AC 2008-464: IMPROVING ENGINEERING EDUCATION THROUGH CREATIVITY, COLLABORATION, AND CONTEXT IN A FIRST YEAR COURSE

Michael Haungs, California Polytechnic State University

Michael Haungs is an Assistant Professor in the Computer Science Department at California Polytechnic State University. He received his B.S. degree in Industrial Engineering and Operations Research from the University of California, Berkeley, his M.S. degree in Computer Science from Clemson University, and his PhD in Computer Science from the University of California, Davis. His interests are in systems research, with an emphasis on: Distributed Systems, Networking, Interprocess Communications, Operating Systems and Parallel Architectures. Current efforts are in the identification and elimination of I/O bottlenecks in distributed systems.

John Clements, California Polytechnic State University

John Clements is an assistant professor of Computer Science at California Polytechnic State University (Cal Poly) in San Luis Obispo. His research interests include programming languages and pedagogic environments, and he is the author of DrScheme's algebraic stepper, used all over the world to show students the evaluation of programs as a sequence of simple reduction steps. He holds a PhD in Computer Science from Northeastern University.

David Janzen, California Polytechnic State University

David Janzen is an assistant professor of computer science at California Polytechnic State University (Cal Poly) in San Luis Obispo, and president of Simex, a software consulting and training company. Previously he worked on telecommunications fraud detection systems at Sprint, and taught at Bethel College in Kansas. His teaching and research interests focus on software engineering with an emphasis on agile methodologies and practices, empirical software engineering, software architecture, and software metrics. He holds a PhD in computer science from the University of Kansas.

Improving Engineering Education through Creativity, Collaboration, and Context In a First Year Course

Abstract

Over the past few years, Computer Science and some Engineering disciplines have suffered from a decrease in student enrollment, poor retention, and low women and minority representation. We suggest three issues with first-year courses that contribute to this trend. First, students find it difficult to see how their assignments and course material relate to real-world applications. Second, students tend to perceive engineering as an individual endeavor requiring little interaction with peers. Last, early engineering assignments are often overly constrained, possibly to ease grading, allowing minimal room for student creativity.

In this paper, we present a model for an introductory freshman-level course that helps address student enrollment and retention issues. Our course is based on three tenets: (1) the course draws problems from, and teaches about, an interesting and relevant domain in which students already are familiar, (2) the course encourages teamwork and peer communication, (3) the student is actively responsible for their education. To address these, the class teaches game design in a collaborative environment in which students are given open-ended assignments to promote creativity. We address instructor grading concerns, various student skill levels, and individual assessment. In our approach, we encourage the implicit acquisition of basic computer science concepts and skills as opposed to directly lecturing about them. Over 60% of the students in our class had no prior programming experience, yet all of the student teams were successful in developing engaging Flash-based games. Student surveys revealed that nearly all students characterize computer science as collaborative, multi-disciplinary, and creative. We believe our class can serve as a model to create other discipline-specific introductory courses.

1.0 Introduction

Project-Based Learning (PBL) has been shown to improve student retention, increase long-term interest, and improve performance in future design courses.[2, 6, 9, 12] Capstone and cornerstone[3] courses are a common home for PBL in many universities. Capstone courses are well-known and cornerstone courses are their freshman-level equivalent. Studies show that cornerstone courses have an even greater impact on the retention of women and minorities[6, 17] – in some cases, retention rates improved up to 27% and 56%, respectively.[2] The growth of capstone and cornerstone courses may be the result of a 1997 National Science Foundation report[5] that calls for engineering educational reform and ABET criteria that emphasize design, teamwork, and communication.

Unfortunately, one of the major roadblocks to cornerstone courses is faculty involvement. Faculty members are generally resistant to the amount of effort required to organize and maintain a successful PBL course.[11] Cornerstone courses involve creating an interesting problem, possibly in an unfamiliar field, working and managing student teams, and assessing individual and team contributions while accommodating the skill level of incoming freshmen. This extra work does not sound particularly appealing given the pressure on faculty to publish and the lack of staff to cover upper-division courses. This is not a phenomenon isolated to cornerstone courses. In his most recent book,[1] former Harvard president Derek Bok takes a critical look at American undergraduate education and suggests that these key factors contribute to the reduction of faculty involvement in the first years of a student's university experience.

