Developing Methods to Assess Innovative Curricula for Women and Non-Majors in Computer Science

PI: Mark Guzdial

October 28, 2003

Project Summary

Computer science educators are aware that they have a problem in motivating and engaging students, especially non-CS-majors and women, who are withdrawing or failing introductory computing courses at an alarming rate, or simply avoiding computing entirely.

There are several innovative efforts to address this problem. For example, CMU has made great progress on increasing the percentage of women in their undergraduate program. We at Georgia Tech have had a successful trial offering of a course in *Introduction to Media Computation* aimed at non-CS majors. By conventional assessment, such as retention and WFD rates, the course was quite successful at its goals. 121 students enrolled—2/3 women. 89% of the class earned an A, B, or C in the course. 60% of the respondents on a final survey indicated that they would like to take a second course on the topic.

While our intervention was successful, we are limited in what we can demonstrate and the claims we can support. Traditional courses differ too much from non-traditional courses e.g., different languages, different content, different ordering of concepts) to easily compare and demonstrate learning benefits. We are unable to show that attitudes towards computer science in general (as opposed to attitudes toward our course) have changed. If there are going to be many efforts to attract under-represented groups into Computer Science, we need some way of evaluating the efforts beyond retention and WFD rates.

We propose to develop measures of CS1 concept learning, attitudes toward computer science, and longitudinal impact of introductory computing aimed at women and non-CS majors. We plan to disseminate these measures through papers and also faculty workshops at CS Education conferences such as ACM SIGSE and ASEE/IEEE FIE. We plan to study faculty use of these measures through email surveys of faculty who attend our workshops or use our course materials

Intellectual Merit: This proposal seeks to create methods to understand the value of innovations to attract women and non-CS majors to computing.

Broader Impacts: The potential broader impacts include helping develop better methods to address the lack of women in computing by providing an infrastructure for comparing innovations, and in doing so, creating a broader-base of computing-literate citizens.

Developing Methods to Assess Innovative Curricula for Women and Non-Majors in Computer Science

Contents

Developing Methods to Assess Innovative Curricula for Women and Non-Majors in Computer Science

1 Goal: Assessing New Routes to Computer Science

Computer science departments are not currently successful at engaging a wide range of students who are taking introductory computer science. The evidence for this statement includes international studies of programming performance [30], declining retention rates [16], and failure rates sometimes as high as 50% [40]. This comes at a time when the need for Information Technology (IT) professionals is growing [12].

At Georgia Institute of Technology ("Georgia Tech"), all students are required to take an introductory course in computing, including programming skills. Our traditional CS1 course, the only one that met the requirement before the Spring 2003 semester, is undoubtedly one of the most unpopular courses on campus, especially among those *not* in explicitly computing-related fields. While this is certainly a problem for the College of Computing at Georgia Tech (where the course has its academic home), it points towards a larger problem for the field.

Alan Perlis in April 1961 made the first argument that programming should be part of a liberal education for *all* students. If Calculus is the study of *rates*, and that's important enough to be part of the liberal education, then so should computer science. Perlis argued that computer science is the study of *process*, which is certainly relevant to even more fields than those concerned with rates. The argument has been echoed and strengthened over the intervening years—by Seymour Papert arguing for a programming as a way of learning about learning [35][36], to Andrea diSessa's arguments for "computational literacy" as a critical component of many fields [9]. As long as non-CS majors have such a dislike for computing, the hope is diminished for computer science as an accepted part of a liberal education and for computing generally to meet its potential for intellectual impact across the range of disciplines, not just in computational science and engineering.

Introductory computer science classes are also not meeting the needs of many women. The rates at which women take computer science classes are falling. While the factors causing women to avoid computer science (and IT careers in general) are multi-faceted, the curriculum does play a significant role [28]. A report in 2000 by the American Association of University Women [1] suggested that part of the problem, at least for women, is that computer science courses are, frankly, too boring. Specifically, they claim that computer science courses are "overly technical" with little room for "tinkering." At a session on increasing enrollment of women in computer science at the latest ACM SIGCSE conference, speakers reported that women who pursued computer science degrees were surprised at how much "creativity" there was in later computer science courses—introductory courses did not highlight that aspect of CS [38]. Additionally, women undergraduate CS students tend to be interested in real applications of computing, as opposed to simply computing for its own sake [28][1]. If we can address the reasons why women are avoiding computer science, we may be able to interest more males, too—those currently expressing disinterest in computing. "Women in engineering programs are kind of like 'canaries in the coal mine"', said Stephen

W. Director, Chair of the Engineering Dean's council. "If women do well in a program, most likely everyone else will also do well in the same type of program." [1]

There are new, innovative efforts to address these issues. Most well-known among these is the work at Carnegie-Mellon University (CMU) which was able to raise dramatically the percentage of women in their undergraduate CS program [28]. New courses are being developed, such as an Alice-based pre-CS1 course [8], whose goal is to meet some of the problems identified as preventing women and non-CS majors from succeeding at Computing. The Alice-based course is succeeding at having higher retention rates than other introductory computing courses [8]. Other innovative new courses include a focus on graphics [29], virtual reality [46], and Web programming (e.g., servlets) [23]. New approaches within existing classes, like pair-programming methods, are leading to improved retention in the class and within the CS major [31, 33].

We at Georgia Tech have developed a course, *Introduction to Media Computation*, to meet the needs of women and non-CS majors. The course was offered as a pilot in Spring 2003 with 121 students, 2/3 women, with *none* of the students majoring in CS or Engineering. These students were from the College of Architecture, Ivan Allen College of the Liberal Arts, Dupree College of Management, and the School of Biology. The hypotheses of this effort were that this course would demonstrate: (1) Improved student retention, (2) better student attitudes toward computer science, and (3) better student learning of computer science.

