Motivation and Nonmajors in Computer Science: Identifying Discrete Audiences for Introductory Courses

Andrea Forte and Mark Guzdial

***Abstract*—Traditional introductory computer science (CS) courses have had little success engaging non-computer science majors. At the Georgia Institute of Technology, Atlanta, where introductory CS courses are a requirement for CS majors and nonmajors alike, two tailored introductory courses were introduced as an alternative to the traditional course. The results were encouraging: more nonmajors succeeded (completed and passed) in tailored courses than in the traditional course, students expressed fewer negative reactions to the course content, and many reported that they would be interested in taking another tailored CS course. The authors present findings from a pilot study of the three courses and briefly discuss some of the issues surrounding the tailored courses for nonmajors: programming, context, choice of language, and classroom culture.**

***Index Terms*—Classroom culture, computer science (CS), motivation, nonmajors, programming.**

I. Introduction

LEARNING to program is a difficult accomplishment for computer science (CS) and non-CS majors alike, but students who have elected to major in CS have a motivational advantage over their nonmajor peers in introductory CS courses. Majors have chosen to learn programming, whereas nonmajors are often required to do so regardless of their personal opinions of its value or utility. Lack of student motivation may contribute to some known problems in CS education, such as high *D-grade withdrawal and failure* (DWF) rates reported by many institutions [1], [2] and low enrollment among females and minorities [3]–[5].

As the predicament of nonmajors in introductory CS courses receives more attention from educational researchers [6]–[10], CS departments find themselves balancing their interest in preparing CS majors for a long curriculum sequence with that of welcoming students from other disciplines whose latent interest in CS may go untapped. Clearly, one class cannot fit all. In traditional programming-first introductory courses, generic problems are often used to introduce foundational programming skills and concepts that will become important to students as they progress to more advanced courses and attempt to solve more complex problems. Unfortunately, since the relevance of basic programming skills is often revealed in later courses, nonmajors who are compelled or decide to take CS1 seldom experience the gratification of using their programming skills in a personally meaningful context. By failing to connect programming and CS concepts with students' diverse interests and backgrounds, traditional introductory courses fail to motivate many students and may even discourage them from pursuing further CS learning.

In a tailored CS1 course, students encounter programming and CS concepts in the context of their chosen discipline, or something related to it. Tailored CS1 courses, designed to accommodate a variety of student interests and backgrounds, can offer a more motivating and engaging context for the learning of programming and CS concepts than traditional course implementations. Stronger motivation and sustained engagement among non-CS majors in CS1 will lead to less anxiety about the course, increased interest in CS, higher achievement, and, eventually, positive changes in nonmajors' perceptions of the discipline.

At the Georgia Institute of Technology (Georgia Tech), Atlanta, introductory computer science is a requirement for all undergraduates. Traditionally, all students have taken the same course, but in spring semester 2003, three separate CS1 courses were offered.

1) *CS1321 Introduction to Computing* is the traditional CS1 course at Georgia Tech. It is taught in lecture sections of 200 students and more, with recitation groups of 20 to 30, using the programming language Scheme with a focus on design process and documentation (using [17]). It covers data structures through lists and trees, including binary search trees. The following is an example of the kinds of problems students are assigned in this traditional course: define several functions such as the function called *list-pick*, which consumes a nonnegative integer and a list and returns the nth position element out of the list, starting from the leftmost element in position zero of the list.
2) *CS1371 Introduction to Computing for Engineers* is an introductory computing course tailored for students of engineering. The section studied uses the programming language MATLAB, includes an introduction to Java, and is taught in a 75-person lecture with recitation

Manuscript received December 4, 2004; revised July 20, 2004. This work was supported in part by the National Science Foundation, Computer and Information Science and Engineering Directorate, Educational Innovations (CISE EI) and Division of Undergraduate Education, Course, Curriculum, and Laboratory Innovations (DUE CCLI) programs under grants, by the Georgia Institute of Technology (Georgia Tech), Atlanta, under the Al West Fund, and by the College of Computing and the Graphics, Visualization and Usability (GVU) Center of Georgia Tech.

The authors are with the Graphics, Visualization and Usability (GVU) Center, College of Computing, Georgia Institute of Technology, Atlanta GA 30332-0760 USA (e-mail: aforte@cc.gatech.edu; guzdial@cc.gatech.edu).

