CS 0.5: A Better Approach to Introductory Computer Science for Majors

Robert H. Sloan
U. Illinois at Chicago
sloan@uic.edu

Patrick Troy
U. Illinois at Chicago
troy@uic.edu

ABSTRACT

There are often problems when students enter a course with widely different experience levels with key course topics. If the material is covered too slowly, those with greater experience get bored and lose interest. If the material is covered too quickly, those with less experience get lost and feel incompetent. This problem with incoming students of our Computer Science Major led us to create CS 0.5: an introductory Computer Science course to target those CS majors who have little or no background with programming. Our goal is to provide these students with an engaging curriculum and prepare them to keep pace in future courses with those students who enter with a stronger background.

Following the lead of Mark Guzdial's work on using media computation for non-majors at Georgia Tech, we use media computation as the tool to provide this engaging curriculum. We report here on our experience to date using the CS 0.5 approach with a media computation course.

Categories and Subject Descriptors

K.4 [**Computers and Education**]: Computer and Information Sciences Education

General Terms

Human factors

Keywords

Computer science major, retention, CS 1, curriculum

1. INTRODUCTION: THE PROBLEM

There is a problem with the first two years of the computer science (CS) major: very high attrition. The *annual* attrition rate among freshman and sophomores majoring in CS in the U.S. has been reported to average 19%, and at some schools to be 66% [4]. At our school, we have observed an attrition rate from freshman to sophomore year of 40–50%, although our sophomore to junior year attrition has been fairly low.

Today this attrition rate is especially troubling because CS departments in the U.S. have experienced a very substantial decrease in the enrollment in the CS major (see, e.g., [3, 21, 23, 24]). Thus many, perhaps most, CS departments currently have too few majors.

The attrition rate for CS majors is much worse than the attrition rate for other majors that have similar requirements for freshman—*except for freshman courses in the major itself.* Therefore, we believe the problem must be the introductory course sequence in CS. Indeed, international studies of programming performance [14], declining retention rates [8], and student failure rates as high as 50% [19] show that CS departments today are *not* successfully attracting a wide range of students to introductory CS courses, nor fully engaging those students who do enroll.

Furthermore, the introductory course sequence seems to be serving the needs of women even more poorly than the needs of men (who make up a heavy majority of CS majors). The enrollment of women in introductory CS classes keeps falling, and retention rates for women are even worse than for men (see, e.g., [2, 12, 13]). Women reportedly tend to avoid CS (and IT in general) in part because they find CS courses "too boring" and "overly technical," with little room for creative "tinkering" [1, 13]. At a session devoted to increasing the enrollment of women at a recent ACM SIGCSE conference, speakers reported that women CS majors were often surprised by how much "creativity" there was in later CS courses, since introductory courses did not highlight this aspect of CS [17]. Women CS majors, in contrast to men, are mostly interested in real applications of computing, and not simply computing for its own sake [1, 13].

Addressing the factors that cause women to avoid CS, may also increase the number of men—in particular, men who currently are not retained in the major. Stephen W. Director, Chair of the U.S. Engineering Dean's council, has testified that, "Women in engineering programs are the 'canaries in the coal mine'. If women do well in a program, most likely everyone else will also do well." [6].

An additional problem is severe underrepresentation of African Americans and Hispanics among CS majors. This problem has been less studied than the shortage of women, perhaps because at many schools there are so few CS majors from these groups that it is difficult to make any statistically meaningful statements about their numbers.

2. PROPOSED SOLUTION: OVERVIEW

We hypothesize that providing two, rather than one, starting points for the CS major will significantly improve the retention of CS majors in general, and women in particular. Thus, rather than there being a single "CS 1" that is taken as the first course in the CS major by every major, there would be two courses: "CS 0.5" and "CS 1".

Briefly, the idea of CS 0.5 is that most incoming majors will take CS 0.5; a substantial minority (perhaps 20–30%) will go directly into a mildly aggressive version of a traditional CS 1 course. The objectives of CS 0.5 are to give students a moderate amount of what we might call "programming maturity," by way of analogy with what our mathematician colleagues refer to as mathematical maturity, and to instill both enthusiasm for, and general knowledge about computer science (which is of course *not at all the same thing as programming*). At our school, the typical incoming CS major has relatively little (and occasionally no) background in programming; but a substantial minority have moderate or substantial background in programming.

