STATEMENT OF TEACHING PHILOSOPHY

EMILY HILL

Engaged students learn. Regardless of subject, engaged students are more likely to gain a deeper understanding of the material and retain the information after the semester is over. As a teacher and mentor, I try to engage students by involving them in the classroom, using frequent and varied evaluation techniques, working with students individually through research, and by encouraging diversity.

INVOLVING STUDENTS IN THE CLASSROOM

In general, I attempt to involve students in the classroom by increasing class participation, imparting my enthusiasm for computer science , and motivating the material.

Increasing Class Participation

My approach to engaging students in the classroom is inspired by a mathematics professor at my liberal arts college. When deriving a proof or solving a problem, he would ask the class to participate. But rather than wait for the classroom to answer, he would randomly selected an index card containing a student's name. The professor would then work with the student to solve the problem or derive the proof for the rest of the class. When necessary, the professor would prod the student, ask leading questions, or review earlier concepts. Although rather mortifying for students in the beginning, this method of engaging students has a number of benefits:

- Motivates the students to pay attention in class because they never know who will be called upon next
- If the selected student has not been paying attention, by focusing on the student individually, the professor can quickly get the student up to speed and provide a review for the rest of the class
- Students learn from other students' struggles, even if they are uncomfortable asking questions themselves
- Creates an open environment for questions as the students become more comfortable with their peers in the classroom, while ensuring no one student dominates the class
- Ensures the professor does not move faster than the entire class can grasp the concepts

As a student, I will admit that I found this method of learning to be extremely painful, since I was embarrassed to reveal my lack of knowledge in front of my classmates. However, I gradually realized that no student in the class understood everything, and was able to relax and focus on learning the material---something I rarely accomplished during undergraduate lectures.

I can adapt this technique for computer science courses by carefully preparing my lecture material to include focused questions. For example, in a software engineering class I might select students to name challenges in software engineering based on their own programming experience. In an introductory programming or algorithms class, I could select a student to help derive a specific algorithm. Although it requires extra preparation in the beginning, I hope this technique will become a natural part of my teaching style over time.

Imparting Enthusiasm

If the professor is bored, why should the students be excited? When engaging students to learn, it helps to be enthusiastic about your subject. In computer science, I'm particularly passionate about automation. Computers are the quintessential automation tool, capable of infinite variation through software. All the problem solving skills computer science teaches are driven by automation, whether the problem is finding a software bug or developing a more efficient algorithm. As a teacher, I try to infuse my lectures with as much

passion as possible, using my presence as a performer to help me communicate my enthusiasm and gauge my audience's response.

Not only am I enthusiastic about the subject, I'm also energized by helping the students understand the material. For example, in lab I noticed students struggling to understand shell scripting and makefiles. To help the students realize the benefits of more advanced command-line software development, I created a shell-scripting and makefile tutorial based on the instructor's example from lecture [see attached tutorial]. In the tutorial I not only explained each line's action, but also how the techniques would reduce their development time.

Motivating the Material

Students are naturally more motivated to learn material that they are interested in or that they see value in beyond graduation. Thus, I also attempt to engage students by motivating the course material, showing how the material is relevant to their future success, and relating the material to their existing knowledge.

For instance, one of the course objectives in an introductory computing class for non-majors is to learn about computer architecture and components. Because I felt the students would most likely use this information in the future to help them purchase a computer, I designed a build-a-computer project [see attached assignment]. Given a budget and a list of components, the students were asked to build a computer that would suit their needs and explain their design decisions based on their own computer use. The students seemed to enjoy both the personalized nature of the assignment as well as its relevance to their future computing needs [see attached feedback].

Furthermore, introductory programming and software engineering courses can be motivated by actual programming and software engineering failures. For example, Verizon's 3G network in Philadelphia was brought down by a missed semicolon, and Washington DC lost cell data service due to an uninitialized variable. More severe examples include the Mars Climate Orbiter crash in 1999 due to ill-defined interfaces (one component used metric units, whereas another used English), or the deaths of cancer patients due to a race condition in the Therac-25 software, which could have been prevented using verification techniques.

