This document is a brief report on the CS/CE ABET status, as of Summer 2009.

Prepared by Dr. Eamonn Keogh.

Executive summary:

We have hired a new faculty member to address a weakness in the area of Software Engineering group. Dr. Iulian Neamtiu has already made significant contributions to improving CS 180 (Introduction to Software Engineering, undergraduate level).
We have experimented with improving several offerings (discussed in the relevant sections below). For example, in CS179 we have brought in professionals for the industry to give guest lectures. These lectures are specifically designed to give weight to the need for life-long learning and maintaining and leveraging off knowledge of contemporary issues. These issues (ABET '*i*' and '*j*' respectively) were empirically observed to be weaknesses in previous offerings.
We are planning to do our annual alumni and employer surveys in August 2009. Because of the relatively poor response rate for the employer survey, we will experiment with a telephone survey this year.

New Faculty Hire to Address Weakness in Software Engineering Area:

Dr. Iulian Neamtiu was hired in 2008 to strengthen the existing Software Engineering group. In the academic year 2008-2009, Dr. Neamtiu has taught two Software Engineering courses, CS 180 (Introduction to Software Engineering, undergraduate level) and CS 245 (Software Evolution, graduate level).

Dr. Neamtiu has adjusted the lecture material for CS 180 to increase the use of formal methods in software construction. The use of formal methods in software specification and software validation leads to a better understanding of software requirements and software behavior, and prepares students to write higher quality software, i.e., more usable and less prone to errors. Moreover, the emphasis on rigor and formality gives students an opportunity to use their training and knowledge in mathematics and logic towards solid software construction methods. For the project part of the course, Dr. Neamtiu has chosen an approach that emphasized flexibility in team formation and implementation strategy, while adhering to strict documentation and schedule guidelines; this approach has the role of exposing students to realistic software development practices, as well as issues and solution that appear in the development of large project in a multi-person team.

Finally, in both projects and lectures, Dr. Neamtiu has tried to emphasize the relationship between software engineering and other engineering disciplines: given the societal impact

software products have today, the importance of ethical conduct in the context of software development and teamwork cannot be overestimated. For the graduate-level CS 245, Dr. Neamtiu has adjusted the lecture and project material to put more emphasis on introducing students to research. The majority of lectures consisted of presenting and analyzing cutting-edge research papers. This procedure familiarizes students with the critical thinking required for evaluating related work, and provides students with a broad enough background so they can start conducting their own research in Software Evolution.

The students also had to conduct a research-quality project, on either a self-chosen topic, or a topic suggested by the instructor. The goal of the research-oriented project was to encourage students to explore a topic in depth, identify challenges, come up with their own novel ideas for addressing these challenges, and document their project in a similar manner to writing a research paper.

Looking ahead, Dr. Neamtiu plans to improve the content and running of CS 180 and CS 245 (based on past experience with these courses) and also take on teaching other software engineering courses, at both undergraduate and graduate levels.

Course Offering Summaries

These summaries will be discussed at Faculty retreat in September 2009.

CS 180, Spring 2009, Instructor: Iulian Neamtiu.

What Changed

Lecture: I changed the textbook to “Fundamentals of Software Engineering (2nd edition)” by Ghezzi, Jazayeri, Mandrioli to increase the use of formal methods in software construction. I believe the use of formal methods in software specification and software validation leads to a better understanding of software requirements and software behavior, and prepares students to write higher quality software, i.e., more usable and less prone to errors. Moreover, this approach gave students “*an ability to apply knowledge of mathematics and science*”.

I tried to emphasize the relationship between software engineering and other engineering disciplines, i.e., the grave consequences of bad practice and bad products in software engineering. This was meant to ensure that students had "an understanding of professional and ethical responsibility" and understood “*the impact of engineering solutions in a global, economic, environmental, and societal context*”.

Project: I changed the team assignment strategy to half self-selection, half instructor-

imposed to better mirror team assignment in real-life software development, which should give students "*an ability to function on multi-disciplinary teams*".

As project topics, I chose substantial, realistic, contemporary applications such as online commerce, office software, instructional software, etc. The goal of these project topics was to both give students "*an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice*" and make sure they had "*a knowledge of contemporary issues*".

