AC 2012-3215: TEAMS, DESIGN, MENTORING, AND MANAGING FOR COMPUTER SCIENCE UNDERCLASSMEN

Dr. David Wilczynski, University of Southern California

David Wilczynski has a long history at USC. He was the first Ph.D. graduate from USC Information Science Institute in 1975, where some of the initial work on Arpanet was done. His research specialty at the time was in Knowledge Representation. In 1984, he left USC for almost 20 years to be an entrepreneur. Most of his work was in manufacturing, both in Detroit and Japan. During that time, he worked on programming real-time systems using an Agent methodology, which he now teach in his CSCI 201 class. He returned to USC in 2002 to teach full time. Mostly, he worries about how to make undergraduate engineering students more professional. Once a tennis player, he is now trying to become a golfer. Bridge, cooking, and his family take the rest of his time.

Dr. Gisele Ragusa, University of Southern California
Dr. Michael Crowley, University of Southern California

©American Society for Engineering Education, 2012
Teams, Design, Mentoring, and Managing for Computer Science Underclassmen

abstract

Given that 85% of computer science undergraduates immediately take jobs after graduation, industry's complaint about their readiness for "real" work must be taken seriously. Several skills are missing, among which are working on a team, professionalism in coding and documentation, and understanding large systems. In addition, working on teams raises issues of being managed and managing. We have developed a novel three-class sequence—C3, C4, then PM—for sophomores, juniors, and seniors that directly addresses these issues.

C3, the third class in our university’s computer science programming sequence, is populated with sophomores and is about object-oriented programming, graphics, and user-interfaces. C4, the last class in the programming sequence, also populated mostly with sophomores, is about advanced programming concepts, including concurrency, design methods, unit testing, and agent programming. We teach pedagogic material for the first half of the semester; then in the second half, students form inter-class teams for their main project. The PM class trains students in project management and other soft skills and then puts them on the line managing the C3/C4 teams. Many of the PM students have already taken the C3/C4 sequence, but we have graduate students who have not. The projects are sophisticated, requiring significant design, scheduling, and teamwork over an eight week period. In addition, because students in C4 have already taken C3, they are in a position to mentor the younger students even though the projects are different from semester to semester.

We have received validation that this approach is working. In class surveys, more than 90% of the students enjoyed it. We have also heard from our industrial review board and company recruiters that this plan is to their liking. Students from this program are receiving top internships and find themselves leading their internship teams. We also get letters from former students about how far ahead of their peers they are in their first jobs.

introduction

Our Computer Science Industry Advisory Board regularly reminded us at our bi-annual meetings that university B.S. graduates are not prepared for the real world; they don't know how to work on teams, they don't know what it means to work on large systems, and they don't know how to write professional code. They said it takes the companies one year to adequately train a new hire for productive work. We heard the same complaint at various workshops and the conclusions of the Engineer of 2020 proposal\(^1\) reinforces this complaint about engineering education in general.

The two course instructors, Wilczynski and Crowley, are well familiar with industrial needs. Both of them went from academic research to industry and entrepreneurial careers before returning to the university to teach. Our students are talented, but the evolving workplace often outpaces our curriculum. We needed to find a way to maintain contact with the needs of industry; otherwise, we are seen by industry as mostly irrelevant. In particular:
1. Working On Teams – No single issue is more important, nor more elusive for an undergraduate program than training undergraduate students to work on teams. At the graduate level, ongoing research projects managed by research professors offer students the team experience and all that it entails. Most engineering programs have senior design classes revolving around a semester-long team project. This is useful, but often, too little and too late. We need projects that encourage cooperation and facilitate mentoring early in an undergraduate’s career. The teaming of two normally independent classes is our key innovation.

2. Large Systems – Our semester orientation precludes large projects. Large projects mean design, documentation, project planning, project management, coding, testing, integration, deployment and maintenance—in other words, the full spectrum of software engineering skills. By spanning two semesters, the students in C3 (csci200) and C4 (csci201) see both sides of a large system—the user interface and animation in C3 and the backend application code in C4—while being introduced to the software engineering process. By coordinating these two classes, we also have a natural hierarchy that puts older, more experienced students in a position to mentor the younger students.