Digital Object Identifier 10.1109/TE.2004.842924

groups of 20 to 30. It covers the same kinds of base content and data structures as CS1321. The following was one of the first assignments in this class: given a matrix A, create several new matrices based on A, such as a diagonal array with the values in the diagonal of A, a matrix with the same values as A except there are 0's in the diagonal, and a matrix whose values are three times the corresponding values in A.

3) *CS1315 Introduction to Media Computation* was given a trial in the spring 2003 semester. It was tailored for non-CS and nonengineering majors and taught in a section of 120 students with recitation groups of 20 to 30, using the programming language Python. Students learned how different media are encoded electronically and how to write basic code to manipulate digital images, audio, video, and text. Six homework assignments required students to program with each requiring the creation of original media. A midterm assignment required students to write code that would create a 7.5 × 9-in collage from (at least) one image, used a minimum of three times with a variety of manipulations, such as scaling, negative image, color shifting, cropping, and any other visual effects they could produce using code.

In order to determine how both the tailored and traditional CS1 courses impacted student motivation and perceptions of CS, researchers examined DWF rates and surveyed students' attitudes toward CS and programming at the beginning, midpoint, and end of the semester. Interviews conducted with students both in the media computation course during spring 2003 and in the traditional CS course during the following summer session provide qualitative insight into the experiences of individual students. The results of the trial course offerings at Georgia Tech are followed by a discussion of the characteristics of CS1 students in each course that distinguishes discrete audiences for CS, the role of programming in nonmajors CS courses, and finally, potential avenues for building on this pilot study.

II. Data Collection

Surveys and interviews were carried out by a research team lead by the first author. Course instructors did not have access to student surveys or interviews until the semester was over. Initial survey response rates ranged from approximately 60% to 70% in the three courses; however, the study suffered from high attrition, and response rates only reached 20% to 45% for the final survey. Frequent responses to survey questions were used to develop codes for categorizing short answers on each survey. Two coders independently tested the codes on a sample of answers and, after some minor changes to the code definitions, reached nearly 90% intercoder reliability. A single coder applied the final codes to the entire set of initial survey answers. Each survey answer could map to multiple codes so that percentages of commonly occuring categories can sum to over 100. Results presented in this paper include only the most common categories.

Seven interviews were conducted at midterm and shortly before finals with female students in the media computation course to establish whether the course was engaging female students, who are traditionally underrepresented in computer science programs. This study defines engagement as perceived authenticity—the student's perception that the material was relevant to the student's future needs [18]. This definition of engagement was selected because it directly addresses a common deterrent to studying CS—perceived irrelevance [4]. Another set of seven interviews was conducted during the summer session immediately following the pilot study to capture the experiences of female students in the traditional course. Female students were chosen because a wealth of literature exists on the underrepresentation of women in computing and on the likely reasons that women generally demonstrate low levels of interest in CS. The authors' assumption is that many of the reasons women do not choose to participate in computing will also apply more broadly to other groups of students who do not exhibit interest in CS. If women's experiences in tailored CS1 courses differ from those in traditional CS1 courses, many other students' experiences will likewise differ.

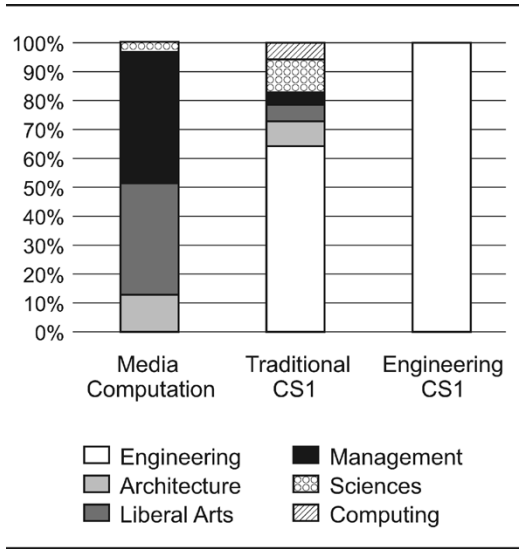

Fig. 1. Composition of courses by students' college of major.

An attempt to compare student learning among courses failed. Common exam problems were distributed to instructors in the courses. The results were incomparable because of differences in course pace, scheduling of exams, modifications made by individual instructors, and deep differences in course content (e.g., nested conditionals are common in Scheme and Python but rarely are used in MATLAB). In general, students in all three courses seemed to exhibit many of the same misconceptions of programming concepts described in CS education literature [22]. The question of actual programming achievement is one that will continue to be addressed in future evaluation efforts.