The course was quite successful at the goal of improving student retention, but it's not clear if it met the goal of impacting student attitudes or learning. 121 students enrolled – 2/3 women. 89% of the class earned an A, B, or C in the course, i.e., 11% Withdrawal-Failure-or-D (WFD) rate. 60% of the respondents on a final survey indicated that they would like to take a second course on the topic–but at the same time, only 9% claimed that they would ever take another course in computer science! They did learn enough programming to successfully write original programs. Students wrote eight programs (six collaboratively and two individually) creating or manipulating pictures, sounds, HTML pages, and movies, with some of their programs reaching 100 lines of code. When we attempted to use isomorphic exam problems (as in [18]) to compare students between the classes, we ran into difficulties with course sequencing and language differences.

We do not believe that media computation is the *only* context in which students not traditionally attracted to computer science might succeed. There are certainly others. For example, we have described elsewhere another potential CS1 course organized around information management in which students might build Web harvesting programs and data visualizations [21]. The general approach we are exploring is to use an application domain of information technology that is rich in CS concepts and relevant to the students, then explore introducing computing concepts in terms of that domain. The Media Computation project is a trial and model of that approach.

We anticipate that there will be *many* innovative approaches developed over the next five

[1] Testimony by Stephen W. Director, Chair, Engineering Dean's Council, American Society of Engineering Education, to the Commission on the Advancement of Women and Minorities in Science, Engineering, and Technology Development, Washington, DC. July 20, 1999

to ten years, that seek to improve the motivation and engagement with computer science among students in groups not currently succeeding at computer science. We predict that all of these approaches will reach the same problem. It's easy to show that one's innovation changes WFD rates or retention rates. It's another thing to show that the innovation is actually making a difference:

- How do we know that students are really *learning* after the changes? One way of lowering WDF rates is to simply make the class easier. Clearly, we argue that we have *not* done that, as would any of the developers of other new courses or new methods. But that's a hard claim to develop evidence for.

- Are the innovative classes changing attitudes about computer science? Or are the other classes facing the same muddle of attitudes that we have—students like our class, but say that they still don't like computer science? Attitudes are important to address if we care about developing a well-educated citizenry (e.g., through non-CS majors). If our non-CS major students become the leaders of this country, and still believe that CS is irrelevant or uninteresting, we will have done them a disservice.

- How do the innovations impact the students beyond the course? We would like to see the students go on to take more computer science classes, and perhaps some percentage become computer science majors. But we would also like to see an impact within the students' major. Introductory computing courses for non-majors are growing (e.g., [24] [46] [29]), but we have little idea of how those courses is actually changing how students relate to technology. For example, we would like to know if students who take introductory computing courses feel that they can learn and use new technology, perhaps more easily (in their estimation) than others without computing education.

In this proposal, we propose to create three measures that go beyond issues of retention and WFD rates to show them impact of educational innovations targeting women and non-CS majors. Our measures will focus on attitudes toward computer science and technology, learning of concepts related to CS1, and on longer term impacts of the computing course on the students' later academic career. We begin by describing our proof-of-concept trial offering of the course (funded by NSF CCLI grant #0231176) and its evaluation results, because it was the results from that trial that has led to our motivation to develop these methods. We end by describing our dissemination plans, which include faculty workshops at ACM SIGCSE and ASEE/IEEE FIE conferences to teach broadly (e.g., including our methods, but also including measures such as retention and WDF rates) how to assess the impact of educational innovation targeting under-represented groups in computer science.

2 Results from Proof-of-Concept Trial

Introduction to Media Computation (CS1315 at Georgia Tech) is our new introduction to computation and programming contextualized around creation and manipulation of media. The explicit goal is to interest groups typically disinterested in such courses (especially women and non-CS majors). The course was offered as a pilot in Spring 2003.

The course was a success on several measures.

- 89% of the students who enrolled earned an A, B, or C. Our "best-practice" point of comparison in the literature had a 66.4% success rate for non-majors in a specialized CS1 [33].

- The course may have met our goal of getting female students to find CS interesting, relevant, and creative where they hadn't previously, but as we'll explain, our data has some contradictions in it yet.

- We attempted direct comparisons of student performance on similar exam problems given in our traditional CS1, a prototype Engineering-focussed CS1 course, and the Media Computation class. We found that such comparisons are fraught with difficulties that rendered the results inconclusive. Nonetheless, student performance in a problem described in the literature on CS education was relatively comparable between the Media Computation and traditional CS1 students.

Our assessment of the course, and the comparison with the other courses, went beyond measures of success to explore how and why the class was successful. For example, collaboration may have played as large a role in the success of CS1315 as the context. When asked the open question on the final survey, "What absolutely must *not* change about the course?", nearly 20% of the Media Computation student respondents named the collaboration tool CoWeb [15] [19] and over 20% mentioned collaboration in general. This is significant because we know that the social context of computer science is an important inhibitor to success in CS for many students [1] [28], so we can expect that many new interventions will similarly have to address the question of collaboration.

2.1 Implementation of the Course

The choice of multimedia as the course context in this project allows us to address several of the issues identified as critical for increasing the number of women in computing, such as the lack of relevance in traditional computer science classes [1, 28], the lack of creative outlet [38], and the tediousness of traditional problems in CS classes [1]. Several of the academic units at Georgia Tech emphasize that multimedia literacy is a critical skill for educated professionals in the future [43]. Multimedia is a significant applications domain—especially true at Georgia Tech, with Turner Broadcasting, CNN, and the Cartoon Network based within a few miles of campus. Multimedia is clearly not computing for its own sake. Further, the multimedia approach allows us to give students assignments that are open-ended and creative. The emphasis on multimedia allows the class to be more concrete, more relevant, and more creative than traditional CS1 approaches.

The premises and core concepts of the Media Computation course start with all media are being published today in a digital format. Digital formats are amenable to manipulation, creation, analysis, and transformation by computer. We call these activities *media computation*. Software is the tool for manipulating digital media. Knowing how to program thus becomes a communications skill. Core computer science concepts can be introduced through media computation. For example, programs can get large and cumbersome. Abstraction

is our tool for managing program complexity and allowing programs to become even larger yet more flexible. However, computing has limitations. There are some programs that cannot complete in our lifetime, and knowing that these limitations exist is important for technological professionals.