Engaging Students through Evaluation

Although engaging students in the classroom is the first step in teaching, frequent and varied evaluation can provide further motivation. Even absent students will generally attempt to learn material on their own in order to maintain their GPA. To keep students motivated to learn each week, I evaluate class participation and give out short weekly quizzes. Weekly quizzes serve two purposes: quizzes encourage students to learn the material as the course progresses, rather than waiting until an exam; and quizzes give a progress report to both the student and professor, suggesting what topics may need to be studied further. For example, when my introductory computing class had trouble with number systems and struggled to convert decimal numbers into binary, I spent extra time going over the material in class and investigated alternate ways to present the lesson.

Exams and projects help motivate students to learn throughout the semester. When possible, I like to assign both individual and group projects that help develop students' analytical problem solving skills and grow their collaborative abilities. Especially for programming and software engineering courses, I try to give programming assignments that start from scratch as well as start from an existing implementation. When starting from scratch, students can exercise their full creativity and see the entire design process from requirements to development and testing. In contrast, starting from an existing code base or requiring the use of a specific library exercises different development skills and more closely mimics current software engineering practices.

Lastly, in my desire to see my students succeed after graduation, I endeavor to include at least one oral or written assignment. For example, in my introductory computing class I assigned a paper about the ethical implications of a computing technology of the student's choice. Not only did the assignment exercise the student's writing skills, it also encouraged them to explore computer science from a different perspective

(human rather than technical). In a programming or software engineering course, I could ask students to give an oral presentation about their approach to solving the latest project. Such a presentation would not only encourage critical thinking about design and implementation decisions, but also motivate students to discover innovative solutions to existing problems.

RESEARCH MENTORING: PROVIDING INDIVIDUAL ATTENTION

I believe research mentoring is one of the best ways to motivate students to pursue computer science. With the individualized attention from research and independent studies, students are completely engaged in the research problem. Plus, one-on-one and small group meetings often lead to mentoring opportunities that can provide guidance and motivation to pursuing careers or graduate studies in computer science.

During my dissertation research I had the opportunity to mentor seven undergraduate students. Because a large portion of my work focuses on analyzing code (i.e., reading, not writing code), the nature of my thesis research lends itself to working with students at all levels, from freshmen to graduate students. In fact, I have mentored and successfully collaborated with sophomore, junior, and senior undergraduate researchers. In addition to my thesis research easing student collaboration, my diverse software engineering research background enables me to work with students in a variety of research areas.

Because each student is unique, my mentoring style may vary from person to person. Although I always broadly outline projects for the time allotted, there are some variables with a new student that cannot be anticipated. For example, a student's actual knowledge may be more or less than what it looks like on paper, requiring varying amounts of lead time before the actual research begins. Thus, I try to start off with a short, week-long research assignment to get a feel for the student's skill set and interests before nailing down a specific schedule with deliverables. From my own experience, I know undergraduate researchers sometimes need a push to keep going and maintain a steady level of progress, and I try to set reasonable short term (daily or weekly) and long term (monthly) deadlines, based on the student's skill level and work habits. One of the rewards I look forward to in leading an active research program is observing how others become competent researchers under my leadership and mentoring as we work together on research projects.

ENCOURAGING DIVERSITY

Lastly, I attempt to engage students in computer science by encouraging diversity. I have a history of commitment in attracting people to pursue computer science, especially woman and minorities. As an undergraduate student I initiated and developed two high school outreach programs: one to teach high school students about computer architecture, the other to teach high school girls Flash and HTML basics as well as discuss career opportunities in computing. As a graduate student, I helped co-found CISters, a group that organizes activities to recruit and retain women in technology-driven fields at the University of Delaware. As part of CISters, I co-organized a case study of the status of women in computing at UD, which resulted in travel funding for 10 graduate and undergraduate women to attend Grace Hopper. Finally, I have participated as a co-mentor in the CRA-W/CDC Distributed Research Experiences for Undergraduates program for women and minority students. Of the four undergraduate women and minority students I've mentored, three are either attending or plan to pursue graduate studies in computer science. With both teaching and research activities I hope to continue increasing the number of women and minority students pursuing careers in computer science.