III. Student Characteristics

Students were surveyed during the first week of class to establish relevant characteristics of each group, such as programming experience, perceptions of CS, and general demographics. The distribution of this initial survey data sample by the students' college of major is represented in Fig. 1. Because the data was collected in the spring semester when virtually no CS majors

TABLE I
SELECTED STUDENT RESPONSES TO THE QUESTION "WHAT IS COMPUTER SCIENCE TO YOU?"

	Don't Know	How Computers Work	Program-ming	Using Computers for a Purpose	Scary/ Difficult	Required Class	The Internet
Traditional CS1	4.9%	6.1%	51.2%	19.5%	8.5%	7.3%	0.0%
CS1 for Engineers	11.4%	18.2%	45.5%	31.8%	2.3%	4.5%	0.0%
Media Computation	4.7%	7.0%	48.8%	16.3%	10.5%	15.1%	8.1%

take introductory CS, the populations sampled were comprised almost exclusively of nonmajors. The percentage of male and female students in each course sample varied considerably; media computation had by far the highest representation of females, with an average of 65% for all three surveys (early, mid, and end of semester), followed by traditional CS1 at 37% female, and CS1 for engineers at 17%.

Fewer media computation students reported programming experience than peers taking traditional CS1 and CS1 for engineers. Of those who reported some experience, traditional CS1 and CS1 for engineers students reported an average of 1.6 and 1.7 years of experience, respectively, whereas no media computation student reported more than one year of experience. The question "What is computer science to you?" was asked of all three classes' students, and the answers are summarized in Table I. Media computation students were unique in defining CS in terms of the Internet and were most likely to define CS as something difficult or frightening. Engineers were more likely than students in either of the other courses to think of CS as the study of how computers work.

IV. MEASURING STUDENT MOTIVATION AND ENGAGEMENT

Two indicators of students' attitudes toward a class is the rate at which they drop the course and the rate at which they ultimately succeed in passing it. From fall semester 2000 through fall 2002, DWF rates for CS1 at Georgia Tech averaged about 28%. Other institutions have reported DWF rates ranging from 25% to 50% [1], [2]. Both drop rates and DWF rates for the courses studied in spring 2003 indicated that nonmajors responded favorably to the tailored courses. Drop rates were radically lower in the tailored courses. The combined DWF rate for students in the section studied of the traditional CS1 course was 42%—over double that of CS1 for engineers and nearly four times as high as media computation. (See Table II.) The overall DWF rate for the traditional CS1 course in the term under study was 37.5%, which was high compared with its average 29.7% DWF rate for the previous year, before tailored courses were introduced.

Survey responses also suggest that nonmajors in the tailored courses enjoyed the CS1 experience more than their peers in traditional CS1. For example, at midterm, students were asked what they liked best about the courses they were taking. When responses to the question "What do you like best about this class?" were analyzed, some notable differences emerged. (See Table III.) In traditional CS1, 15.2% of the students indicated that they found nothing enjoyable about the traditional course; only about 3% of respondents in each of the tailored CS1 courses responded with such cynicism. Not surprisingly, the type of content that was specifically tailored to each audience was mentioned by over half of the media computation respondents (Internet/media) and nearly one third of the engineering respondents (data analysis/visualization). Engineers were more impressed with their new programming skills than either of the other two groups, probably because engineering students are more likely to need and use programming outside of class than other non-CS majors. (Across all three courses, nearly 28% of engineering students reported programming outside of class; the only other two majors who reported programming outside of class were liberal arts students at 8% and business students at 5%.) Finally, even when asked to specify something positive about the course, over 12% of the respondents in the traditional CS1 commented on programming or the language used for the course being useless or disagreeable.

TABLE II
D-GRADE WITHDRAWAL AND FAILURE (DWF) RATES IN EACH COURSE

	DWF Rate	Drop Rate
Traditional CS1	42.9%	10.1%
CS1 for Engineers	18.7%	11.1%
Media Computation	11.5%	2.5%

To better understand what students found engaging about each course, the authors asked them to name something they had learned that they found particularly useful, interesting, or surprising. (See Table IV.) Again, students surveyed in the traditional CS1 course returned the highest percentage of unenthusiastic responses to the course material—18.2% reported that they had learned nothing interesting, useful, or surprising in CS1 by midterm. Over a quarter of the engineering students surveyed reported that useful content was their favorite aspect of the tailored CS1 for engineers, but not one student in traditional CS1 named the utility of course content as a favorite feature.