We used the programming language *Jython* for the course. Jython[2] is a variation of the Python[3] programming language which has been designed to be easily used by novices and non-technical users. Jython is a variation of Python written in Java, which allowed ease of access to Java's Media API. We considered other languages (e.g., Java and Scheme), but selected Jython based both on survey responses from students and teachers [17] and because Jython's design is grounded in the research findings on facilitating learning and performance by novice programmers (e.g., [41], [32], [34]).We developed *JES* (Jython Environment for Students), modelling it after the popular DrScheme environment [10].

The general structure of the course was 11 weeks of media-relevant material (e.g., sound, pictures, movies, text manipulation, network, and database), with four weeks of computer science that answers questions that arose during the course of the 11 weeks. Student homework assignments started with the implementation of simple Photoshop-style filters (average score 92% with a standard deviation of 22.1). Some of the assignments invited creativity, such as the third homework which required creation of a collage, but allowed students to select whatever pictures they wanted in the collage (average score 86.3% standard deviation 26.0). These collage programs became large–the average number of lines of code (of students who gave us consent to look at their homework results) was 64 with a standard deviation of 32.4. The largest was 166 lines of code. The final homework assignment involved programmed Web access and creation of animations (average score 84.6% standard deviation 29.2). Final exam questions spanned a wide range of computing concepts, including issues of networking, databases, and computing theory.

We used our existing collaboration technology, CoWeb [20, 15], to support student question asking and sharing of media artifacts. The *constructionist* theory of learning suggests that the creation of public artifacts creates a strong potential for learning [37]. We have had good success in the past using our collaboration tools to support sharing of multimedia artifacts to encourage critique and motivate learning [26][5][47]. The CoWeb became the central place for question-asking and discussion, which is a critical role in CS1, since comfort in asking questions has been highly correlated with attrition in CS1 [45].

2.2 Evaluation Methods and Results

Our evaluation effort had three hypotheses: (1) the prototype course will improve student retention, especially among non-CS-majors and women; (2) the prototype course will improve student motivation toward computer science; and (3) the prototype course will lead to good student learning, perhaps better than in a comparable course.

Our evaluation effort was conducted under the review of the Georgia Tech Human Sub-

[2] `http://www.jython.org`
[3] `http://www.python.org`

jects Review Board, to whom our proposal and all our instruments were submitted for review as they were developed. Students were informed about the research project, the instruments, how they might be impacted, and potential risks. Assessment was organized such that the PI and the assessment team[4] designed the assessment, but the assessment team gathered and analyzed all data apart from the PI, since he was also the teacher of the course.

We gathered and analyzed several sources of data.

- We collected demographic data, which we used to address the retention question. We retained copies of course work from consenting students.
- We took surveys during the first week of the term, midway through the term, and in the last week of the term. We used the same surveys in two other classes: A section of our traditional CS1 course ($n = 127$), and a section of the prototype Engineering-oriented CS1 ($n = 75$). From these surveys we learned important information such as the significantly lower prior background in computer science and programming among the Media Computation students than the other students.
- We attempted to develop and distribute polymorphic problems (same problem, but changed in language or other minor details for the course context, a method we used successfully in [18]) that were offered in all three CS1's on exams–for example, on computing an average test grade and determining a corresponding letter grade, and on the infamous Rainfall Problem [42].
- We interviewed a sample of female students in the *Media Computation* class on their attitudes toward computer science and the course. We conducted seven interviews in total–three early in the semester, and three at the end of the term with the same students, along with another student at the end.

2.2.1 Evaluation of Retention

Retention rate is defined operationally in this study in two ways. The *completion retention rate* is defined as the ratio of the number of students completing the course to the number of students enrolled in the course. The *course success retention rate* is defined as the ratio of the number of students completing the course with a C or better to the number of students who enrolled in the course. (Several academic units at Georgia Tech require students to re-take courses if they received a D.)

The literature on both rates for introductory CS courses is rather negative. Completion rates of less than 30% are not uncommon in CS1 [40]. In an email survey on the ACM SIGCSE members list, course success retention rates of less than 50% were reported at several schools [16]. The "best-practice" comparison that we have for non-majors, in a pair-programming CS1 class, had a 66.4% course success retention rate [33] (in that study, compared to a non-majors success rate of 55.9% in a traditional CS1). Our traditional CS

[4]Graduate research assistants Andrea Forte and Rachel Fithian, and undergraduate research assistant Lauren Rich

class typically does remarkably well in comparison with the literature, with a completion retention rate of 80-90% and a course success retention rate better than 70%, so it offers a high standard to meet.

The Media Computation course completion retention rate for Spring 2003 was 98%. Our course success retention rate was 89%. During that semester, the comparison section of our traditional CS1 had a 57.1% course success rate.

2.2.2 Evaluation of Motivation

The stated goal for the prototype course was to address the disinterest in computer science, especially among non-CS-majors and women. We were concerned with three kinds of motivational factors:

- Do students find the current course engaging? An engaging course is more likely to lead to learning [27] [39] [4].
- Do students find computer science engaging?
- Would students consider taking future computer science courses? While the goal of the prototype course is to simply address issues of disinterest, positive response to this question would suggest that the media computation approach has potential for attracting new CS majors.

When asked what they like about the class in the midterm survey, the students affirmed that we're succeeding at creating a course that students recognize for its relevance, particularly for non-CS majors:

> "Very applicable to everyday life."
>
> "I dreaded CS, but ALL of the topics thus far have been applicable to my future career (and personal) plans."
>
> "The professor answers questions in class and online and is concerned about our success in the class. He also seems to understand that most of us are not engineers and most likely won't do straight programming in the future—just the way of thinking is important."
>
> "I think that we're doing things that I could actually use as an architecture major–I like dealing with pictures and sounds.

