BUILDING A THRIVING CS PROGRAM AT A SMALL LIBERAL ARTS COLLEGE

Timothy Urness and Eric Manley
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
515 271-2118
timothy.urness@drake.edu, eric.manley@drake.edu

ABSTRACT

In this paper we describe several techniques that have helped increase enrollment in the computer science program from 23 computer science majors in 2008 to 42 computer science majors in 2010 – an increase of 82.6%. We discuss issues related to curriculum, programming assignments, and professor-student interactions that have made the discipline more attractive and manageable to a variety of students within the setting of a small liberal arts college.

INTRODUCTION

Liberal arts colleges promote a diverse study of disciplines to emphasize breadth of education, analysis, and integrity in students' intellectual experiences. The goal is to provide a learning environment that prepares students for meaningful personal lives, responsible citizenship, and ultimately, professional accomplishments. At first glance, the liberal arts approach may seem contradictory to an effective method of teaching computer science – a field closely tied to real-world problems and engineering. However, a computer science program can be strongly supported by the liberal arts paradigm [14]. Liberal arts programs in computer science typically emphasize problem-solving, applications of theory, communication, and intellectual skills, not just current technology trends or operational details that may change rapidly. The material content of computer science requires analyzing, synthesizing, and organizing information. These life-long skills are fundamental to a liberal arts education [8]. In this paper, we discuss several of the issues related to curriculum, programming, and professor-student interactions that have made the discipline more attractive and manageable to a variety of students.

CURRICULUM

The curriculum of a liberal arts college emphasizes breadth of study. As a result, students must take a wide variety of courses from outside their chosen area of study. Thus, only approximately 39 hours of classroom time in computer science or mathematics courses can reasonably be required for a major [8]. We acknowledge that each school has a unique environment and several of these suggestions may not be applicable or have the same effect on student enrollments or experiences. However, we have found that these recent changes in our curriculum have encouraged an influx of students to study computer science.

Use Introductory Course to Recruit

Entering students typically have misconceptions about computer science. In a study by Deborah Wiley, a teacher in North Canton Ohio, 66 percent of high school

students had no idea of what the field of computer science is about. Only 7 percent believed that computer science involved programming and networking [5]. While strategies to inform and recruit high school students have shown to be effective [9], we have chosen to focus our efforts on students that have already selected and enrolled in the college. We feel that our limited resources can be best utilized by introducing students to computer science, stressing the potential it has to assist other areas of study and contribute to a liberal arts education. This approach has resulted in a number of students adding computer science as a second major or minor.

Using the approach of attracting current students, it is important to view the introductory course (CS1) as a recruitment tool. Ultimately, the introductory course should be an invitation to computing that highlights applications and usefulness of algorithmic thinking and processing. We have found success in not requiring any programming or computing experience for this course – only some basic mathematics (e.g. college algebra). The goal is for students to take the course and have their views of computer science enlightened by introducing algorithms, programming, basic fundamentals, and applications to other disciplines. In the process, we hope to debunk any previous and erroneous attitudes and beliefs the students may have [2, 7].

Simply providing an excellent introductory course will not necessarily entice students to enroll in the course. In our case, increasing the number of majors was likely a result from adding the introductory computer science courses to the list of courses that satisfy a liberal arts general education requirement. Since computer science is a versatile discipline, the case can be made for the introductory course to be placed in several different general education categories (e.g. information technology, critical thinking, quantitative analysis).

Hooking Nonmajors with a Broad CS0

It has been our experience that CS0 can also be effective in introducing and recruiting students to the discipline of computer science. In contrast to programming-centric introductory courses, introductory courses that provide a broad survey of computer science principles have attracted a much higher percentage of female students and had success in student retention [4, 13].

Simplify Curriculum

Once a student's interest is piqued by an introductory course, it is important to demonstrate that the curriculum is simple and manageable. As suggested in [8], courses with prerequisites should be minimal as to not deter a student that takes the introductory course during his or her sophomore year. Demonstrating that the major (or minor) can still be fulfilled in the senior year without having declared it as a freshman will allow for the inclusion of many more students.

Pay Attention to the Student Experience

Computer science can be an intimidating field of study, full of complicated jargon and ever-changing technology. This is particularly daunting for students in disadvantaged populations or students that have not yet formed an academic identity. As suggested in

[5], an approach to increasing women in computing is to "pay ferocious attention to the quality of the student experience." We believe this applies to increasing enrollment numbers in general – increasing the *quality* of education experiences for all students.

In an effort to make computer science approachable, we eliminate "weed-out" courses that affirm only the brightest or most-committed students. We feel that the professor should be an advocate for each student, allowing grace for students that have a wide range of computing experiences. As professors, we have no influence on the students' background or prior availability of technology. We do, however, have control over how the student perceives computing in the classroom. By paying attention to the student experience, the unfortunate preconceptions of the discipline can be overcome provided every student is given the attention and opportunity to succeed.