In Section 2.1 we describe why we think this CS 0.5 approach is a good idea, and then in Section 2.2 describe why we chose to use a lightly modified version of Guzdial's Media Computation course for non-CS-majors [7, 9, 10] as our CS 0.5.

2.1 Why CS 0.5?

First, let us point out that it is common to have many entry points for college-level study of established technical disciplines. At our school there are several "first" courses in each of Mathematics, Physics, and Chemistry. In all three of those cases there are at least two courses that are fairly common choices for majors in the discipline to take as their first course.

If anything, beginning CS majors have a wider range of backgrounds than beginning chemistry or math majors. Certainly at our school, we have: (1) a majority of CS majors who have had relatively little, occasionally zero, programming background, and (2) a substantial minority with a moderate to significant programming background. We imagine that this mix of beginning CS majors is typical for the large number of schools that have selective but not highly selective admissions.

These two groups create trouble when they are placed in the same classroom for the same first-semester course: the majority with little background are intimidated, and the minority with substantial background are bored. We suspect that the intimidated students with little background have a huge attrition rate from the CS major. We want to put these students in their own course, to create a welcoming environment that will encourage them to continue in CS. This should, we believe, level the playing field when all students are recombined in the aggressively paced traditional CS 1 course and improve the retention rate of these less experienced CS majors.

There are also pedagogical reasons for making the split. One reason is simply that it is very difficult to teach an introductory course to students with extremely diverse backgrounds in the subject.

Furthermore, the amount of material to be covered in the introductory sequence is currently an awkward size: too large for our traditional CS 1 followed by a CS 2 course on data structures, but not quite large enough for a standard three-course sequence (despite concerted efforts in this direction [18]).

Starting the CS 1 course with students who all have a basic knowledge of programming can provide a robust solution to this dilemma.

2.2 Media Computation as CS 0.5

Having decided that one should have a CS 0.5 that comes before a mildly aggressive traditional CS 1, one next has to decide what to teach in CS 0.5. As we said above, we want to provide our students with some "programming maturity," and we want to engage and excite our students with computer science.

We chose to adapt the Python version of Guzdial's Introduction to Media Computing course, which was developed for use as a non-major's introduction to CS at Georgia Institute of Technology ("Georgia Tech"), as our CS 0.5 course. That course has students writing Python programs to manipulate images (e.g., creating Photo-shop style filters), sounds, animations, and text.

We felt that most of Guzdial's argument's about why that course was good for Georgia Tech's non-CS-majors were also good arguments for using it for CS 0.5. In particular, we think the multimedia approach is a good one because most students enjoy multimedia already and are thus likely to be very interested in manipulating images, animations, and music themselves. We also like the Georgia Tech approach that allows one to first work with the multi-media material itself, then with programming shortly thereafter.

Python is also a good choice of language, because it is not too close to Java, the language used in the rest of the sequence. We want to make sure that CS 0.5 is providing only programming maturity, and not any specific content for the next course, so that it is easy for students who are experienced programmers to skip CS 0.5.

An additional reason to choose the Georgia Tech course is that attracting and retaining more women students was a specific design goal of the Georgia Tech course. We too hope to increase our retention of women.

3. OUR CS 0.5 IMPLEMENTATION

3.1 Before our changes

Prior to Spring 2005, at our school, we used a two-course introductory sequence that was somewhat based on this CS 0.5 idea. The first course, "old CS 0.5" was taken by almost all CS majors. A very few who specifically asked to skip it were allowed to begin with the second course.

The second course both before and after the changes made was a mildly aggressive CS 1 course that has been using the Java programming language for the past several years.

Background: Old form of our CS 0.5. Our previous course had two main sections. Section 1 covered HTML, introducing computation by teaching the effects of using various HTML tags on web pages. This section also included topics on basic operating system commands and file and directory manipulation. Section 2 involved using JavaScript to extend HTML pages into interactive applications. We taught some basic control structures) and covered variables, types, arrays, functions, and parameters. This section occupied at least two-thirds of this course. In it, students created JavaScript programs to explore the various topics.

3.2 Our new introductory sequence

The changes described in this paper were implemented starting in Spring 2005; we now have two and a half years of experience with our new introductory sequence. A key change has been being extremely aggressive about the CS 0.5 approach; in particular we make great efforts to convince all students with programming experience to take our placement test to get into the next course after CS 0.5.[1] The test requires students to write a specified, fairly simple, short program in the programming language of their choice that requires using nested control structures, such as a conditional inside a loop. Furthermore, students are given one course credit towards graduation and placement in the next course if they pass the placement exam. Giving the course credit is important for two reasons:

1. It is strong encouragement for student who can pass the placement test to go on to the next course, and

2. it gives students the message that CS 0.5 is *not* remedial, but rather *normal*.