Students who have positive attitudes about CS and who enjoy computing are likely to continue in computer science. While it is too early to tell what CS courses Media Computation students will take in the future, we asked on the final survey whether or not they would be interested in taking a more advanced media computation course. 60% of the female respondents answered that they would take a second course (overall course average of 63%).

Surprisingly, when asked *on the same survey* whether or not they plan to take more CS courses, only 6% of those same female respondents responded affirmatively (9.3% overall).

Why would 60% be willing to take media computation, while only 6% planned to take more CS courses? One possible explanation is that there is currently no advanced media computation course, and, given the current selection of CS courses, Media Computation students don't see a compelling reason to take more. Another might be interpretation of the question: Students may interpret "CS courses" as "traditional CS courses" as opposed to specialized courses like Media Computation.

Students indicated that the course changed their attitudes about programming in general: they now view it as something that they could imagine themselves doing in the future. On the final survey, 30% of the students indicated that they believed that they would program again in the future.

2.2.3 Evaluation of Learning

We developed several problems that we attempted to put on all three courses' exams and quizzes. Unfortunately, a number of issues such as different rates of development and sequencing of concepts in each course, different times and numbers of exams and quizzes, and problems equating concepts in different languages prevented us from distributing the problems as uniformly as we would have liked. In addition, those questions that did make it onto multiple exams were modified by the individual course instructors and teaching assistants.

In general, we found that the different programming languages used, the differences in directions given on exams, and the kinds of examples provided on the exams created unique conditions for each course and rendered the results fundamentally incomparable. For example, the most common problem among the Engineering students' was mishandling of `ELSE` clauses. Media Computation students never used an `ELSE` clause. It's difficult to compare these results in terms of learning.

A more reliable indicator of the Media Computation students' programming achievement might be found in students' attempt to solve a classically difficult problem, such as *The Rainfall Problem.* The rainfall problem is: "Write a program that repeatedly reads in positive integers, until it reads the integer 99999. After seeing 99999, it should print out the average." At Yale in the mid-80's, Elliot Soloway gave this problem to several groups of students [42]. Of first semester CS1 students, only 14% of the students got it correct (completely correct other than syntactic errors). Only 36% of CS2 students got it correct, and only 69% of students in a junior-senior level systems programming course. This was one of the first empirical demonstrations that learning about programming was less than what we might have expected. (This finding was replicated in [30].)

Our students were not directly comparable to the Yale students. For example, we hadn't covered the same things (e.g., at the time of the exam, they had only seen `WHILE` once in class), and we had to change the wording of the problem accordingly. Nonetheless, 14 people out of 113 (12%) who took the test "got it"—by Soloway's standards that means they had a correct solution (aside from syntactic errors). With partial credit, the average on the problem was 45%.

The traditional CS1 also included a variation of the rainfall problem on *two* of their tests, but we were unable to obtain the raw problems to code ourselves and compare. On the first

test (which stipulated the use of tail recursion), the result as graded by teaching assistants was an average score of 53%. On the second test (which was open to any kind of iteration), the result was an average score of 60%.

Overall, the rainfall problem continues to be a hard problem for CS1 students. These results don't show that the Media Computation students are learning programming remarkably well, but they do show that these students aren't out of the ballpark either. The bottom line is that it's still an open question how much the Media Computation students learned about programming or CS1-relevant concepts, and how they compare to students in other classes.

2.3 Dissemination and Self-Sustained Distribution

During the time of this project, we have been very active in disseminating the results and materials, helping others to adopt the materials, and taking steps toward self-sustained distribution.

- At SIGCSE2003, Guzdial hosted a tutorial on *Media Construction Projects in Computer Science Courses* which covered the basics of doing media computation in Python, Java, and Squeak. The 11 participants all stated on the final evaluation that they would recommend this workshop to others.

- A paper on the planning and development of the Media Computation course was presented at ITiCSE 2003 [17].

- Guzdial spoke on the course at a meeting of representatives from all the CS departments of the University System of Georgia. Several schools expressed interest in adopting the course. Charles Fowler of Gainesville College, a two-year college, is offering the course as a trial this summer, with is currently offering two sections of the course. Guzdial was also invited to speak at Kennesaw College and Augusta State College, each four year colleges. Guzdial is actively working with Kennesaw on their plans to trial the course.

- Prentice-Hall has signed a contract with Guzdial to produce a book *Introduction to Media Computation*. A pre-release version of the book will be available in January 2004, with an expected release of the completed book in the 2004-2005 academic year.

3 Project Plan

While our trial results are strong, it's also clear that our support for our hypotheses about changing attitudes and learning is incomplete. At the same time, we see that similar efforts to target innovation at women and non-CS majors have similar problems and challenges that we are facing.

- For example, the CMU effort used interviews to track female students' attitudes before they began their interventions, but have not reported on any measures to check

attitudes across their female population as they implemented their curricular changes. Issues of learning and long-term impact are less significant in this work, since it targets CS majors.

- Efforts to teach the same content but in a new manner, such as in the Pair Programming work [31] [33] [44], may be interested in issues of learning (e.g., are students still learning as much with the new methods?), but will probably want to address issues of attitude and long-term impact as well.

- Work to develop new kinds of introductory CS classes (such as the Alice work [8], the graphics and VR focussed classes [29, 46], and the Web-based Scheme course [24]) will need to address questions of how much CS1 content is being learned, whether attitudes toward computer science are changing (e.g., can we expect these students to take traditional CS classes later in their academic careers?), and what the long term impact is on their relationship to computing.

Over the next few years, we can anticipate more efforts to innovate in CS education explicitly targeting women and non-CS majors. We expect that these efforts will probably reach the same place that the rest of us are at now—we can easily assess retention and WDF rates, but we are challenged to assess learning, attitudinal change, and long term impact.

