilitating communication about algorithms:

1. *Do visualizations focus conversational participants on relevant topics?* Through pilot studies and consultation with my advisory board (see next section), I will develop a scheme for coding transcribed utterances and actions

[4]In this study, I have tentatively chosen to use art supplies, rather than ALVIS and SALSA, because I do not want software usability to be a confounding factor. This may change, however, if the usability of ALVIS and SALSA are sufficiently high by the time of the experiment.

according to their relevance to algorithms. Note that I already have extensive experience with developing, refining, and applying content coding schemes through prior studies of collaborative science learning (Suthers & Hundhausen, 2001, 2002). For example, utterances that describe a procedural step of an algorithm might be coded as "algorithmic step; if such utterances rely on concurrent pointing actions to the visualization, they would be modified by a "representational" code. To ensure the reliability of the coding system developed, I will have multiple analysts code 20% of the transcripts, and verify their inter-rater agreement (Shrout & Fleiss, 1979). Based on such coding, I will compute the percentage of utterances dedicated to various relevant topics, and perform non-parametric tests for statistical significance between the "low fidelity" and "high fidelity" groups.

2. *In what ways do visualizations help students and instructors to build shared understandings of algorithms?* In a follow-up qualitative study, I will use Interaction Analysis (Jordan & Henderson, 1995) to explore the ways in which visualizations mediate conversations about algorithms. In this analysis, I will focus not only on points in the interaction at which visualizations successfully mediate conversations, but also on points at which communication breaks down—that is, participants' mutual intelligibility is lost. If breakdowns appear abundant or follow recognizable patterns, I may elect to pursue a quantitative analysis of communication breakdown based on the methodology of Doerry (1995).

Exploring Cultural Dimensions

At the cultural level, learning takes place as learners become fuller participants of a community by increasingly taking on the roles and responsibilities of community experts (Lave & Wenger, 1991). The guiding research question is, "In what ways, and to what extent, do visualization construction and presentation enable students to become fuller participants of the community being reproduced through an undergraduate algorithms course?"

I will explore the cultural dimensions of visualization construction and presentation through two separate empirical studies. First, in my pilot offerings of the studio-based algorithms course, I will have a graduate research assistant conduct ethnographic studies of the course, in order to obtain an insider's perspective on what it is like to be a student in the course. The graduate assistant's perspective, along with my own perspective on the instructional aspects of the course, will be combined to produce a rich, ethnographic account of the community-building value of the studio-based algorithms curriculum. To assist us in assessing students' centripetal movement in the community, we will administer a carefully-designed *attitude questionnaire* (see, e.g., Hundhausen, 1999, app. G) at both the beginning and end of the course; changes in student attitudes towards algorithms, as measured by the questionnaire, will be included in our account.

Second, I will explore and develop a research methodology for gauging level of community membership based on performance in visualization construction and interpretation tasks. Rooted in Cultural Consensus Theory (Romney, Weller, & Batchelder, 1986), a product of cognitive anthropology that derives a formal statistical model for gauging one's level of community membership based on one's level of agreement with others in the community, the methodology will offer a means of measuring learning at a cultural level. The basis of this measurement is the extent to which students' performance in algorithm visualization reading and writing tasks converges on that of community experts (algorithms instructors) over the course of a semester-long algorithms course. Below, I propose a "consensus study" based on algorithm visualization construction and interpretation. Because I am aiming to apply consensus theory to a novel domain (previously, cognitive anthropologists have used the theory to gauge agreement on far simpler tasks like naming plants; see, e.g., Boster, 1985), the design of the study I sketch out here must be seen as evolving, and may be refined over the course of the funded research. In this process, I will benefit extensively from collaboration with Consensus Theory expert A. Kimball Romney (see "Advisory Board" below). For a fuller treatment of my current ideas on this evolving methodology, see (Hundhausen, 1999).