For many reasons, one would expect media computation students to be the most difficult audience to motivate and engage in CS. Programming activities are not generally emphasized in their fields of study; their curricula do not share many cognate courses with CS, and they reported low levels of prior programming experience, which studies have shown is a good predictor of success in CS1 [11], [12]. Survey responses, homework assignments, and interviews with media computation students indicated that students were not only enjoying the material, but also taking advantage of the creative aspects of the course by using their new skills outside of the class. Students reported writing programs to play popular songs backward and alter personal photographs. Students often turned in homework assignments that included far more complex code than was

TABLE III
SELECTED STUDENT RESPONSES TO THE QUESTION "WHAT DO YOU LIKE BEST ABOUT THIS CLASS?"

	Nothing	Media/ Internet	Graphing/ Data Manipulation	Program-ming	General IT Skills	Negative Response to Language/ Programming
Traditional CS1	15.2%	0.0%	0.0%	6.1%	15.2%	12.1%
CS1 for Engineers	3.2%	0.0%	32.3%	25.8%	9.0%	0.0%
Media Computation	2.2%	56.2%	0.0%	11.2%	0.0%	0.0%

TABLE IV
SELECTED STUDENT RESPONSES TO THE QUESTION "WHAT IS THE MOST INTERESTING, USEFUL, OR SURPRISING THING YOU'VE LEARNED IN THIS COURSE?"

	Don't Like it/ Nothing	Enjoy Content	Content is Useful
Traditional CS1	18.2%	12.1%	0.0%
CS1 for Engineers	12.9%	16.1%	25.8%
Media Computation	0.0%	21.3%	12.4%

required, and in interviews several media computation students described a playful attitude toward programming. On the final survey, media computation students were asked whether they plan to take more CS courses. Only 6% responded affirmatively; however, when asked whether they would be interested in taking advanced media computation (which is not currently offered), the percentage of affirmative responses rose to over 60%. Clearly, media computation has captured the interest of many students who, otherwise, would not choose to pursue CS learning.

V. PROGRAMMING FIRST, SECOND, OR NOT AT ALL

The reluctance of many students to learn programming, combined with the increasingly marginal role of programming in general computer use, has led some institutions to eliminate programming altogether in introductory computing courses for nonmajors. Arguments against programming-first CS1 implementations for nonmajors suggest that emphasis on programming leads to negative perceptions of CS as a discipline and that programming is incompatible with many students' needs and interests and uses concepts that are not transferable to other domains [10], [13]. Accordingly, some institutions have developed breadth-first or application-centered CS0 courses in which students become acquainted with problems from interesting areas of CS that normally lie beyond the scope of CS1, or practice problem solving, using a variety of applications without actually programming.

Tailored CS1 courses call for a fundamentally different perspective on programming for nonmajors pursuing a liberal education. First, programming itself is not viewed as inherently uninteresting or inaccessible to nonmajors; instead, traditional presentations of programming are viewed as inappropriate and lacking motivation for many students. Tailored courses offer audience-appropriate contexts for engaging in programming tasks. Second, just as traditional literacy involves both reading and writing, computational literacy necessarily involves the ability to create computational artifacts, not simply use them [14], [15]. By concealing programming from nonmajors by removing it from their experience altogether, their computer-mediated modes of expression are effectively limited to those dictated by the creators of the applications they use.

Understanding computer language represents an increasingly important facet of general education for those students who will not pursue CS as a career but who wish to understand their culture and environment better. Consider the role of calculus in general education. Calculus is the study of rates and generally considered part of a liberal education. Computer science, the study of process, is equally (if not more) important to a wide variety of fields [16]. Finally, because one of the objectives of introductory CS is to generate interest in computer science, even to attract potential CS majors, fundamental programming skills are essential to the students' ability to pursue this interest further. If non-CS majors do not encounter programming in introductory CS courses, many of them will never encounter it at all—for some students, that means eliminating the prospect of ever discovering an interest in or aptitude for programming.

Finally, the observation has been made that the tailored courses described in this study cover concepts that are less abstract than those covered in traditional CS1 and that this difference offers a possible explanation for the differences in DWF rates and attitudes in the three courses. The authors agree—the concepts in the tailored courses are less abstract and are thus less difficult to learn. While abstract knowledge is clearly powerful, it is not clear that it can be taught well in introductory classes [21]. CS classes that are too abstract may be a cause of lower retention rates [4], and the advantages of abstract CS introductory classes for nonprogrammers are unclear since such approaches have not resulted in demonstrable transfer of knowledge (including problem-solving skills) to new contexts [19]. The question of whether the tailored courses deal with *enough* difficult concepts and abstraction might be addressed through a longitudinal study of student success later in their careers.