We propose to develop methods to address learning, attitudinal change, and long term impact on attitudes and use of technology. We plan to develop these in the context of our course, but then to disseminate them and evaluate how they are used in new contexts. Our course is being offered to 300 students in Fall 2003 semester, with plans to offer 400 seats in the Spring 2004 semester, with similar levels of offerings in future years.

The summary of our project plan is:

- To develop a survey to measure attitudes toward computer science, which will be validated internally and with interviews.

- To develop an inventory of CS1-related concepts (knowledge units as in [2]) in a language-independent context so that we can get a sense of how well students in innovative curricula are actually learning computer science. We will validate the inventory using expert review by ABET/CSAB accreditation auditors. We will also use repeated offerings with a sample of students to establish reliability.

- To develop a survey to measure impact of introductory computing on attitudes and use of technology beyond the scope of an individual course. We plan to establish validity with interviews and use repeated offerings with a sample of students to establish reliability.

- We plan to offer workshops at ACM SIGCSE (Special Interest Group in CS Education) and ASEE/IEEE Frontiers in Education (FIE) conferences to help faculty learn how to assess curricular innovations targeting women and non-CS majors. Our plan is to discuss issues like measurement of retention and WDF rates, as well as the measures

that we are currently developing. Through our publisher, we also plan to directly seek out faculty interested in our curriculum to invite them to use our measures as well. We plan to follow up with email surveys to learn how effective the measures are at the other institutions.

All of our development and assessment will be overseen by the Georgia Institute of Technology's Human Subjects Review Board. All the researchers working on the project will be certified by the Board as capable of doing human subjects research. All our instruments will be reviewed by the board to respect the rights and safety of the subjects.

3.1 Developing a Survey to Assess Attitudes towards Computer Science

Many students' attitudes toward computer science today, especially women and non-CS majors, are abysmal as described by several studies [28] [1]. An important result of any introductory course should be an appreciation for what's interesting and useful about a topic. Students should emerge from introductory computing courses with attitudes that are positive about the usefulness of computer science and about its relevance.

Our results in the *Media Computation* course have been contradictory on this point. Our students emerge feeling that programming to manipulate samples in arrays and pixels in matrices is relevant and interesting, and worthy of additional courses—but they are still overwhelmingly disinterested in "computer science."

We plan to develop a survey that asks for fine-grained distinctions about what interests the students about their current course and its content and about what they perceive as computer science. We are interested in understanding our current Media Computation students attitudes, but also in developing a measure in order to establish a baseline about student interests that we can use in comparison as the class evolves. We want to understand issues such as:

- What is it that students find interesting about technology, the innovative course that they are taking, and its content?
- What concepts, skills, and practices are they learning that they find relevant for themselves now and for their future career?
- What are their perceptions of what computer science is? How relevant is computer science as they perceive it to their current and future careers?
- How has their attitude toward computer science changed with respect to the interventions they've participated in? How do students think about computer science with respect to the known inhibiting factors to engagement with CS that have been identified such as social setting, tediousness, and creative expression?

We plan to ask the same question in multiple ways to establish internal validity [6]. For example, we may ask students to rate their agreement with the statements "Computer

science will be useful to me in my career" and "I do not expect to find computer science relevant to my future career." If we find radical differences in these kinds of statements, we will need to re-word and re-work the questions, using our interview process as input.

Our plan to develop and validate the attitude survey is through an iterative process of interviews and survey development using methods described in [13] [3] [7]. We plan to use interviews to develop the survey questions. We will then use these surveys, and interview a sample of students who answer the surveys to ask them:

- Did we ask the right questions? Are these issues important, or are they the wrong ones?
- Are there issues that you wanted to express or that are relevant that are not addressed by these questions?
- Did the questions make sense?

During the second and third years, we plan to use repeated offerings with a sample of students, to establish the reliability of the survey.

3.2 Developing a CS1 Concepts Inventory

We have found it difficult to establish what students are learning in an innovative CS1 course. We can anticipate that many new educational developments will differ from traditional CS1 courses on features such as order of topics, kinds of topics covered, context for topics, and programming language used—as ours does. Because of differences in the order in which topics are covered, it's difficult to compare learning on midterm exams. Even when we simply focus on end of term measurements, simple programming activities are difficult to compare when there are significant language differences (e.g., as in our case, Python vs. Scheme vs. Matlab).

We propose to construct an instrument to measure concepts that one would expect to be covered in a CS1 course, as defined by the *ACM/IEEE Computing Curriculum 2001* [2], in a manner as language independent as possible. We would expect this instrument to be used at the end of a course. Example topics that we would like to include follow:

- We can ask what concepts like *iteration*, *conditionals*, and *assignment* mean, without presenting their syntax, but perhaps with asking the students to present the syntax of the language they learned (making the instrument language-independent, but the evaluation of the instrument langauge-dependent).
- We can ask about data structures such as *arrays*, *matrices*, and *trees* without requiring language explanations.
- We can ask about algorithmic complexity and similar theoretical concepts.

Our plan for content validation is to use expert review by ABET accreditation for CS programs auditors [6]. We will be asking the reviewers to look over our questions and tell us if we're asking questions that are actually relevant to CS1 concept knowledge, or are we (for example) simply asking easy to ask questions that aren't what knowledge about CS1 is about. We plan to start at a formative level [11] with our colleagues who are ABET accreditation auditors (e.g., Dr. Rich LeBlanc[5] and Mr. Mike McCracken). During our second year, we plan to seek out ABET accreditation auditors outside our institution and ask them to review the questions, to determine content validity by a panel of experts [6].

We will establish reliability by repeated use in a sample of our students in the *Media Computation* class. We will ask for a re-take of the instrument within a week with a sample size of approximately $n = 50$.