3. Professional Code and Documentation – There is more to programming than coding. Real software teams use a repository to store their source code and communication tools to manage their documentation. Our teams use Subversion for source control and a wiki for their design, meeting reports, and so forth.

These three components provide our students more “real life” experiences and require students to be “actively” engaged in their learning processes. This curricular and pedagogical change is also succinctly aligned with strong learning and instructional theory. Accordingly, to meet industry needs, we overcame some of the structural impediments of university-style classes to solve these problems.

review of the literature

Engaging students in course experiences that mimic real life experiences and actively engaging them in their learning is supported by educational research. Active learning techniques engage students in their learning process by allowing them to practice skills that they will learning in professional practices, as well as provide them with just-in-time feedback through peers and instructors that helps students assess the degree of their attainment of learning. Instructors no longer convey knowledge in a predominately one-way conversation, but become facilitators of educational experiences designed to help students master specific concepts or skills specific to their future profession. This kind of approach has yielded increased achievement of student learning outcomes and is the focus of much educational research. Our approach provides students with a combined project-based and problem-based learning experience. This pedagogical approach has emerged as an interconnected, contemporary practice in engineering education. Illustrative of this practice, Martin and Devinish’s engineering education research on project-based pedagogical approaches to curriculum delivery revealed that this particular approach is learner-centric and allows for a focused inquiry that students engage with in teams.
with problem foci. This research indicates that project-based learning provides students with opportunities to engage in joint problem-solving with real life engineering relevance\(^{21,17,13,20}\), a research-based practice that is supported by the recommendations of both the National Academy of Engineering\(^{22}\) and the Center for the Advancement of Engineering Education\(^{18}\), and is now required within accredited undergraduate programs by ABET.

design and instrumentation of the program

The work to be described takes place in USC’s computer science department within the Viterbi School of Engineering. Within computer science we offer four majors, CS, CECS (Computer Engineering/Computer Science), CS/Games, and CS/Business. The CS and CECS programs are ABET accredited. We have about 400 undergraduates, divided roughly equally among the four majors. Differentiation among the majors takes place in the junior year. All computer science students take the required C3/C4 sequence. In any given semester, attendance in C3 or C4 ranges from 40-70 students. Currently, enrollment is up due to the perception that computer science offers excellent employment opportunities.

In our initial conception, we simply wanted to improve the experience and skills set of all our undergraduate computer science students. The program described in this paper evolved in three unplanned stages. The University’s Fund for Innovative Teaching (FIUT) supported each stage with independent grants.

In 2004, an FIUT grant\(^4\) supported the redesign of C4 from a hodge-podge of advanced programming concepts to a class focusing on large systems. The intent was to change the student’s perspective from algorithm design and coding to a wider system view involving the software engineering process. Students are taught this new view by immersing them into an existing body of multi-threaded code for a restaurant simulation driven by concurrent agents. They are asked to enhance the project by adding new agents and scenarios. Along the way, they learn about threading, concurrency, shared data, agents, messaging, operating systems, unit testing, etc., all taught in the context of the restaurant application. We feel that working within an existing code base is the way they will encounter code in their first industry jobs.

In the restaurant simulation, there is an animation and user-interface, but the class focuses on the backend application code. There is simply no time (or motivation) to make the front-end more sophisticated; that is not the point of the class.

A few years later, we noticed that the C4 class was developing sophisticated backend code with a supplied front-end, while C3 was developing sophisticated interfaces driven by simple backend code. We wondered about combining the two classes to develop a system with sophisticated interfaces driven by sophisticated backend code. In 2007, a second FIUT grant\(^5\) supported reorganizing the classes to team C3 and C4 at mid-semester to do just that.

For the first half of the semester, C3 is about the design and implementation of graphical user interfaces (GUIs). About one-half of the lecture time is spent learning theories and practices on how to design user interfaces; the other half is spent learning how to use the Java programming language to implement GUIs. Students learn about usability, prototyping, data presentation, and
methods for systematically designing user interfaces. In the process, students learn the following Java mechanisms: Java Collections Framework; inheritance, polymorphism and generics; event-driven programming and interfaces; GUI components and layouts. Their main assignment during that first half of the semester is to build a simple game. During that time, C4 works on the restaurant simulation.