Despite these issues, many faculty are taking up the challenge to create meaningful first-year experiences. In this paper, we propose a model for a cornerstone course that can ease its adoption. The key to our model is teaching the students life-long learning.[10] In addition, we include assessment activities and lectures to help facilitate the transition from high school to college. Providing support to adjusting to college has been shown to be beneficial to a student's retention and peformance.[7] We present our cornerstone pilot course, *Introduction to Interactive Entertainment*, discuss how it conforms to our model, and give some preliminary results on its efficacy.

2.0 Class Model

We have designed a model for a cornerstone class suitable for many different engineering disciplines. Our model is based on three tenets that guide our selection of course goals and requirements. In this section, we present those tenets and why we chose them, course goals, and the requirements we believe are necessary to ensure courses derived from this model enjoy the benefits of PBL and life-long learning. In the next section, we'll describe our experience with a cornerstone course we derived from this model.

2.1 Tenets

One reason proposed for the low enrollment of women and minorities in Computer Science is that they don't see the social relevance of the field.[15] It is important that students are exposed early to significant problems in the field to gain an appreciation of it. Sturm and Moroh write, "By emphasizing those areas of Computer Science and Computer Engineering that most require [social relevance and teamwork], it may be possible to attract more women (and minorities) to computing professions.[15]" Teamwork has also been shown to contribute to general student success.[4] Every cornerstone course should involve teams of students working on socially relevant issues.

Project-based learning is a method to expose students to constructivist learning. Instead of the objectivist view (traditional) that learning is the process of fact accumulation, constructivists believe that "learning is determined by the complex interplay among learners' existing knowledge, the social context, and the problem to be solved.[16]" This type of learning is a major contributor to the success of PBL courses and should be present in cornerstone courses as well.

It is also important that students gain confidence early in their college careers. They should work in a domain that is familiar and, in which, they can immediately contribute. This is important for two reasons. First, some suggest that self-esteem issues could be one of the reasons for the low representation and retention rates of women and minorities.[15] In general, improving the first experiences of college life for freshman significantly increases retention rates.[7] Second, it solves the pedagogical issue of teaching students with wide and varied backgrounds. For example, first-year Computer Science courses include students that took programming courses in high school, giving them a clear advantage over those that did not. By choosing a domain that most students feel comfortable discussing, teachers reduce this disparity.

Given the above, our three tenets are:

1. The course draws problems from, and teaches about, an interesting and relevant domain with which students already are familiar,
2. the course encourages teamwork and peer communication, and
3. the student is actively responsible for his or her education.

2.2 Course Goals

Course goals provide the general outcomes of the course. They guide the choice of course lecture topics and specific learning activities. Our course goals are that the students:

1. practice and understand the importance of good design,
2. develop problem solving skills,
3. develop communication and teamwork skills,
4. understand iterative design, implementation, and testing, and
5. receive an introduction to an engineering discipline (big picture).

Learning objectives speak to specific activities the student should be able to perform after completing the course. We do not discuss learning objectives in this paper as they are often tied to the specific topic chosen. For example, a cornerstone course teaching game development might have a learning objective related to "playtesting" which is a critical step in the incremental implementation process of game development. This activity is specific to the given domain and contributes to the fourth course goal, "understand iterative design, implementation, and testing."

2.3 Requirements

We have developed a set of course requirements to ensure that courses created from our model are consistent, stay true to our tenets, and conform to our course goals. Our requirements are:

1. Students work on a project in a relevant, interesting, and accessible domain,
2. teaching design is the major focus of lecture,
3. students work in teams on the course project,
4. students write a final report on their project that describes their project's design choices, implementation, and assessment,
5. the course has a midterm and a final,
6. students present their work a minimum of two times to the class,
7. a knowledge of programming must not give a student an overwhelming advantage in the course, and

8. students are actively responsible for their learning (constructivist learning).

Some of the requirements directly relate to our course goals and tenets, but some of our requirements warrant further discussion. Students are going to be required to learn the necessary technical skills to build a solution to the supplied problem in a lab environment, through independent study, and via course collaboration tools (such as a message forum or a wiki). A minimum amount of time should be spent in lecture discussing these technologies. For example, if students are using *Lego Mindstorms* to construct their final deliverable, they themselves should be responsible for learning that environment. Thus, we have the second requirement.