Quizzes: I used unannounced quizzes to gauge student understanding of material taught in previous lecture.

What Worked

Lecture: The informal feedback I received from students on emphasizing formal methods in software construction was positive.

Project: Students applied concepts taught in the lecture (software process, testing) to their projects without the instructor explicitly requiring them to do so, which means the concepts I taught could be successfully applied in practice. The team project approach I used emphasized flexibility in team formation and implementation strategy, while adhering to strict documentation and schedule guidelines and worked out really well. All teams were able to meet (and sometimes exceed) the requirements and meet the deadlines. Based on student feedback, I note that they became aware of challenges and solutions of getting a team to work together towards achieving a goal.

Quizzes: The quizzes proved to be beneficial in identifying the sorts of problems/topics students had difficulty with. Once I identified those problematic spots, I went over the material/did more problems until the feedback I received from students showed me the concepts were understood.

What did not work

One of the software verification topics (proving programs correct by using axiomatic semantics) was insufficiently practiced during the lecture---I dedicated it about half a lecture. The students told me that they felt the preparation I provided was insufficient. To address that, I picked up the topic again, did more exercises in class, and provided students with sample problems and solutions. Next time I am (or someone else is) teaching the class, an entire lecture has to be dedicated to the topic, and a homework on axiomatic semantics must be added, to make sure students grasp these concepts.

CS 161: Instructor: Laxmi N. Bhuyan

We adopted the new 4th edition of the book "Computer Organization and Design" by Patterson Hennessey. This book was adopted because it contains new materials dealing with advanced computer architecture concepts, such as pipelined design, instruction level parallelism (ILP), hazard detection and branch prediction. This gives a brief introduction to our students on how the state-of-the-art computers are designed.

CS 179M, Spring 2009, Instructor: Eamonn Keogh.

What Changed:

I have decided to have student do peer evaluation of other members in their group. I explain why to the students at the first meeting, when discussing ethics and profession responsibility.

What Worked:

I invited B. Wolfe, a produce manager from ERSI with 10 years experience to give a guest lecture on “Eliciting User Requirements”. The students seemed to appreciate her insights (based on anonymous feedback solicited at next meeting). Ms. Wolfe indicated ERSI would be willing to do this every quarter in perpetuity. I have sent an email to all faculty that will teach 179 in 2009/2010 to inform them of this.

I invited Dr. David G. Kay, computer scientist and lawyer at Donald Bren School of Information and Computer Sciences at UCR, to give a talk on “Computer Science and Ethics”. Based on anonymous feedback solicited at next meeting, students seem to feel that they need more exposure to intellectual property law basics.

Since the project was web-based I adopted a new textbook on web interface design, Steve Krug’s Don't Make Me Think! A Common Sense Approach to Web Usability. Based on the quality of interfaces and explicit references to various chapters of the book in the students binders this seems to be a good choice. Students liked that the book was terse and pragmatic.

Because of the student’s poor performance public speaking in my last offering of 179, I have created new presentation/materials on giving a talk (which is required of the students in weeks 8 and 9), I have shared this faculty that will teach 179 in 2009/2010.

What did not work:

Students have problems meeting deadlines, and I was forced to give multiple extensions. Dr. Frank Vahid has useful slides on time management that he uses in workshops for entering students. We should adapt and reinforce in the beginning of 179.

CS 10, Spring 2009, Instructor: Kris Miller.

What Changed:

We started using a new graphics package this quarter in CS10. The new graphics package adds color, ability to capture key presses, and images, while still keeping the simplicity of use we had with the previous graphics package. This graphics package is in-house (developed by one of our undergraduates), so we are no longer tied to our textbook for its graphics package. By just adding color and key presses, the students are able to come up with much more exciting applications compared to what they could do with the old graphics package.

We restructured and redesigned most of the lab content this quarter. We added exercises to take advantage of the new graphics package, but also added more "easy" and "difficult" exercises to better handle the widely different levels of programming experience we see in CS10 students. We've always struggled keeping a balance between not discouraging the absolute beginners and not losing the interest of the more experienced programmers. In the past we had some more advanced exercises that students could get bonus points for completing. Unfortunately, this meant only the students that typically don't need the bonus were getting the bonus points and we were still frustrating the beginners since they could seldom get the bonus points they really needed.