Then, teams are formed to design and build a new system starting only from a specification. Team size can vary from 8-15 students. The new project’s scope is much larger than students normally face. The individual teams meet together to do the design and implementation planning. The integration of C3’s user-interface work with C4’s back-end code is the primary challenge. And, of course, since the C4 students have already taken C3, there will be natural mentoring about user-interface and animation issues.

The projects have evolved, but are now based on work that Wilczynski did during his entrepreneurial career when his company built manufacturing applications in an area called cell control. In the fall we do an assembly cell. In spring we do a glass-processing line. Schematics of the cells, which the students will build and animate, are shown in figures 1 and 2 in the appendix. Here are links to the specifications the students start from:

http://www-scf.usc.edu/~csci201/factory/TheKittingCell.htm
http://www-scf.usc.edu/~csci201/factory/TheGlassLine.pdf

During the eight weeks, the teams have six deliverables:

1. **End of 1st week:** A design for a v.0 skeleton. The skeleton is just bare bones code exploring parts of the system. The point of v.0 is to get the teams off to a quick start and have some code working without worrying about a fully integrated system.

2. **End of 3rd week:** The v.0 skeleton. This code is stored in the Subversion repository as are all code deliveries. For the C3 part of the team this is their first exposure to the repository. The C4 team members used Subversion the semester before and are in charge of training their C3 teammates.

3. **End of 4th week:** A final design that includes both v1 and v2. This includes interaction diagrams, agent/backend design (for C4), frontend design (for C3), and the API between the front and backend.

4. **End of 5th week:** v.1: A working version of the normative scenario for the cell.

5. **End of 7th week:** v.2: A second and full delivery of the cell with all the non-normative cases handled.

6. **Week 8 - the last week of class:** Each team makes a presentation. They demonstrate their system, discuss what their team did well, what they did poorly, and their lessons learned. The presentations are given in a celebratory atmosphere. The students are enthusiastic about showing off their work. We know many of the teams are surprised that they “did
it.” Recently, one of the teams produced a spectacular kitting cell. A student made a 20 minute video posted on youtube.com—http://www.youtube.com/watch?v=Cy6PmjEtpe8.

The C3/C4 experiment had been successful in virtually every way except for designation of a project leader. For the first two years, we deliberately skirted the issue and let project leaders emerge from the students on the team. Unfortunately, the result has often been that the best member of the team becomes the defacto head of the project and does the bulk of the planning as well as the programming. To avoid this result, we hired former C4 students, now typically juniors or seniors, to head the projects. In fall 2009, an FIUT grant supported paying student-managers for the C3/C4 teams.

We formed 8 teams and hired 8 student-managers, each to work for 10 hours per week for the 8 week project. We instituted this practice as a means of developing a peer mentoring approach within the courses where managers (in other words more experienced students) supported “rookie” student in the projects. We were encouraged by the enthusiasm of the new managers. In presenting the manager idea at an industry meeting, a mid-level manager from Microsoft thought the idea so special, that he volunteered to mentor our managers for one hour per week for the entire eight week period. He ran a video-conference every week to discuss progress, problems, the next deliverable, etc. He has each team manager produce a weekly report using a template he designed. This visibility to Microsoft is a bonus that we could never have envisioned. We know the managers worked diligently to satisfy his requirements while at the same time giving direction to the teams and visibility to us into the teams. The whole process has a chaos about it that is eminently satisfying since it looks just like a company trying to carry out several independent projects simultaneously.

The FIUT manager funding was for only one year. For a steady supply of managers, we created a new class, PM (csci401), to teach software project management. For the first half of the class, they study the software development lifecycle and project management from a book. Some of the chapters are: Understanding the Project; Building Epic Experiences; Designing Software; Communication and Relationships; Power and Politics; Requirements Gathering; What to Do with Ideas; Leadership; When Things Go Wrong. Many of the so-called soft-skills are clearly represented. In addition, the project managers also do training exercises for giving presentations, resolving conflicts, honing communication skills, and so forth.

In a recent Atlantic Monthly article, the author said, "Today's [university] students need a stronger focus on teamwork, persuasion, and entrepreneurship; a better integration of liberal arts with technological literacy; and an emphasis on the social intelligence that makes for creative collaboration and leadership." That social intelligence includes "...the capacity to bring the right people together on a project, the ability to help develop other people, and a keen sense of empathy." All of these are goals of our class.