The fourth requirement addresses two issues. First, since the class is mainly about problem solving and design, it is crucial to have some form of assessment that demonstrates their problem solving process and application of the design principles taught in lecture. This type of assessment is difficult to do via final examinations or project demonstrations. Second, communication skills are a well-known trouble area for engineering students and, whenever possible, engineering curriculum should give them opportunities to exercise those skills.

As stated, it is important to have students work in teams, yet some form of individual assessment is necessary as well. The fifth requirement handles this need. However, there is another goal of this requirement. The course also serves as an introduction to university life and challenges. Thus, the fifth requirement also exposes the student to the type of examinations they will encounter through the rest of their collegiate careers.

2.4 Faculty Support

Typically in introductory classes, students are required to do many rote activities to help the student become familiar with basic skills. These type of activities require multiple weekly assignments to be graded for each student. The main activities of our model, presentations and the final report, are team activities which reduce the sheer number of items to be graded. For example, at CalPoly, San Luis Obispo, our introductory computer science course has approximately 20 items to grade per student. In our cornerstone course, see Section 3.0, we have 9 individual items to grade per student and `(4 * TotalStudentCount / TeamSize)` group assignments to grade. For our class size and team sizes, we roughly have the equivalent of 11 individual assignments to grade – a savings of 55%.

Due to the collaborative nature of the course and active learning, laboratory hours are used as times when students discuss technical difficulties with each other and work to solve those issues instead of following detailed instructor exercises. Since the course concentrates on design, students are free to use various resources (Internet, library, domain experts) to help them overcome difficulties. Stated another way, *it is the student's responsibility to learn the technical aspects of the given domain.* During lab, the instructor serves as just another available resource to the student.

2.5 Disclaimer

Keep in mind that students resist change.[18] Some students may have already taken college courses or have family members that have described typical college course experiences. At the very least, practicing life-long learning is something they almost assuredly did not encounter in high school. To cope with this, an instructor should:

- clearly state the objectives and goals of the course on the syllabus,
- clearly state the responsibilities of both the instructor and the student on the syllabus, and
- discuss the benefits of teamwork and active learning in class.

We would like to stress one last point. A cornerstone course should not be treated like a "weeder" course. The goal of this course is to encourage those that are on the fence, not to scare them away.

3.0 CSC 171: Introduction to Interactive Entertainment

CSC 171, Introduction to Interactive Entertainment, is our pilot cornerstone course. We chose game development as the domain in which to teach design for a number of reasons. First, the gaming industry is experiencing huge growth, increasing the demand for college graduates with knowledge in this domain. Second, it meets our first cornerstone course requirement in that it is relevant, interesting, and accessible. It was clear from the first day that every student in the class had ideas to express and thoughts to share on game design. Third, we are starting a new game development program at Cal Poly, San Luis Obispo. In this section, we will discuss the key components to CSC 171: lectures, development environment, labs, examinations, presentations, and the final project.

3.1 Lectures

The majority of lectures taught game design, drawing heavily from Salen and Zimmerman.[13, 14] We also discussed a game development and implementation process that is very similar to general software development methods taught in computer science. The lectures had numerous class exercises to bring active learning into the classroom. We dedicated three lecture hours to presenting a "big picture" view of computer science to help prepare students for future computer science courses. We also took the opportunity to have industry professionals come and talk to our students (Eric Plante, Electronic Arts and MK Haley, Disney). This turned out to be an invaluable experience.

3.2 Development Environment

Picking the right development environment really determines if the students can have a successful quarter. After much research and consultation with industry advisors, we chose Adobe Flash. Flash has a development environment suitable for both novice and expert users. Flash animations can be created either from an intuitive graphical user interface or through a scripting language, ActionScript. The students were able to immediately be productive in the environment and manage their own migration to using more complicated ActionScript programming.

Another reason we chose Flash is the multitude of online resources available. Many tutorials exist for almost every aspect of game design, from collision detection to character animation. If needed, there are also plenty of Flash books available to augment student learning. It is amazing how much a student can learn when the motivation is "How do I tell if two objects in Flash collide?" verses "How do I complete Lab #2?".

Finally, with Flash projects students can easily share their work with other students and family members over the web.