So, this quarter we added more exercises and restructured how the points were awarded for the labs. Now, the students have to finish some minimum number of the "easier" exercises to get credit for the lab. If they finish these before the end of the lab period, they are required to work on the later more difficult exercises, but are not required to finish them in lab to get full credit. Typically, I would go over the most difficult exercise in the next lecture. This was the first quarter I can remember when I didn't have students complaining about the labs being too hard or on the other hand, complaining about having to stay the full 3 hours. Even better, students saw the more advanced exercises as a challenge, despite the fact there were no specific points awarded for them, not even bonus points now. I quite often had students coming to office hours or emailing me with questions about these exercises.

We redesigned most of the home programming assignments. Again, this was motivated by the added functionality of the new graphics program, but we used the opportunity to address some of the deficiencies of the old assignments. One of the negatives with the old moonlander assignments we had been using for years was that it seemed like we were

spending more time explaining basic physics to the students in office hours rather than explaining the programming concepts. The toughest physics concepts in these new assignments were changing the direction of a velocity or changing the velocity by some constant. I had just as many, if not more, students coming to our office hours this quarter, except now we were spending all of our time explaining programming concepts.

Another change before the quarter began was obtaining a different classroom than the one assigned to me. I was originally assigned the same classroom I had for the Winter quarter. During the Winter quarter I had two sections of CS10 in HMNSS 1503 and one in Sproul 1340. I use interactive and collaborative exercises in lecture to help reinforce the concepts I am lecturing on. I noticed that the students in HMNSS 1503 were not really able to work effectively in groups since the seats are theater-style and very cramped. The students in these 2 sections noticeably struggled on the quizzes and exams compared to the students in the other classroom where the seats could move around.

Another change was adding points to the collaborative exercises in lecture. I did this to hopefully increase attendance in the lectures as well as give more motivation to the students that did attend to take the exercises more seriously. While it did appear to affect the students that were coming to lecture, it did not increase attendance. I believe this was because I did not post these points in their online gradebook in a timely-enough manner. Next quarter I will make sure these points gets posted much sooner.

Finally, I sent out the Early Warning Program notices earlier this quarter. Last quarter I did not send out the warning emails until after their midterm scores. By the time they met with the EWP mentors, it was really too late to help the students catch up. This quarer I based whether a student would get a warning on their quiz scores after their 3rd quiz (Monday of week 4). Many students were now able to meet with the EWP mentors before the midterm.

What did not work:

The students still really struggle with understanding the written specifications for the programming assignments. I will add some diagrams and generally clean up the specifications based on the feedback I got from students and TAs this quarter.

The new assignments did not emphasize good testing and debugging skills and the new edition of the textbook does not have a separate chapter on this topic anymore. In fact, some of the new assignments actually made it harder for the students to debug their own program. I will add some new lab exercises specifically aimed at getting students to unit test their functions and learn basic debugging skills. I will also add an assignment that focuses on testing functions separately from the program they will later use it in.

Finally, the learning curve for all of the new tools along with learning to use the Linux command line is an onging concern of mine. I will spend the summer researching the use of a simplified IDE for CS10 students. Right now I am looking at simplifying a widely

used IDE like eclipse or possibly using an IDE like CodeBlocks that is similar to eclipse but much easier for beginning to students to use and even setup on their home computers.

CS 13, Spring 2009, Instructor: Teodor Przymusinski

What Changed:

This is the first time the course has been taught. I therefore selected a new book Engineering Problem Solving with C++, 2/E, Delores M. Etter, Jeanine A. Ingber, ISBN-10: 0136011756, ISBN-13: 9780136011750, Prentice Hall 2007 and a new syllabus for the course.

What Worked:

The book generally worked out quite well. It includes a number of case studies and exercises based on problems from physics, chemistry and engineering and thus should make the students better aware of the applications of programming to other branches of engineering.
Every week the students had to do a homework assignment and a programming project. As a result, they should have gained a lot more programming and problem solving experience.