Our student-managers are given power. They cannot hire, fire, or grade, but they have day to day control of managing the progress of their teams. If they find a student missing meetings or ignoring work responsibilities, the manager can give them a Corrective Action Plan (CAP)—see Figure 3 in the appendix—which documents the deficiencies. The CAP must be mitigated or else the professors get involved. Getting a CAP is punitive, but the purpose of the project is to teach
responsibility to a team. For students traditionally trained by universities to work independently, team loyalty is not always natural.

analyses

There were several outcomes we expected at the beginning of this experiment:

- Students will be able to work effectively on a team, meeting deadlines, fulfilling the responsibilities of working on "larger" projects.
- Students will have more pride in their work because they know it will part of a larger system and their contribution will be seen by their peers.
- Experienced students will learn how to "mentor" younger students.
- Managers will learn what it means to lead a team by a real experience.
- Students will get the best job and intern positions.

Assessment for the experiment is straightforward:

- Are we projects completed as specified and on time? Professors will judge the quality of the work. Managers played a key role in this assessment. Currently, the projects are graded via the deliverables described above.
- Are students pleased with the results? Engineering students are regularly polled about their classroom experience. They were happy to participate, many writing long essays about their experience in response to our informal survey. In the survey the main question we asked was: “Did you find the experience useful, fulfilling, thrilling, or what?” Here is a typical response: “It was useful and fulfilling. It was exactly what it was supposed to be, a useful experience for learning what it’s like to work in a team. Completing the project, seeing it work, and seeing everyone’s part come together, was actually quite satisfying, and I very much enjoyed the moment where I realized I was proud of what we’d finished.” However, not all the comments were positive. Some students felt they were overworked; some mentioned they didn’t like the peer pressure; one thought the C3/C4 linkage was not consistent with the “university stand-alone class” organization.
- Was the management experience worthwhile? Managers write reports on their experience. They detail meeting behavior, how effective their teams were in meeting deadlines, as well as documenting problems and how problem students were handled. They also give a presentation at the end of the semester. As professors, we watched them grow. We are surprised at how committed the managers are to their teams and how sophisticated they become by the end of class.
- Are students getting the best intern positions? Currently, Microsoft, Google, and Intel make special presentations to the C4 class recruiting them for summer internship positions. Obviously, these are plum positions. These same recruiters have told us how effective our students are on their intern teams; we have heard this from four or five different recruiters. Here is a letter (slightly edited for length) from one student describing his internship hiring experience: “During the spring of my sophomore year I was at an interview for a bank in Pittsburgh, Pennsylvania. I was there with approximately 20 other candidates. During the down time between our interviews, I was working on the Factory Project. Around 5 minutes into my work a crowd gathered watching. Having seen the fully working factory project they asked about it. During my demonstration the group had grown to around 8 people. When I finished my demonstration one of them asked ‘So you’re a senior interviewing for full time?’
When I told him that I was in fact a sophomore interviewing for an internship, the group began to murmur to each other. One of them asked with an air of slight disbelief, ‘What school do you go to?’” We have many letters like this.

- Are students getting the best jobs? One student in the C4 class emailed: “I wrote about the project on my resume. It seems to be attracting a lot of attention from all sorts of IT/CS related companies.” In talking to him, he said that one interviewer spent fully half the time asking about this project. That is exactly the effect we were looking for. Wilczynski, who teaches the senior design class, sees these students upon graduation. He gathers data by asking students where they are going to work and we know the students are getting top positions at the best companies.

- Are the students better prepared for their industrial careers? We intend to keep our advisory board informed about what we are doing and their feedback will tell us if we are succeeding. We often have students specifically mention that certain skills they have learned in class were important in their jobs. We expect more feedback from these students about how their team skills made them better teammates as employees and researchers.

All of the above assessment is anecdotal or from surveys. We do some formal assessment. All required computer science classes, including C3 and C4, use a methodology we designed for our ABET accreditation. The professors and their graders review all the work and the graders follow our ABET assessment procedure. That ABET assessment only tells us how the class outcomes align with ABET program outcomes.