Notice, when we discuss our good results, that we are selecting out the *least experienced* students to remain in CS 0.5.

Since Spring 2005 we have used the Python version of the Guzdial text, and a modified version of his lecture slides and assignments; we are continuing to make more changes. Our changes to date have included adding a bit more material about computers and how they work, covering the material at a slightly slower pace than Georgia Tech, modestly more "regular" programming assignments, additional weekly tiny programming "finger exercises", and requiring programming to be done by individual students.

The last point is obviously controversial given the great interest in pair and team programming today. However, the majority of our faculty, for better or worse, absolutely insist that students be able to create small programs, or small modifications of large programs, on their own.

4. OUR RESULTS TO DATE

We provide two sorts of results here. Our primary interest is in how well the CS 0.5 approach works to retain students. These results are very positive. A secondary interest is in further studying how well the media computation approach to teaching CS work.

At our school, the CS major is part of the College of Engineering. The beginning of CS major's course sequence is intended above all for the CS majors, but there are several other groups who also properly take it: CS minors, and those engineering majors who choose to take the CS major's computer science courses, which is one option for most engineering majors. The results we give, both "new" and "old" are for students in the College of Engineering, since this group includes the heavy majority of the target audience, and was identified in both our anonymous surveys and grading reports.[2]

Following Guzdial and Tew et al., we define "success" in the course to be obtaining a grade of A, B, or C; "success" is the opposite of the "WFD rate" of students earning D, F, or withdrawing. (The total pool is students enrolled at the end of the brief shopping period not counting a few who received Incomplete's.) Our main result is that in the 2.5 year period before Spring 2005, the success rate was 75.9%. In the four semesters our new courses has run, the success rate has been 84.1%. (The new course was not offered in Spring 2006, another casualty of declining enrollments.) Note that the success rate should have *decreased*, all things being equal, because starting in Spring 2005 we made much greater efforts to remove the best students from the course, placing them in the next course.

Our school's CS 0.5	Enrollment	Success Rate
Fall 2002	61	74.8%
Spring 2003	38	76.7%
Fall 2003	51	68.6%
Spring 2004	22	82.9 %
Fall 2004	15	93.3 %
Average "Old"	37	**75.9%**
New Spring 2005	18	94.4%
New Fall 2005	29	90.0%
New Fall 2006	42	76.2%
New Spring 2007	24	83.3%
Average "New"	28.3	**84.1%**

Table 1: Success rate with CS 0.5 before and with new approach. Averages weighted by enrollment.

This data suggests that the new approach is leading to greater student success, particularly given that one difference between the "new" and the "old" approaches is that under the "new" approach several very well prepared students were placed out of CS 0.5 who in the past would have taken it.

4.1 Student demographics

As can be seen from Table 1, our school had a sharp decrease in beginning CS students in 2004–2005, part of a well known nationwide trend. Indeed, enrollments for 2002–2004 were down considerably from the peak we experienced in 1999–2000.

Table 2: Gender of Survey Participants in Media Computation. Our school data is Spring+Fall 2005 Engr. College students; other is Fall 2003.

Georgia Tech		Gainesville		Our school	
Male	Female	Male	Female	Male	Female
51.1 %	48.9%	37.5%	62.5%	95%	5%

In Tables 2 and 3, we give the gender and ethnicity breakdown for the students in our version Media Computation as

[1]We even make one of the offerings of the placement exam be the optional first-week lab session of CS 0.5.

[2]Prior to the 2004–2005 school year, College of Engineering students were the overwhelming majority of students in the course. In 2004–2005, both the fall "before" course and the spring "new" course had large numbers of students from outside the College of Engineering. A handful were likely CS minors and/or students considering changing to the CS major, but the heavy majority appear to have been students wrongly advised by liberal arts and business advisors during a severe shortage of elective courses for their students that CS 0.5 was the non-majors course. (In fact, there is a distinct non-majors course that had filled.)

Table 3: Ethnicity of Survey Participants in Media Computation. Data is from 2003 at Gainesville and Georgia Tech, and from Spring 2005 at our school.