What did not work:

In spite of the fact that CS10 is a required prerequisite, many students had a very limited knowledge of C++ to begin with. I was therefore forced to discuss and review a number of topics that should have already been covered in CS10.

CS 100, Spring 2009, Instructor: Teodor Przymusinski

What Changed:

I taught this course already a number of times and I am still not quite sure what the right approach is. This time I significantly increased the number of programming projects and the overall weight of lab assignments, including homework and in-lab assignments. The reason was that software engineering principles are most of the time stated in a very imprecise and informal language which makes it difficult to use them for student testing.
I also significantly expanded the coverage of software components, including the discussion of Java Beans and Microsoft's .NET framework.

What Worked:

Every week the students had to do a homework assignment and a programming project. They also had to do a pretty realistic and involved term project that should provide than with a much better understanding of software requirements, design and coding.

What did not work:

I used the same book as before. I very much like the scope of the book's coverage but I consider it also to be very poorly written. It is also hard to read for the students. So far I was unable however to find any other textbook that would better satisfy both of these aspects.

CS 12, Spring 2009, Instructor: Brain Linard

I have a requirement that students must pass 6 out 8 programming assignments in order to pass the course (a pass on an assignment is 80%). In order to encourage them to go back & learn from their mistakes and from the graders' feedback, I allow them to fix & re-present a failed assignment within the next week and obtain a passing grade.

This is the only technique I have ever found for "forcing" students to go back and review their work and improve it.

However, this quarter, more than ever before, many of the students were simply "parking" a barely intelligible botch of a program by the initial deadline in order to get say 10%, and then actually doing the assignment (poorly!) in the following week: in effect, they were using the policy as a kind of back door late submission, with the result that they were getting further & further behind.

Corrective action for next quarter: I have changed the policy a little: they may receive only 40 additional points for a reshow, so in order to get a passing grade on an assignment they have to get at least 40% on the first attempt, and thus have to spend more time on the assignment at the proper time.

Another observation: especially in cs12, I prepare a number of tutorials on the "harder" topics to supplement the lectures, the text & the lab materials. If I may say so myself, these tutorials are excellent: tightly focused, precisely targeted, and very accessible. To my dismay, I found that a quarter or more of the class was not even reading them! (moodle has a "big brother" mode that tells me exactly who accessed what and when). So after the mid-term, I prepared a chart showing the correlation between use of two of the tutorials and mid-term grades (students who read both did significantly better on the exam, of course).

It seemed to have an effect for a couple of weeks, but unfortunately it then seemed to wear off again.

Corrective action for next quarter: I am discussing this with the undergrad committee, and preparing a report for the faculty retreat in September 2010.

And again: in both classes, I have a weekly quiz (8 quizzes => 15% of the course grade). They are challenging (I have a reputation that “you can't guess his multiple choice questions”), but every question relates to material explicitly covered in lecture. I email students their score the following day, together with a pdf chart of the class stats for each question.

Then, on the next quiz, I repeat (generally unchanged!) any question that more than half the class got wrong - so everyone knows in advance at least 1 or 2 of the 10 quiz questions. And guess what? About a quarter of the class regularly got them wrong the second time around!

And again: I give students detailed & complete reports on their grades throughout the quarter, and specifically just before the final exam, so they know to within 1/10 point where they stand.

This quarter, I suspect that a number of them saw that they could pass even with a poor showing on the final exam (20% of the total), and so didn't bother studying for it: and the results (for cs61) were among the worst I've ever had, for an exam that very closely followed both the sample exam I gave them, and past finals.

Corrective action for next quarter: I am adding a requirement that they cannot pass the course without passing the final exam.

Especially in cs12, I frequently have the class work (in teams) on some simple exercise, either to prepare for a new topic, or to practice the current topic. Usually, I allow 10-15 minutes for these exercises, but this quarter, I twice gave them something more substantial, requiring up to 30-40 minutes of the class. Both times, it generated real energy in the class, and made them much more receptive to a detailed analysis of the topic.

I can't do lengthy in-class exercises like this too often, but I will do at least two or three of them.