Encourage CS Minors

The number of majors in a program is an often-used and potentially misleading metric for the success of a college program. Students with multiple majors, minors, or concentrations add to the health of a program, but may not be counted as a major. A low number of students enrolled in a course, however, will seemingly always get the attention of an administrator with the power to cancel the course. The far-reaching applications of computer science make it a natural complement to other disciplines as a minor. A program filled with computer science minors can lead to a vibrant, healthy, computer science department. Promotion of the computer science minor program can be done in a variety of ways: announcements in CS0 and CS1, department website, emails, etc.

Incorporate Current Technology Trends

A balance that must be found for each computer science professor is how much technology to teach vs. how much theory and concepts to introduce. Mobile application development can be an attractive method for attracting students' attention while still cultivating algorithmic and critical thinking within the discipline. Our experience is that offering an iPhone application development course in the context of software engineering has created a large amount of excitement and enthusiasm within the department.

PROGRAMMING AND PROGRAMMING ASSIGNMENTS

Students will (likely) spend more time working on assignments than in the classroom. The assignments also constitute a large percentage of the final grade. Thus, from the students' perspectives, the assignments are extremely important to the course and their perception of the discipline. To support student interest, it is critical that the assignments are relevant, manageable, not trivial, and highlight the concepts stressed in the classroom. The following "rules of thumb" can help professors develop meaningful, appropriate assignments that cultivate and support interest in the class and discipline.

Make Assignments Meaningful

Nick Parlante of Stanford University has coordinated the publishing of several "nifty" assignments at the annual SIGCSE conference [11]. These assignments were selected from criteria of being fun (nifty), useful for a broad audience (topical), scalable for easy and open-ended assignments, adoptable, inspirational and thought-provoking.

Assignments that focus on highly technical aspects ("geeky") can result in the lasting impression that the real-world problems that computer science can help solve are shallow. Programming assignments that stress concepts and encourage students to utilize logic and problem-solving are likely to attract and retain a diverse collection of students [5]. It has been ours, and others, experiences that the assignments that involve "real world" problems or ask open-ended questions are the most likely to inspire creative, dedicated submissions and can encourage students to pursue the discipline further [6].

Facilitate Student Programming

A controlled lab environment allows a professor to make sure all of the compilers will work the same, the system (probably) won't crash, and everything will work as expected. Requiring students, however, to either purchase specific computer hardware or come into a lab to work on an assignment indicates that the technology they are interacting with is not the same as the ubiquitous computing they interact with on a daily basis. It is highly advantageous that students be able to work on their assignments wherever and whatever their home computer may be. Thus, choosing a development environment and technology that is platform independent, reliable, and flexible will make the discipline more accessible and attractive to students. Furthermore, creating thorough, readable, introductory descriptions can reduce the amount of frustration for a beginner.

Visual Programming Environments

A popular technique for getting students hooked on computer science is to use visual programming environments such as Scratch, Greenfoot, or Alice. These environments make programming accessible and immerse the programmer in a media-rich environment, which is appealing to larger audiences who might otherwise disregard computer science because of preconceived perceptions about programming.

PROFESSOR-STUDENT INTERACTIONS

"... building a relationship of respect between teacher and student for women and minority students is the first order of business – at all levels of school. No tactic of instruction, no matter how ingenious, can succeed without it." [12] As the previous quote illustrates, the professor-student relationship is extremely important. Small class sizes and small professor-to-student ratios are some of the reasons typically stated by students that attend small liberal arts colleges. The professor-student relationship can be foundational for a student deciding to pursue computer science. The following are a few suggestions to help develop and strengthen these important relationships.

Out-Of-Class Activities

We have had great success in opening up the ACM programming competition to any student that would like to participate. Many introductory students have chosen to take part in this all-day computing event which has been an effective recruiting tool. Students end up spending time with majors, making friends, and eventually adding computer science as a major or minor. The incentive for the students is a one-time Friday night practice "party" in which we hold a mock competition, followed by pizza, soda, and a Ms. Pac Man tournament. Since many of the students may be new to computer science, we don't expect them to be competitive. Instead, we encourage their effort and highlight

the experience (e.g. new techniques learned, the fun of the Ms. Pac Man contest, the food, etc.). The result has been a high retention rate for the annual programming contest and an increase in majors and minors.

Communication and Approachability

A way to build a relationship of respect between professor and student is to treat every interaction with the utmost importance. Several ways we have found to be effective in our approachability are to hold extended office hours and respond to email or wiki questions as soon as possible. When students get prompt feedback, they are more likely to retain the information and the answer can be most effective. Similarly, when grading assignments or exams, a timely response is as important as a quality response. In order for students to learn from mistakes, it is most beneficial to point out these mistakes as close as possible to when the mistake was made. Delaying the feedback cycle, even if the feedback is extremely thorough, increases the chances that the student not learn from the mistakes made and the student may become frustrated with the process or professor.