3.3 Developing a Longitudinal Evaluation of Impact on Technology Attitudes and Use

An additional area of evaluation that we plan to introduce is longitudinal tracking of Media Computation students as they continue their careers. We want to know:

- Do these students take more computer science courses?
- Do these students become computer science majors or minors?
- How does the course impact the students' use and learning of computation within their own majors? For example, do Liberal Arts majors who use Photoshop learn it more easily or use it better (by their own estimation) than students who do not take the course? Do Architecture majors learn AutoCAD more easily or use it better?

Our plan is to use a combination of email surveys and in-person or phone interviews.

- At the end of each year, 100% of the students who have taken the Media Computation course will be surveyed by email to ask about the CS classes they've taken, their majors and minors, and their use of computational tools.
- At the end of each year, some small sample of students who have taken the course (perhaps 3–5 students) will be interviewed (by phone or in person) to understand how the course has effected their later careers and interaction with computation. These data will also be used to inform further development of the course materials and to validate our instrument, using a method similar to what we plan to do for our attitude survey, e.g., asking if the questions seem appropriate, what additional issues or impacts would they consider relevant, etc.

[5]Dr. LeBlanc was also part of the development team for CC2001 [2]

3.4 Dissemination Plans and Formative Evaluation with Faculty

We have a three part plan for disseminating our measurement instruments and methods. The first part is fairly traditional: We plan to write papers for ACM SIGCSE, the international ITiCSE (Innovation and Technology in CS Education), and ASEE/IEEE FIE conferences using our instruments to evaluate our own innovations, both as a model for others to follow and to set a benchmark wo which others can compare their own innovations (using our methods, or improvements upon our methods). We have been successful at using this approach in other, related research. In our research on computer-supported collaborative learning (CSCL) (sponsored, in-part, by the PI's NSF Career award, grant #REC-955048), we developed a set of measures for how well a CSCL environment was being used by students [14]. The measures were used by others in assessing their own environments (e.g., [22]), and formed the basis of other research to develop theory explaining the kinds of findings that having these methods of comparison enabled [25]. Thus, we believe that using our methods in our own research will set a model that can be an effective force for dissemination.

The second part is to offer faculty workshops at ACM SIGCSE and ASEE/IEEE FIE[6] on methods for measuring impact of education innovations targeted at women and non-CS majors. We plan to introduce our instruments, as well as others that are well-known (such as retention and WDF rates) and methods that are developed by others over the next few years.

The third part is to use the book for the *Media Computation* course as a source of potential faculty targets. Working with our publisher, we plan to get email addresses of faculty who seek an evaluation copy of the book, and invite them (whether they use our book or not) to use our methods to assess whatever innovations they attempt to attract women and non-CS majors in CS courses. If the faculty are interested in a course such as this, it may be that they are planning to attempt some educational innovation to attempt to attract a non-traditional audience, even if they eventually choose a different path than our course. We plan to develop a manual as well as a CD with copies of our instruments to send to interested faculty. We will continue to make all our materials (for the course and for assessment) available on our development website[7], and we plan to create a project website (with a more easily remembered URL) for wider dissemination.

Our plan is to follow up with faculty who attend our workshop using email surveys on a regular (e.g., every six month basis) to learn how their use of the instruments is going, to learn about their results, and to solicit feedback on how we might improve our manual or the instruments.

4 Conclusion and Timeline

Our goal is to help ourselves and others to assess the value of the innovations that are being developed to target women and non-CS majors to improve their odds of success in introductory computer science. It's easy to show that more women stay in a course or program, or that more non-CS majors succeed where they may have withdrawn or failed

[6] Workshops are sparsely attended when offered at ITiCSE.

[7] `http://coweb.cc.gatech.edu/mediaComp-plan`

in a comparison traditional CS1 course. It's much harder to show that there is learning equivalent to a traditional CS1 going on, to show that there are changes in attitudes, and to show that there is a significant longer term impact, beyond the end of the course. We plan to use our own course and students as a testbed, but we will be active in engaging others to use our instruments and give us feedback, so that we can create measures and methods that truly benefit the community. We don't believe that the usefullness of our methods will be restricted to settings where the targets for innovation are women or non-CS majors. That's just where we'll be starting from. We expect that our methods will be useful in assessing a broad range of innovations where the objectives include CS1-level learning, impacting attitudes toward CS, and generating impact on a longer-term scale than a single class.

We believe that the potential broader impact of this work is to create an infrastructure for comparing innovations to attract under-represented groups (such as women) to computing and to create a broader base of computing-literate citizens by helping non-CS majors to be more successful at computing. By making it possible to compare innovative methods, on the basis of learning, attitudes, and longer-term impact, we believe that we create the opportunity to develop better methods and to improve existing methods. Thus, creating assessment methods to go beyond retention and WFD rate will help us to create better innovations to target under-represented groups.

We are proposing a three year effort. Our timeline follows:

- Year One: We will develop initial versions of all three of our instruments: Attitude survey, CS1 concepts inventory, and longitudinal impact survey. We will iterate on all three in the three semesters of the year. We will also conduct interviews to validate the two surveys, and we will ask colleague ABET accreditation auditors to help us with content validation of the inventory in a formative stage. We plan to continue publishing papers on our findings in the course.

- Year Two: We will continue to iterate on the instruments, but we will more actively seek to validate them and test them for reliability. We will begin proposing workshops at ACM SIGCSE and ASEE/IEEE FIE conferences during year two. Also during year two, the book for the *Media Computation* class will be published, so we will begin targeting faculty interested in the book to use our methods.

- Year Three: We will focus on reaching out to faculty using our methods (through the workshop or book) to invite their feedback and to use it in updating the instruments. As we update the measures, we plan to re-validate and re-test for reliability as needed. Again, we plan to publish our findings, with a particular emphasis on use of our methods as a way of establishing a model for others.

References

[1] AAUW. *Tech-Savvy: Educating Girls in the New Computer Age*. American Association of University Women Education Foundation, New York, 2000.

[2] ACM/IEEE. Computing curriculum 2001. *http://www.acm.org/sigcse/cc2001* (2001).

[3] Allen, D., Ed. *Assessing student learning: From grading to understanding*. Teachers College Press, New York, 1998.