VI. TAILORING CS1

The procedure for tailoring a CS1 course involves identifying discrete audiences and recognizing interests that lend themselves to learning about computation. The selection of an appropriate introductory programming language is a long-debated, fundamental concern. For engineers, whose professions may require modest amounts of coding, language selection is an important aspect of tailoring; many students are eager to learn a language that will help them perform their jobs. One engineering student who took the traditional CS1 course in the summer session 2003 reported in an interview that, "I think if anything [should be required], it should be the second one

[CS2], like Java. A lot of people know that Scheme isn't used and so a lot of people just want to copy the homework and get through it. They're not very motivated. I've heard that Java and C and all those, they are different, and I'd like to learn C and Java and all that." Language choice is not the only element to consider when tailoring CS1 for engineers. As revealed in surveys, engineering students place great importance on data manipulation and visualization tools. Graphing equations and visualizing large data sets is a context for programming that seems to appeal to engineers [20].

For students concerned with communication, the choice of language is less central to motivation; for these students, providing a meaningful and expressive context for programming is essential. All media are being published today in digital formats that are amenable to manipulation, creation, analysis, and transformation by computer. In the context of images, text, sound and video, knowing how to program becomes a communication skill that many students enjoy. As one student expressed in a survey, "I think that we're doing things that I could actually use as an architecture major—I like dealing with pictures and sounds."

The authors believe that the content of tailored CS1 courses is only partially responsible for their positive reception among nonmajors. The culture of the courses also contribute to students' success and confidence. Making sure that students feel comfortable asking questions in the classroom is not just conventional wisdom; a study on factors contributing to success in CS1 found that the best predictor of success in introductory computer science courses is the students' comfort level [12]. In tailored courses, students learn CS1 material with peers who share similar backgrounds, goals, and frustrations. One media computation student revealed that when posting questions on the class website "...sometimes I ask a specific question and he [the professor] asks for clarification. I would feel different in a class with a bunch of CS majors. But since we are there with a bunch of management—other students—it's kind of more comfortable."

Tailoring CS1 courses also involves a reassessment of learning objectives. In terms of programming, traditional CS1 at Georgia Tech initiates students into the practice of software development. CS1 for engineers requires students to become adept at writing relatively short pieces of code from scratch. Media computation students are expected to comprehend, combine, and modify existing code to achieve desired outcomes. Naturally, assignments should support the unique objectives of each course and should relate to the context in which the course material is being presented. To explain that digital images are encoded as matrices of numerical data and then ask students to manipulate such matrices is insufficient. Instead, assignments should be directly related to the tailored context in which matrices are presented. A contextual assignment may involve manipulating or analyzing a given digital image in a way that utilizes students' new knowledge of matrices.

VII. Conclusion and Future Work

Three discrete audiences for CS1 have been identified at Georgia Tech, and three corresponding tailored courses have been designed and offered with encouraging outcomes. Initial results suggest that students in tailored courses are more likely to "stick it out," less likely to dislike introductory CS material, and, in the case of media computation, more likely to consider further CS learning. Still, many questions remain unanswered. For example, to what degree can fundamental programming achievement of students in traditional and tailored courses be compared? How much transfer of knowledge and skills from introductory computing courses occurs in later contexts? Would similar results would occur at other institutions?

The effects of differing classroom cultures on student retention and performance are also unclear. Practices that are likely to affect student motivation and attitude in general may especially affect students from fields of study in which standards of competition and collaboration may not harmonize with those of traditional computer science culture. Future evaluation will focus on identifying classroom practices that contribute to nonmajors' favorable reactions to and improved learning of introductory computer science skills and concepts.

Acknowledgment

The authors thank all the students who helped create the Media Computation class, those who assisted with the evaluation, the students in all three courses who volunteered to participate in our studies, and the National Science Foundation, which provided funding for this effort.

References

[1] N. Herrmann, J. C. Popyack, P. Zoski, C. D. Cera, R. N. Lass, and A. Nanjappa, "Redesigning introductory computer programming using multi-level online modules for a mixed audience," in *Proc. 34th ACM Special Interest Group Computer Science Education (SIGCSE) Tech. Symp. Computer Science Education*, 2003, pp. 196–200.

[2] N. Nagappan, L. Williams, M. Ferzil, E. Wiebe, K. Yang, C. Miller, and S. Balik, "Improving the CS1 experience with pair programming," in *Proc. 34th ACM Special Interest Group Computer Science Education (SIGCSE) Tech. Symp. Computer Science Education*, 2003, pp. 359–362.