In the consensus study, two kinds of participants—students (community newcomers) and instructors (community experts)—will perform two different tasks: (a) *visualization reading*—meaningfully interpreting a graphical representation of an algorithm; and (b) *visualization writing*—constructing one's own graphical representation of an algorithm. If a distinct community of algorithmaticians really exists, one would expect a high level of agreement among its experts in the way they perform these tasks. For example, experts tend to "read" a given visualization in a similar way.[5] Likewise, given an algorithm, experts "write" similar visualizations that describe it—that is,

[5]For empirical evidence that central members of a community of practice read the representations of the community in a similar way, see (Petre & Green, 1993), who studied digital circuit designers.

visualizations that capture the visualization in terms of a similar semantics.[6] The challenge, of course, lies in characterizing agreement in these tasks *quantitatively*, so that results can be fed to Consensus Theory's statistical model.

To quantitatively assess the level of agreement among community members' visualization reading and writing activities, I propose a technique that draws on the *semantic level analysis* technique developed in prior work (Douglas et al., 1995, 1996). Given a sample of visualizations that describe an identical algorithm, semantic level analysis involves mapping the lexical entities, attributes, and transformations of those visualizations to the underlying semantics of the algorithm. Algorithm semantics is expressed in terms of a pseudocode-like description of the algorithm; each mapping between a lexical item of an AV and a variable or statement in the underlying pseudocode-like description is called a *semantic primitive*.

Using this mapping technique, one can derive a set of semantic primitives for each visualization in the sample. A visualization's semantic primitive set can be seen as a summary of the way in which the visualization represents the target algorithm. The quantitative agreement between any two visualizations in the sample can then be calculated by dividing the intersection of the two visualizations' semantic primitive sets by their union.

With this evolving method for measuring agreement, I have a foundation for a consensus study of cultural learning. First, a group of expert participants (course instructors) constructs and interprets visualizations of two different algorithms. Their level of agreement, as well as the "culturally-correct" sets of semantic primitives derived from their solutions, is used as a baseline for the consensus study. At the beginning of an algorithms course, student participants perform the same construction and interpretation tasks on one of the first of the target algorithms; they repeat the tasks at the end of the course for the second target algorithm. Using Consensus Theory's statistical model, one measures students' convergence on expert cultural competence based on their longitudinal improvement in these tasks. To test the efficacy of the studio-based course's ability to build community, one compares average student convergence in the studio-based course against average student convergence in a "control" course that is taught using conventional teaching methods. Statistically significant differences in levels of convergence are grounds for concluding that one course promotes cultural learning better than the other.

Additional Educational Activities

The research activities outlined in the previous section focus squarely on education. In line with the CAREER program's emphasis on faculty development and educational enrichment, this section describes additional educational activities that will be supported through this research.

Curriculum Development

Funded by a grant from the Sloan Foundation, my department is in the process of developing completely on-line bachelor's and master's degree programs. In this endeavor, I have been involved both as an advisor and as a developer of curricular materials. In the coming academic year, I will develop and pilot my department's preliminary on-line versions of the introductory and second-semester computer science courses.

Given the challenges entailed in getting students to actively participate in on-line courses, the research proposed here would enhance my department's on-line computer science curriculum. Nearly all computer science courses introduce some sort of notation that is used to build and discuss representations. For example, the introductory course I am teaching stresses a disciplined approach to programming in which one creates a high-level object-oriented design before one programs the solution. Such high-level object models are used even more extensively in our software engineering courses. Likewise, our human-computer interaction courses encourage the development of user interface sketches before actually implementing a user interface.

Clearly, the design of on-line activities rooted in the construction and discussion of representations can benefit from the results of the proposed research. For instance, the low fidelity visualization software I will develop could be used to rapidly prototype conceptual design ideas for subsequent presentation in on-line forums. Likewise, the studio-based curriculum I am developing lends itself well to implementation on-line; an on-line course could pose a series of programming or design problems, solutions to which students would be required to work out on their own (or perhaps in an on-line studio), and post to an on-line forum for discussion and feedback. To support such discussion forums, my colleague, Dan Suthers (see "Advisory Board" below), is developing a collection of tools for on-line

[6]This is, in fact, borne out by recent empirical studies that explored the human visualization of algorithms; see (Chaabouni, 1996; Douglas et al., 1995, 1996).

artifact-centered discourse. Our work could be amalgamated in a project that adapts a studio-based curriculum to an on-line paradigm.