WOMEN IN COMPUTING

According to Computing Research Association, women have earned 11.8% of the CS bachelor degrees in recent years [15]. Recruiting women to computer science is a topic of much research and discussion [1, 3, 10]. Our goal is to create an environment within our program that encourages the participation of students, regardless of gender. Thus, we attempt to teach computer science as a discipline that offers a great deal to interdisciplinary collaborations and practical applications, rather than the technological "geeky" stereotype that can sidetrack women from engaging in the discipline.

By focusing on creating a thriving program, our percentage of women in the program is currently 20%. We feel we have been successful in not deterring, and possibly encouraging, women to consider the major through the following actions: We do not require students to enter the program having any programming experience. It has been our experience that students that are "fresh" to computer science and have strong analytic skills have the potential to be excellent computer scientists. Therefore, whenever possible, and especially in the introductory courses, we stress the "experience is not a prerequisite" message that has been successful in recruiting women [5]. To do so, the curriculum must be simple enough to accommodate the novice student. Additionally, as previously mentioned, our beginning CS1 course starts from the ground floor – there are no prerequisites for our introductory courses. Any student, provided they have a strong work ethic, can take the introductory course and earn an A. Using CS0 and CS1 as recruitment tools has allowed for us to break stereotypes incoming freshman have regarding computer science [2]. Similarly, we have found that using a broad CS0 course can lay a solid foundation for studying computer science and attract women, as supported by reference [1]. In fact, over the last three semesters, women have made up just over 50% (46/91) of the students in CS0. Finally, our computer science program is complementary to several other majors (math, physics, and even biology). Having a high number of double majors reduces the "geek" image of CS majors on campus that can be beneficial to increasing the number of women in computer science [5].

CONCLUSION

Liberal arts colleges promote a diverse study of disciplines to emphasize breadth of education, analysis, and integrity in students' intellectual experiences. Computer science programs in small liberal arts colleges must effectively balance the ideals of a liberal arts education and current technology trends. We have highlighted several strategies involving curriculum, programming assignments, and professor-student interactions that, we feel, have made the discipline more attractive and manageable to a variety of students. The ultimate goal of education is to enrich students' lives. We feel that a computer science program has the potential to present a student with a unique perspective of analytical problem solving, application of theory, and analyzing, synthesizing, and organizing information. Under the best circumstances, this kind of program will thrive at a small liberal arts college.

REFERENCES

[1] Alvarado, C., Dodds, Z., Women in CS: an evaluation of three promising practices, *Proceedings of Symposium on Computer Science Education (SIGCSE '10)*, 57-61, 2010.

[2] Bennett, C., Urness, T., Using daily student presentations to address attitudes and communication skills in CS1, *SIGCSE Bull.,* 41, (1), 76-80. 2009.

[3] Cohoon, J. M.. Must there be so few?: including women in CS. *Proceedings of the 25th international Conference on Software Engineering*, 668-674, 2003.

[4] desJardins, M., Littman, M., Broadening student enthusiasm for computer science with a great insights course, *Proceedings of Symposium on Computer Science Education (SIGCSE '10)*, 157-161, 2010.

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

[6] Layman, L., Williams, L., Slaten, K., Note to self: make assignments meaningful, *SIGCSE Bull.,* 39, (1), 459-463, 2007.

[7] Lewis, C., Attitudes and beliefs about computer science among students and faculty, *SIGCSE Bull.* 39, (2), 37-41, 2007.

[8] Liberal Arts Computer Science Consortium, A 2007 model curriculum for a liberal arts degree in computer science. *J. Educ. Resour. Comput.* 7, (2), 2007.

[9] Morreale, P., Kurkovsky, S., Chang, G., Methodology for successful undergraduate recruiting in computer science at comprehensive public universities, *SIGCSE Bull.,* 41, (1), 91-95, 2009.

[10] Othman, M. and Latih, R., Women in computer science: no shortage here!. *Commun. ACM,* 49, (3), 111-114, 2006.

[11] Parlante, N., Nifty assignments, 2011, http://nifty.stanford.edu/ retrieved January 11, 2011.

[12] Steele, C., Race and the schooling of black Americans, *The Atlantic Monthly*, 4, 68-78, 1992.

[13] Turner, E. H., Albert, E., Turner, R. M., Latour, L., Retaining majors through the introductory sequence, *SIGCSE Bull.,* 39, (1), 24-28, 2007.

[14] Walker, H. M., Kelemen, C., Computer science and the liberal arts: a philosophical examination, *Trans. Comput. Educ.,* 10, (1), 1–10, 2010.

[15] Zweben, S., Upward trend in undergraduate CS enrollment; doctoral production continues at peak levels. *Computing Research News,* 21, (3), 2009