[3] AAUW Educational Foundation Commission on Technology Gender and Teacher Education, "Tech-Savvy: Educating Girls in the New Computer Age," American Association of University Women Educational Foundation, Washington DC, 2000.

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

[5] National Science Foundation, "Women, Minorities and People With Disabilities in Science and Engineering: 2000," NSF, Arlington, VA, 2000.

[6] C. Eastman and S. Weiss, "Accommodating diversity in computer science education," in *Teaching the Majority: Breaking the Gender Barrier in Science, Mathematics, and Engineering*, S. V. Rosser, Ed. New York: Teacher's College Press, 1995, pp. 160–168.

[7] D. Joyce, "The computer as a problem solving tool: a unifying view for a nonmajors course," in *Proc. 29th SIGCSE Tech. Symp. Computer Science Education*, 1998, pp. 63–67.

[8] N. Kock, R. Aiken, and C. Sandas, "Using complex it in specific domains: Developing and assessing a course for nonmajors," *IEEE Trans. Educ.*, vol. 45, no. 1, pp. 50–56, Feb. 2002.

[9] J. Comer and R. Roggio, "Teaching a Java-based CS1 course in an academically-diverse environment," in *Proc. ACM Special Interest Group Computer Science Education (SIGSCE)*, 2002, pp. 142–146.

[10] M. Urban-Lurain and D. Weinshank, "Is there a role for programming in nonmajor computer science courses?," in *Proc. 30th ASEE/IEEE Frontiers in Education Conf.*, vol. 1, Kansas City, MO, Oct. 18–21, 2000, pp. T2B/7–T2B/11.

[11] D. Guerer and T. Camp, "Investigating the Incredible Shrinking Pipeline for Women in Computer Science: Final Report," 2001.

[12] B. C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course: A study of twelve factors," in *Proc. SIGSCE 2001*, 2001, pp. 184–188.

[13] J. Marks, W. Freeman, and H. Leitner, "Teaching applied computing without programming: a case-based introductory course for general education," in *Proc. 32nd SIGCSE Tech. Symp. Computer Science Education*, 2001, pp. 80–84.
[14] A. Forte and M. Guzdial, "Computers for communication, not calculation: media as a motivation and context for learning," in *Proceedings of Hawaiian International Conference of Systems Sciences*, 2003.
[15] A. diSessa, *Changing Minds: Computers, Learning and Literacy*. Cambridge, MA: MIT Press, 2000.
[16] M. Guzdial and E. Soloway, "Computer science is more important than calculus: The challenge of living up to our potential," *ACM SIGCSE Bull.*, vol. 35, no. 2, pp. 5–8, 2003.
[17] M. Felleisen, R. Findler, M. Flatt, and S. Krishnamurthi, *How to Design Programs: An Introduction to Programming and Computing*. Cambridge, MA: MIT Press, 2001.
[18] D. W. Shaffer and M. Resnick, "'Thick' authenticity: new media and authentic learning," *J. Interactive Learning Res.*, vol. 10, no. 2, pp. 195–215, 1999.
[19] D. B. Palumbo, "Programming language/problem-solving research: a review of relevant issues," *Rev. Educational Res.*, vol. 60, no. 1, pp. 65–89, 1990.
[20] D. Kaplan, "Courses for nonmajors: Teaching computation to undergraduate scientists," in *Proc. 35th SIGCSE Tech. Symp. Computer Science Education*, New York, 2004, pp. 358–362.
[21] S. Turkle and S. Papert, "Epistemological pluralism and the revaluation of the concrete," in *Constructionism*, I. Harel and S. Papert, Eds. Norwood, NJ: Ablex, 1991, pp. 161–192.
[22] E. Soloway and J. Spohrer, *Studying the Novice Programmer*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1989.

Andrea Forte received the M.L.I.S. degree from the University of Texas at Austin, in 1998. She is currently working toward the Ph.D. degree at the College of Computing, Georgia Institute of Technology, Atlanta.

The main area of her research is learning sciences and technology, with a focus on social contexts for learning.

Mark Guzdial received the Ph.D. degree in education and computer science and engineering from the University of Michigan, Ann Arbor, in 1993.

He is an Associate Professor in the College of Computing, Georgia Institute of Technology, Atlanta. His research focuses on learning sciences and educational technology, especially in the context of student-composed multimedia.