Student Apprenticeship

Given the positive impact of my previous apprenticeships with undergraduate (Thomas Naps) and graduate (Sarah Douglas) mentors on my own development, I look forward to participating in them from the other side—as a mentor—so that I might give back to upcoming students some of the energy and guidance that my mentors gave to me. Having benefited from four years of summer mentorship as an undergraduate, I strongly believe in the power of undergraduate mentoring to shape future researchers. With CAREER funding, I will hire one undergraduate student per year for a summer research apprenticeship. In addition, I will mentor two graduate students. For one of these graduate positions, I will recruit a Ph.D. student interested in pursuing original research into visualization and human-computer interaction. Finally, this work will benefit many more students in our department: those involved both in the studio-based courses I will teach, and the empirical studies I will conduct. By experiencing innovative instructional methods and seeing educational research first-hand, these students may be inspired to stay in a degree program they might otherwise have left, or even to continue on to graduate school.

Advisory Board

The six researchers briefly introduced below have agreed to serve on an advisory board for this project, and will contribute valuable expertise and experience. I have budgeted for one trip per year to meet with each of my advisors. This will enable me to collaborate more closely with them, and it will help my advisors keep abreast of my progress.

Alan Blackwell is a University Lecturer in Computer Science at Cambridge University, a Research Fellow of Darwin College and the Cambridge-MIT Institute, and co-director of Crucible, the Cambridge Network for Research in Interdisciplinary Design. Alan's research focuses on the use of visual representations to communicate abstraction to end-users; he contributed to the well-known "cognitive dimensions" framework. (Green & Petre, 1996).

Sarah Douglas is professor and Chair of the computer and information science department at the University of Oregon. Sarah brings to this project many years' experience in designing graphical systems for end-user programming (see, e.g., Douglas, Doerry, & Novick, 1992) and education (see, e.g., Douglas, Peterson, & Udovic, 1998), as well as a passion for designing technology firmly rooted in empirical research.

Thomas Naps is a professor of computer science at the University of Wisconsin–Oshkosh, having recently moved there from Lawrence University in Appleton, Wisconsin, where he established the Algorithm Visualization Laboratory and accompanying software (Naps, 1990) over the past decade. Author of over 20 textbooks on algorithms and data structures, Thomas brings to this project both expertise in algorithm design and analysis, and extensive experience with teaching algorithms and data structures with the help of AV technology.

A. Kimball Romney is a professor of anthropology at the University of California–Irvine, and has been an influential researcher in cognitive anthropology since the 1950's. Co-developer of the Consensus Theory Model on which my cultural studies of visualization are based, (Romney et al., 1986), Kimball brings to this project a strong background in quantitative anthropology and statistical methods.

John Stasko is an associate professor in the College of Computing and the Graphics, Visualization, and Usability Center at the Georgia Institute of Technology. One of the pioneers of algorithm animation, John developed the widely used TANGO algorithm animation as part of his dissertation research at Brown University (Stasko, 1989). John brings to this project extensive experience in empirically studying algorithm visualization effectiveness (e.g., Byrne et al., 1999; Kehoe, Stasko, & Taylor, 2001; Stasko et al., 1993).

Dan Suthers is an assistant professor in the Information and Computer Sciences Department at the University of Hawai`i. For over a decade, Dan has been exploring the use of visual knowledge representations to help make scientific reasoning and argumentation more accessible to students (see, e.g., Suthers, Toth, & Weiner, 1997). Dan brings to this project valuable experience in the design and analysis of empirical studies of the role and value of representations in collaborative learning (see, e.g., Suthers & Hundhausen, 2001, 2002)

Timeline

The following timeline provides a high-level picture of how the funded research will proceed:

Year 1. Recruit and hire a Ph.D. student. Offer undergraduate independent study credits for SALSA and ALVIS development. Complete development of initial version of ALVIS and SALSA through iterative usability testing. Port ALVIS and SALSA to SmartBoard electronic whiteboard platform. Prepare detailed syllabus and curricular materials for studio-based algorithms course. Establish "studio-based visualization" web site and post preliminary curricular materials, software, source code, and user's manual.