[4] Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., and Palincsar, A. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist 26*, 3 & 4 (1991), 369–398.

[5] Craig, D., ul Haq, S., Khan, S., Zimring, C., Kehoe, C., Rick, J., and Guzdial, M. Using an unstructured collaboration tool to support peer interaction in large college classes. In *International Conference of the Learning Sciences 2000*. Ann Arbor, MI, 2000, pp. 178–184.

[6] Cronbach, L. Test validation. In *Educational Measurement (2nd Ed.)* (Washington, D.C., 1971), R. Thorndike, Ed., American Council on Education.

[7] Cross, K. P., and Angelo, T. P. *Classroom Assessment Techniques: A Handbook for Faculty*. NCRIPTAL, University of Michigan, Ann Arbor, 1988.

[8] Dann, W., Dragon, T., Cooper, S., Dietzler, K., Ryan, K., and Pausch, R. Objects: Visualization of behavior and state. In *ITiCSE 2003: Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, D. Finkel, Ed. ACM, New York, NY, 2003, pp. 84–88.

[9] diSessa, A. *Changing Minds*. MIT Press, Cambridge, MA, 2001.

[10] Felleisen, M., Findler, R. B., Flatt, M., and Krishnamurthi, S. *How to Design Programs: An Introduction to Programming and Computing*. MIT Press, Cambridge, MA, 2001.

[11] Flagg, B. N. *Formative Evaluation for Educational Technologies*. Communication. Erlbaum and Associates, Hillsdale, NJ, 1990.

[12] Freeman, P., and Aspray, W. *The Supply of Information Technology Workers in the United States*. Computing Research Association, New York, 1999.

[13] Gardner, H. Assessment in context: An alternative to standardized testing. In *Changing Assessments: Alternative Views of Aptitude, Achievement, and Instruction* (Boston, 1992), B. R. Gifford and M. C. O'Connor, Eds., Kluwer Academic Publishers.

[14] Guzdial, M. Information ecology of collaborations in educational settings: Influence of tool. In *Proceedings of Computer-Supported Collaborative Learning'97*, R. Hall, N. Miyake, and N. Enyedy, Eds. LEA, Toronto, Ontario, Canada, 1997, pp. 83–90.

[15] Guzdial, M. Use of collaborative multimedia in computer science classes. In *Proceedings of the 2001 Integrating Technology into Computer Science Education Conference*. ACM, Canterbury, UK, 2001.

[16] Guzdial, M. Summary: Retention rates in cs vs. institution. Message posted on acm sigcse moderated members list, Georgia Tech, April 23 2002.

[17] Guzdial, M. A media computation course for non-majors. In *Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE) 2003 Conference* (New York, 2003), ACM, ACM, pp. In–Press.

[18] Guzdial, M., and Kehoe, C. Apprenticeship-based learning environments: A principled approach to providing software-realized scaffolding through hypermedia. *Journal of Interactive Learning Research 9*, 3/4 (1998), 289–336.

[19] Guzdial, M., Ludovice, P., Realff, M., Morley, T., Carroll, K., and Ladak, A. The challenge of collaborative learning in engineering and math. In *Proceedings of IEEE/ASEE Frontiers in Education (FIE) 2001 Conference*. IEEE, Reno, NV, 2001.

[20] Guzdial, M., Rick, J., and Kehoe, C. Beyond adoption to invention: Teacher-created collaborative activities in higher education. *Journal of the Learning Sciences 10*, 3 (2001), 265–279.

[21] Guzdial, M., and Soloway, E. Computer science is more important than calculus: The challenge of living up to our potential. *Inroads – The SIGCSE Bulletin 35*, 2 (June 2003), 5–8.

[22] Hewitt, J., and Teplovs, C. *An analysis of growth patterns in computer conferencing threads*. Lawrence Erlbaum Associates, Mahwah, NJ, 1999.

[23] Hickey, T. Incorporating scheme-based web programming into computer literacy classes. In *The Scheme 2002 Workshop* (Pittsburgh, PA, October 2002).

[24] Hickey, T. Scheme-based web programming as a basis for a cs0 curriculum. In *Proceedings of the ACM SIGCSE Conference* (2004), p. Submitted.

[25] Hudson, J. M., and Bruckman, A. S. The bystander effect: A lens for understanding patterns of participation. *Journal of the Learning Sciences 13*, 2 (2004), Forthcoming.

[26] Kehoe, C. M. *Supporting Critical Design Dialog*. Unpublished ph.d. dissertation, Georgia Institute of Technology, 2001.

[27] Malone, T., and Lepper, M. Making learning fun: A taxonomy of intrinsic motivations for learning. In *Aptitude, Learning, and Instruction.*, R. Snow and M. Farr, Eds., vol. 3 of *Conative and Affective Process Analyses*. LEA, Hillsdale, NJ, 1987, pp. 223–253.

[28] Margolis, J., and Fisher, A. *Unlocking the Clubhouse: Women in Computing*. MIT Press, Cambridge, MA, 2002.

[29] Marks, J., Freeman, W., and Leitner, H. Teaching applied computing without programming: A case-based introductory course for general education. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM Press, New York, 2001, pp. 80–84.

[30] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *ACM SIGCSE Bulletin 33*, 4 (2001), 125–140.

[31] McDowell, C., Bullock, H., Fernald, J., and Werner, L. The effects of pair-programming on performance in an introductory programming course. In *The Proceedings of the Thirty-third SIGCSE Technical Symposium on Computer Science Education, 2002*, D. Knox, Ed. ACM, New York, 2002, pp. 38–42. Pair-programming (social) reduces attrition rates.

[32] Miller, L. A. Natural language programming: Styles, strategies, and contrasts. *IBM Systems Journal 20*, 2 (1981), 184–215. Languages require iteration where aggregate operations are much easier for novices.