3.3 Labs

Labs served as collaboration hubs. The class did require the completion of a few simple exercises in lab so that students could demonstrate familiarity with the tool. However, for the most part, lab was a place to discuss ideas, experiment, and collectively learn Flash.

3.4 Examinations

As specified by the course model, CSC 171 had examinations. Again, the examinations served to provide individual assessment as well as help give the students test-taking experience.

3.5 Presentations

The students were required to do two presentations. The first presentation modeled a typical "game pitch" that a game developer would have to do to raise funding, get industry interest, or acquire corporate approval to start working on the project. The presentation included a rough prototype (one group acted out their game idea), concept drawings, and a justification why this game would be fun to play. The second presentation was the discussion of their final product in which they had to give a description of their games and its rules, discuss their development process, including playtesting results, and defend their design choices. They had to conclude their presentation with a full demonstration of their final game.

3.6 Final Project

The students had to complete four milestones associated with their final project. The first and last milestones were the presentations described above. The other two milestones conformed to the standard game development practice of creating a working prototype containing 30% and 60% of the features of the game. Their final paper consisted of 10 pages (minimally) giving a detailed description of their game, design, development process, and playtesting assessment.

3.7 Conclusion

It is our impression that the course was an empowering experience for the students and really gave them a sense of the things they could accomplish with a computer science degree. A number of the students remarked that they were looking forward to their introductory courses because they "saw the need."

4.0 Assessment

It's too early to comprehensively assess the effect of this class on student learning and retention, but based on student surveys and final grades, our preliminary results are encouraging. We also present the final projects that were submitted and use them as a means of assessing whether or not we (and the students) were able to meet the course goals.

4.1 Survey

To measure student attitudes during and after the class, we conducted two student surveys; one in the second week of class, and one on the final exam. These surveys aimed to measure the students' perception of their own success, their interest in the major and the resulting career, and their enthusiasm for the subject.

Question	Before %	After %
“Are you interested in a career in Computer Science?”	77	81
“Are you likely to continue as a CS Major?”	77	77
“Rate your ability to work with others.”	97	100
“Rate your ability to develop a program.”	39	45
“Rate your ability to critically think about a program.”	74	71

Figure 1: Responses of 4 or 5 to selected “1–5” questions

Question	Before %	After %
“Have you taken any other Computer Science classes?”	39	42
“Were any CS courses offered at your high school?”	45	35
“If so, did you take one?”	35	42
“Is Computer Science collaborative?”	97	100
“Is Computer Science multidisciplinary?”	100	97
“Is Computer Science creative?”	90	97
“Is Computer Science intimidating?”	55	80

Figure 2: Responses of ‘yes’ to selected yes/no questions

Figures 1 and 2 summarize the responses of the students to selected questions before and after the class. We omit from this table questions related to gender, age, race, and class year. Table 1 shows the fraction of students (expressed as an integral percentage) that chose either 4 or 5 in response to the given “1 through 5” question, and table 2 shows the fraction (expressed as an integral percentage) that answered ‘yes’ to a given question.

The results suggest that the students are broadly enthusiastic about Computer Science as a career option. Of the 31 students that took the final exam, 81% answered either 4 or 5 (on a scale of 1–5) in answer to the question, “What is your opinion of Computer Science as a career?” Additionally, 77% answered either 4 or 5 in response to the question “Do you plan to continue as a computer science major?” All thirty-one respondents indicated that they liked working with others in the course’s projects.

On one hand, there’s no doubt that computer science remains intimidating. At the beginning of the class, 55% believed computer science to be intimidating. At the end of the class, 80% believed computer science to be intimidating.

On the other hand, there was nearly universal agreement among the respondents that computer science was collaborative, multi-disciplinary, and creative. On each of these three questions, more than 96% of the students indicated that they found computer science to possess the given trait. This is especially interesting since “several studies have shown that more female than male students worry that a computing degree will not allow them to work with people.[8]”

Also, we were pleased to note that there was no statistically significant difference between the final grades earned by the 5 females and 26 males in the class, as measured by a two-sample t-test with $p < 0.05$. In fact, there was no significant difference between the female and male populations in their responses to any of the evaluative questions on the final survey. Naturally, the small sample size of the females in the class make it difficult to draw many conclusions, and we look forward to gathering more data as time goes by.