Expanding on that assessment, we plan to utilize a multi-dimensional rubric with articulated scoring criteria that expands upon the bulleted points described above and assigns both a numeric score to each project component and adds a narrative criteria for each component. This assessment practice falls in alignment with a recommended best practice from the American Evaluation Association for objectively and comprehensively assessing projects in courses.

Some problems arise because this is a university, not a company setting. The most serious is scheduling. Team members have different schedules. Finding times for meetings and integrations is difficult. We have organized their 2-hour labs to be at the same time, but more meetings are required. The most successful teams seem to be the ones whose meetings are the most productive. The team that did the kitting cell video claimed never to have all night sessions; rather, they had an open meeting time every day from 5-11pm in our computer laboratory. Team members came and went as they could and did other homework when their presence was not required. Team meetings came together dynamically and on demand. We currently have pre-reserved five rooms for 3 hours each evening Monday through Thursday. In addition, the facilities manager now keeps the computer laboratory open round the clock all weekend during the weeks of the team project.

One of the challenges that we face with our course redesign approach is under-performing team members. This problem is minimized when using problem based learning. However, on occasion, under-performance occurs within teams. To counter this effect, we have built peer-mentorship into the course. The highly capable managers can mentor the students directly or find
a team member to help. Importantly, under-performance is a real life occurrence in industries. Our intent with the peer mentorship aspect of the course is to build leadership in our team members in a safe, course-based practice environment. In industries, people can be fired and replaced for under-performance. Neither option is available to us. Instead, we have peer mentorship, peer pressure to perform up to the expectations of the group, and ultimately, grades and student evaluation to motivate students. Mentorship helps both the leaders and the under-performing team members. Team members receive advice from the mentors that leads to improved performance and mentors receive real life leadership practice in courses.

conclusions

Trying to improve the undergraduate computer science experience means finding ways of fulfilling the true needs of the student. For engineers destined for industry, our experiment provides the training that industry tells us is critical: working on teams, developing large systems, writing professional code. Our industry contacts are telling us that our plan is working. Our graduates report they are doing well. This experiment in pedagogical reform represents an attempt to deliberately link computer science coursework to skills and strategies that are required in computer science industries. The reform of the course pedagogy and curriculum has led to positive results for the students and has improved preparation for them in real life skills necessary for them in their future employment.

references
4. FIUT Award, 2004-2005 to David Wilczynski and Michael Crowley, Computer Science Department, Viterbi School of Engineering: Changing and Innovating Computer Science 201: Principles of Software Development
5. FIUT Award, 2007-2008 to David Wilczynski and Michael Crowley, Computer Science Department, Viterbi School of Engineering: Teaming Students from Computer Science 201 and 200 http://cet.usc.edu/resources/awards_grants/fund/winners/wilczynski_crowley.html
7. FIUT Award, 2009-2010 to David Wilczynski and Michael Crowley, Computer Science Department, Viterbi School of Engineering: Adding Managers to the 200/201 Experiment http://cet.usc.edu/resources/awards_grants/fund/winners/wilczynski_crowley_10.html
18. CAEE, 2010. Enabling Engineering Education Success: Final Report to the National Science Foundation of the Center for the Advancement of Engineering Education. CAEE.
appendix

Figure 1: The Kitting Cell

Figure 2: The Glass Line
Figure 3 – Corrective Action Plan

<table>
<thead>
<tr>
<th>Problem</th>
<th>Expectation</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Communication</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Lack of phone, email, text messaging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ No wiki documentation (task cards)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Meeting Behavior</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Missing meetings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Rudeness, Attitude</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Poor Performance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Not prepared at meeting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Missing deadlines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Unsatisfactory work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Plagiarism</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The **Expectation** is a description of the behavior we expect of you, which you have NOT done so far. The **Measurement** tells how you have mitigated the poor behavior for which you have been cited. The CAP will be **CLOSED** when the problem is solved.

Failure to successfully close this CAP will result in action as decided by the Professor. In general, the Professor will NOT be involved in a CAP if it is closed and the student has no further problems.

<table>
<thead>
<tr>
<th>Student’s Signature</th>
<th>Date</th>
<th>Manager’s Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor’s Signature (if needed)</td>
<td>Date</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>