	Georgia Tech	Gainesville	Our school
African-Am.	6.4%	0	11.1%
Asian	0	7.0%	27.8%
Caucasian	80.8%	96.2%	44.4%
Hispanic	0.3%	0	16.7%
Other	5.4%	3.8%	0

CS 0.5 for CS majors, and for comparison those of the students who took Media Computation for non-CS majors at both Gainesville Community College and Georgia Tech [20].

The courses for non-CS-majors taken together had a modest majority of women, perhaps reflecting the fact that women now make up somewhat over half of all undergraduates nationwide. We were dismayed that the percentage of women in our class was so very low; it seems to reflect the continuing nationwide drop in women entering CS. All the women in our sample were successful students; but there were too few of them to be meaningful. If the number of women *entering* the CS major at all is extremely low, then undergraduate retention cannot be an sufficient tool to increase the number of women in CS—the key areas of effort must be college *recruitment* and the K–12 level.

While overwhelmingly male, our group of CS-majors is much more ethnically diverse than the other two schools' non-CS-majors. The one D earned in our Spring 2005 was by an African-American male, but the low enrollment makes it impossible to draw any statistically meaningful conclusions about success rate versus ethnicity.

4.2 At the end of the course

Students were given voluntary anonymous surveys with the final exam, where they were asked to report on their programming ability, their experiences in the class, and their interests in computer science and multimedia. Similar but distinct surveys were given to the non-CS-majors in 2003 at Georgia Tech and at Gainesville.

Table 4: Programming skills—our school

	Before	After
Good	15%	77%
Weak	23%	23%
None	61%	0

Our students were asked to rate their programming skills at the start and end of the course as "none", "weak", or "good." These results are not directly comparable with the other two schools, because they used a five-point rating scale. To the extent that one can make comparisons at all, our results are perhaps mildly stronger, but broadly similar. At all schools, students reported strong improvement in programming skills. For us, this was vital, since this is one of the key outcomes we feel our CS-majors will need to be successful in the next course.

Increasing the course's relevance to students was a common goal of all three courses. We appear to have been very successful, more so than the other two schools. This is not surprising; we certainly hope CS majors will find this material to be relevant; still our survey results are reassuring.

A question about relevance of the homework was worded differently by us than by the other two schools. We had 92% of the class agreeing or strongly agreeing with the statement "Doing homework helped me succeed in this course." Georgia Tech and Gainesville had only 31–39% agreement with the statement "Homework assignments were relevant to me."

In all three schools, however, students were directly asked about the usefulness of the course both "in other areas of my life" and "later in my professional career." We present results in Table 5 (on next page). Notice that Georgia Tech and Gainesville had very good numbers here, but we had outstanding numbers here. Well over a third of students "strongly agreed" that skills from this course would be useful in both life and their career, and over three quarters "agreed" or "strongly agreed".

5. RELATED WORK

The Georgia Tech Multimedia course's curriculum is detailed in [9]; more information about the approach is given in [7]. An adaptation of that course to a community college (Gainesville) setting is discussed Tew et al. [20], who also compare results between Georgia Tech and Gainesville. In this paper we are able to provide a third data point of comparison for the success of this multimedia approach, and we do so. All three course's use Guzdial's textbook [10].

There have been many recent innovative efforts to address the shortage of women CS majors [5, 11, 13, 15, 16, 22], with the most famous effort probably being that of Carnegie-Mellon University. Carnegie-Mellon University dramatically increased its percentage of women CS undergraduates [13], but the biggest single factor in that success was a change in admissions. Carnegie-Mellon's success probably cannot be replicated at more than a few dozen schools in the U.S. Carnegie-Mellon as a very prestigious school with a very large, very talented applicant pool, has the luxury of turning away many applicants who would be generally successful at Carnegie-Mellon. At most schools, including our school, this luxury does not exist. Every applicant that the school believe would be successful is admitted.

6. CONCLUSIONS

We have introduced a CS 0.5 approach to beginning CS based on Guzdial's media computation course. Results from the first class of students are very strong—a very low WFD rate, and students report that the course is highly relevant. Moreover, we obtained these results with a class that was over 25% African-American and Hispanic.

7. REFERENCES

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

[2] T. Camp. The incredible shrinking pipeline. *Commun. ACM*, 40(10):103–110, 1997.

[3] E. Chabrow. Declining computer-science enrollments should worry anyone interested in the future of the U.S. IT industry. *Information Week*, 2004.

[4] J. M. Cohoon and L.-Y. Chen. Migrating out of computer science. *Computing Research News*, 15(2), 2003. Available on line at URL `http://www.cra.org/CRN/articles/march03/cohoon.chen.html`.