4.2 Final Projects

The most important assessment for our course (and model) are the final project deliverables. If the games where unplayable, broken, or not interesting then we were not successful in meeting our stated course goals (see Section 2.2). We are glad to report that the games submitted were excellent, especially considering that over 60% of this freshman class had no prior programming instruction. Of course, our proclamation that the projects were excellent is subjective. Please judge for yourself:

`http://www.csc.calpoly.edu/˜mhaungs/171_projects.html`

5.0 Future Work

This paper proposes a model for an introductory course that can be applied to a wide variety of course topics. Our most significant goal is therefore to demonstrate this by leading such courses ourselves. One of us has proposed a robotics course, and another has proposed a course on music. Each of these provides a broad spectrum of possible projects, and we look forward to seeing how these will play out.

Another important task is to continue the evaluation of the students that took the class in this quarter. In particular, we hope to answer several important questions with surveys and grades from the following two quarters:

1. Are these students more likely to continue as CS majors, and
2. How does their performance compare to others who did not take the class?

Conclusion

We suggest that collaboration and life-long learning are key ingredients to a successful freshman experience and can help improve retention rates, performance, and make the major more appealing to a wider, diverse set of students. Without a doubt, teaching cornerstone courses requires a substantial time investment. We believe that the constructivist course design model we present here can substantially reduce the effort required to effectively conduct such classes.

Bibliography

[1] Derek Bok. *Our Underachieving Colleges: A candid look at how much students learn and why they should be learning more*. Princeton University Press, 2006.

[2] L.E. Carlson D.W. Knight and J.F. Sullivan. Staying in engineering: Impact of a hands-on, team-based, first-year projects course on student retention. In *Proceedings of the ASEE conference and exhibition*, page Session 3553. ASEE, 2003.

[3] C.L. Dym. Learning engineering: Design, languages, and experiences. *Journal of Engineering Education*, 88(2):145–148, 1999.

[4] F.K. Fink. Integration of engineering practice into curriculum – 25 years of experience with problem-based learning. In *Proceedings of the 1999 Frontiers in Education Conference*, 1999.

[5] National Science Foundation. Systemic engineering education reform: An action agenda. *NSF98-27*, 1997.

[6] M. Hoit and M. Ohland. The impact of a discipline-based introduction to engineering course on improving retention. *Journal of Engineering Education*, 87(1):79–85, 1998.

[7] L. Lund and D. Budny. Working with students and parents to improve the freshman retention. In *31st ASEE/IEEE Frontiers in Education Conference*, page Session T3E. ASEE, 2001.

[8] David Nagel. Women lose ground in it, computer science. http://campustechnology.com/articles/52710/, 2007.

[9] B.M. Olds and R.L. Miller. The effect of a first-year integrated engineering curriculum on graduation rates and student satisfaction: A longitudinal study. *Journal of Engineering Education*, 93(1):23–35, 2004.

[10] Mike Osborne. *The Pedagogy of Lifelong Learning*. Routledge, 2007.

[11] L.G. Richards and S. Carlson-Skalak. Faculty reactions to teaching engineering design to first-year students. *Journal of Engineering Education*, 86(3):79–85, 1997.

[12] J. Richardson and J. Dantzler. Effect of a freshman engineering program on retention and academic performance. In *Proceedings of the 2002 Frontiers in Education Conference*. Institute of Electrical and Electronic Engineers, 2002.

[13] Salen and Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2004.

[14] Salen and Zimmerman. *half-real: Video Games between Real Rules and Fictional Worlds*. The MIT Press, 2005.

[15] D. Sturm and M. Moroh. Encouraging enrollment and retention of women in computer science classes. In *Proceedings of the Annual National Educational Computing Conference*, 1994.

[16] Maureen Tam. Constructivism, instructional design, and technology: Implications for transforming distance learning. *Educational Technology and Society*, 3(2), 2000.

[17] T. Monogue V. Wilson and C. Malave. First year comparative evaluation of the texas a&m freshman integrated engineering program. In *Proceedings of the 1995 Frontiers in Education Conference*. Institute of Electrical and Electronic Engineers, 1995.

[18] M. Weimer. Why students resist innovative teaching methods. *The Teaching Professor*, page 2, December 2003.