[33] Nagappan, N., Williams, L., Ferzil, M., Wiebe, E., Yang, K., Miller, C., and Balik, S. Improving the cs1 experience with pair programming. In *Twenty-fourth SIGCSE Technical Symposium on Computer Science Education* (New York, NY, 2003), D. Joyce and D. Knox, Eds., ACM, pp. 359–362.

[34] Pane, J. F., Ratanamahatana, C., and Myers, B. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies 54* (2001), 237–264.

[35] Papert, S. Teaching children to be mathematicians versus teaching about mathematics. Ai memo no. 249 and logo memo no. 4, MIT, 1971.

[36] Papert, S. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York, NY, 1980.

[37] Papert, S. Situating constructionism. In *Constructionism*, I. Harel and S. Papert, Eds. Ablex Publishing Company, Norwood, NJ, 1991, pp. 1–11.

[38] Pfleeger, S. L., Teller, P., Castaneda, S. E., Wilson, M., and Lindley, R. Increasing the enrollment of women in computer science. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM Press, New York, 2001, pp. 386–387.

[39] Pintrich, P. R., and Schunk, D. H. *Motivation in Education: Theory, Research, and Applications*. Prentice-Hall, 1996.

[40] Roumani, H. Design guidelines for the lab component of objects-first cs1. In *The Proceedings of the Thirty-third SIGCSE Technical Symposium on Computer Science Education, 2002*, D. Knox, Ed. ACM, New York, 2002, pp. 222–226. WFD (Withdrawl-Failure-D) rates in CS1 in excess of 30

[41] Soloway, E., Bonar, J., and Ehrlich, K. Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM 26*, 11 (1983), 853–860.

[42] Soloway, E., Ehrlich, K., Bonar, J., and Greenspan, J. What do novices know about programming? In *Directions in Human-Computer Interaction*, A. Badre and B. Schneiderman, Eds. Ablex Publishing, Norwood, NJ, 1982, pp. 87–122.

[43] Soloway, E., Guzdial, M., and Hay, K. E. Reading and writing in the 21st century. *EDUCOM Review 28*, 1 (1993), 26–28.

[44] Williams, L. A., and Kessler, R. R. The effects of 'pair-pressure' and 'pair-learning' on software engineering education. In *Proceedings of the Thirteenth Conference on Software Engineering Education and Training*. 2000, pp. 59–65.

[45] Wilson, B. C., and Shrock, S. Contributing to success in an introductory computer science course: A study of twelve factors. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM, New York, 2001, pp. 184–188.

[46] Zimmerman, G. W., and Eber, D. E. When worlds collide! an interdisciplinary course in virtual-reality art. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM Press, New York, 2001, pp. 75–79.

[47] Zimring, C., Khan, S., Craig, D., Haq, S.-u., and Guzdial, M. Cool studio: Using simple tools to expand the discursive space of the design studio. In *Design Thinking Research Symposium*. MIT, Cambridge, MA, 1999.

5 Budget Justification

To fund this effort, this proposal includes a budget request for a three year project.

- One summer month of the PI's time in all three years, to build the instruments, to plan and conduct the workshops, and to work on dissemination.
- Two graduate student research assistants. One will focus on development of the instruments and the manual/CD for faculty, as well as the formative evaluation of the instruments. The other will focus on use of the instrument and analysis of the data that we gather from it, including testing for reliability and validity.
- Our costs for these personnel include fringe, computing charges (for support within the College of Computing), and graduate student tuition,
- Travel support for attending conferences to disseminate materials and results and conduct workshops.
- Materials and supplies, to support the development and evaluation effort.

6 Facilities

The College of Computing maintains a variety of computer systems in support of academic and research activities. These include more than 50 Sun, Silicon Graphics, and Intel systems used as file and compute servers, many of which are quad-processor machines. In addition, there are more than 1,000 workstation class machines from Sun, Silicon Graphics, Intel, and Apple especially for student use. A number of specialized facilities augment these general-purpose computing capabilities. The hardware that will be purchased for this project will be of similar quality to what the students use, for testing purposes, but will be set up to facilitate development.

The Graphics, Visualization, and Usability (GVU) Center houses a variety of graphics and multimedia equipment, including high-performance systems from Silicon Graphics, Sun, Intel, and Apple. The affiliated Multimedia, Computer Animation, Audio/Video Production, Usability/Human Computer Interface, Virtual Reality/Environments, Electronic Learning Communities, Computational Perception, Software Visualization, Biomedical Imaging, Collaborative Software, and Future Computing Environments labs provide shared facilities targeting specific research areas. These laboratories' equipments will be of use in developing our multimedia projects.

PI Guzdial is the Director of the Collaborative Software Lab, affiliated with GVU. The Collaborative Software Lab has a bank of ten servers supporting our experimental software for studying computer-supported collaborative learning. In addition, we have three Linux workstations, two NT workstations, and two Apple workstations used for development. The focus of the Collaborative Software Lab is on facilitating multimedia collaboration, so multimedia facilities available include a high-end Alesis keyboard, projection facilities, a Canon digital video camera, and a Nikon digital camera.

All of the College's facilities are linked via local area networks which provide a choice of communications capabilities from 10 to 1000 Mbps. The College's network employs a high-performance OC12C (622 Mbps) ATM and GigabitEthernet (1000 Mbps) backbone, with connectivity to the campus ATM network provided via OC12C. The primary campus Internet connection is provided by a direct 100 Mbps link to the service provider's Atlanta switching center, augmented by OC3C ATM and OC12C connections, respectively, to the NSF vBNS (very high performance Backbone Network Service) and Abilene research networks. Georgia Tech is also leading southern regional gigabit network efforts (SoX.net, the Southern Crossroads) as part of Internet2.

Additional computing facilities are provided to the Georgia Tech campus by the Institute's Office of Information Technology (OIT), including five public-access clusters of Sun, Apple, and Dell workstations, a collection of Sun multi-processors which are treated as a single computational resource via login load sharing, and various mainframes.