Year 2. Prepare high fidelity animations, detailed protocol, and pre- and post-tests for experiment. Develop coding system for analyses of student-instructor communication. Teach first pilot offering of studio-based algorithms course during winter semester; have research assistant perform fieldwork in course. Publish updated systems paper on SmartBoard version of ALVIS and SALSA. Continue to post regular updates to the web site. Establish active partnership with at least one other educator and technologist.

Year 3. Run experiment in conjunction with fall, 2004 offering of ICS 311. Analyze quantitative results. Transcribe and analyze videotapes. Publish results of preliminary pilot offering of studio-based algorithms course. Revise studio-based materials and curriculum based on results. Continue to post regular updates to web site. Establish two new partnerships with educators and technologists.

Year 4. Refine consensus study methods with pilot tests, and in collaboration with Kimball Romney. Develop detailed protocol for consensus study. Teach second pilot offering of studio-based algorithms course during winter semester; have research assistant perform fieldwork in course. Run consensus study of studio-based learning in conjunction with second pilot offering. Publish results of experiment and videotape analyses. Continue to post regular updates to web site. Establish five new partnerships with educators and technologists.

Year 5. Run consensus study in conjunction with fall, 2006 offering of ICS 311 (the "control" treatment). Publish results of second pilot offering of studio-based algorithms course. Analyze and publish results of consensus studies. Continue to post regular updates to web site. Anticipate active educator/technologist community by this time, with interest to continue the development of the curriculum and technology beyond the funded period.

Summary and Broader Impacts

Focusing on computer science education, this proposal has motivated the need for a shift in our guiding theory of effectiveness for computer-based visualization. Rather than viewing computer-based visualization as a knowledge conveyor that transfers an expert's mental model to a learner, I have explored an alternative view in which visualization is seen to be effective insofar as it enables learners to participate in a community of practice in increasingly expert ways, and insofar as it mediates meaningful discussions about a domain of interest. Through the presentation of ethnographic field studies and a prototype visualization tool, I have illustrated the implications of this alternative theoretical position for the design and pedagogical use of visualization technology, and I have motivated a research agenda that takes the approach further and empirically investigates its impact on learning.

This research will have important implications for computer science education. Due to the recent explosion in the need for highly-skilled software engineers, undergraduate programs are unable to graduate enough students to meet demand. In my own department, this is especially apparent; our size has recently ballooned to over 1,000 majors, yet we are able to graduate only about 50 students per semester. One of the bottlenecks in our program, and in many other computer science programs, is the junior-level algorithms course, which has a reputation for being extremely difficult because of its abstract, mathematical nature. Clearly, if the studio-based curriculum I am proposing succeeds, the junior-level algorithms course will become more accessible. In addition, as I discussed earlier in the proposal, the approach might also improve the accessibility of other computer science courses, such as software engineering and user interface design. As a result, more students may be inspired, rather than put off, by computer science courses, and more much-needed engineers may ultimately graduate from computer science programs.

The proposed studio-based approach also has applications within other science curricula. For example, a studio-based biology course might have students use an interactive biological system design tool such as the Cardio-Vascular Construction Kit (Douglas et al., 1998) to construct and present their own simulations. Students' presentations would set up ideal conditions under which to discuss the higher-level concepts of pressure and flow. Likewise, physics students might use a Newtonian physics visualization system such as the Envisioning Machine (Roschelle, 1994) to construct simulations of particle motion. In student presentations, visualizations would mediate meaningful conversations about the higher-level concepts of velocity and acceleration. As in computer science, the ultimate benefit would be that students would find the material less intimidating and more accessible, thus enabling them to finish a course or degree that they might not otherwise have completed.