Table 5: Skills from this class will be useful in

	Georgia Tech		Gainesville		Our school	
	Life	Career	Life	Career	Life	Career
Strongly Agree	12.6%	6.8%	6.2%	0	38.5%	46.2%
Agree	47.3%	38.7%	50%	37.5%	38.5%	30.8%
Neutral	23.9%	31.5%	12.5%	25.0%	23%	23%
Disagree	13.1%	16.2%	25%	37.5%	0	0
Strongly Disagree	3.2%	6.8%	6.2%	0	0	0

[5] W. Dann, T. Dragon, S. Cooper, K. Dietzler, K. Ryan, and R. Pausch. Objects: Visualization of behavior and state. In *SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 84–88, 2003.

[6] S. W. Director. 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.

[7] A. Forte and M. Guzdial. Computers for communication, not calculation: Media as a motivation and context for learning. In *Proc. 37th Hawaiian International Conference of Systems Sciences*, 2004.

[8] M. Guzdial. Summary: Retention rates in CS vs. institution. Message posted on moderated ACM SIGCSE-MEMBERS list, Georgia Tech, April 23, 2002. Available on line from URL `http://listserv.acm.org/sigcse-members.html`.

[9] M. Guzdial. A media computation course for non-majors. In *SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 104–108, 2003.

[10] M. Guzdial. *Introduction to Media Computation: A Multimedia Cookbook in Python*. Pearson Prentice Hall, Upper Saddle River, NJ, 2005.

[11] T. Hickey. Incorporating Scheme-based web programming into computer literacy courses. Presented in the Scheme2002 workshop. Available on line at URL `http://www.cs.brandeis.edu/~tim/`, Oct. 2002.

[12] *inroads (The ACM SIGCSE Bulletin)*. Special issue: Women and computing, 2002. Volume 34, No. 2.

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

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

[15] C. McDowell, H. Bullock, J. Fernald, and L. Werner. The effects of pair-programming on performance in an introductory programming course. In *Proc. Thirty-third SIGCSE Technical Symp. on Computer Science Education*, pages 38–42, 2002.

[16] N. Nagappan, L. Williams, M. Ferzil, E. Wiebe, K. Yang, C. Miller, and S. Balik. Improving the CS1 experience with pair programming. In *Proc. Thirty-fourth SIGCSE Technical Symp. on Computer Science Education*, pages 359–362, 2003.

[17] S. L. Peeger, P. Teller, , S. E. Castaneda, M. Wilson, and R. Lindley. Increasing the enrollment of women in computer science. In *Proc. Thirty-second SIGCSE Technical Symp. on Computer Science Education*, pages 386–387, 2001.

[18] E. Roberts, G. Engel, J. H. Cross, R. Shackelford, R. Sloan, R. Austing, D. Carver, C. K. Chang, G. Davies, P. J. Denning, R. Eckhouse, W. King, F. Lau, A. McGettrick, S. Mengel, G. M. Schneider, P. Srimani, and U. Wolz. *Computing Curricula 2001: Computer Science*. IEEE Computer Society Press, 2001. Also available on-line from `http://www.computer.org/education/cc2001/`.

[19] H. Roumani. Design guidelines for the lab component of objects-first CS1. In *Proc. Thirty-third SIGCSE Technical Symp. on Computer Science Education*, pages 222–226, 2002.

[20] A. E. Tew, C. Fowler, and M. Guzdial. Tracking an innovation in introductory CS education from a research university to a two-year college. In *Proc. Thirty-sixth SIGCSE Technical Symp. on Computer Science Education*, pages 416–420, 2005.

[21] J. Vegso. Interest in CS as a major drops among incoming freshmen. *Computing Research News*, 17(3), May 2005. Available on line at URL `http://www.cra.org/CRN/online.html`.

[22] G. W. Zimmerman and D. E. Eber. When worlds collide! an interdisciplinary course in virtual-reality art. In *Proc. Thirty-second SIGCSE Technical Symp. on Computer Science Education*, pages 75–79, 2001.

[23] S. Zweben. 2004–2005 Taulbee survey: Ph.D. production at an all-time high with more new graduates going abroad; undergraduate enrollments again drop significantly. *Computing Research News*, 18(3), May 2006.

[24] S. Zweben. 2005–2006 Taulbee survey record Ph.D. production continues; undergraduate enrollments turning the corner. *Computing Research News*, 19(3), May 2007.