

# TEACHING STATEMENT EMERSON MURPHY-HILL

Based on my experience as both a student and as an instructor, I use six guiding principles when teaching.

**Action and Reflection.** Students may be able to cram for a test, but it is much more important that they retain and use the knowledge learned in the classroom throughout their careers. I believe that it is vital to long-term knowledge retainment and use that students actively participate and reflect on course material. For example, in one classroom activity I sought to illustrate that customers sometimes state ambiguous requirements and that software developers need to resolve that ambiguity before investing in an implementation. In this activity I played the role of a customer and asked several students to sort a pile of coins. After each student sorted by denomination, I told them what the customer *meant* was for the coins to be sorted by year, and that the work would have to be performed again. Upon reflection, students came up with several ways that they could avoid ambiguous requirements, such as by developing an early prototype and asking the customer, "is this like what you want?"

**Clarity.** Many topics are not easy to understand, and a poor presentation makes understanding especially difficult. I believe in teaching material as clearly as possible, through frequent examples, explicit learning goals, and focused repetition, among other techniques. For example, to clearly demonstrate that robot software is required to balance deliberative and reactive behavior, I showed students a video of Boston Dynamics' Big Dog robot recovering walking after being kicked from the side. I then used the demonstration to motivate why the requirements of mobile robots impacts software architectural decisions. One student noted that such examples "do stick in my mind better and help me understand that principle."

**Context and Relevance.** Many students have difficulty understanding topics because they seem divorced from the students' own experiences. I believe in putting material into a context that students will understand and appreciate, and in making it relevant to the world outside the classroom. For example, when explaining the practice of refactoring, I showed students how to refactor code using a development environment that they had used previously, thereby relating the lesson to their past experience. I also showed examples of refactorings that open-source developers performed on real software, thereby showing that the practice is relevant in a larger, practical context.

**Diversity.** Diverse students make the classroom experience richer; bringing with them different perspectives, backgrounds, and opinions, which everyone can learn from. For instance, non-traditional students returning to finish their degrees often bring real-world experience that can provide motivation and context for traditional students. On a personal level, as a student I took several classes with an older student who had worked for the National Security Administration; on many occasions this student vividly illustrated both computer and physical security from his work experience. I plan to encourage diversity in the classroom through outreach and mentoring of traditionally underrepresented students.

**Scholarship Skills.** While mature students, especially nascent researchers, often have the knowledge to advance scientific understanding, they often lack skills for using and communicating that knowledge. I believe that students need to be advised about the skills necessary for successful scholarship. One scholarship skill that I strongly advise students to cultivate is to regularly have their work critiqued by peers. For example, I created a venue for students to practice a research talk in front of their peers to help prepare them for their oral qualifying exam. As a consequence of participation, many students' ability to communicate their research improved; as one student remarked, "the questions [peers] asked were repeated at the [examination] and I feel glad I was ready for them."

**Continuous Improvement.** I am not a perfect teacher; in the classroom, some things will not go as planned, some students will not grasp the material, and some days will be better than others. I believe that the path to being a better teacher is by frequent reflection on my teaching activities and their outcomes. I use several techniques to continuously improve my teaching, including peer review, informal and formal student evaluations, and, after every lecture, a postmortem of what went well, what didn't go well, and what surprised me. For example, in one class, mid-term evaluations showed that classroom activities were useful for the majority of students ("very effective in recapping the concepts," as one student wrote), yet some students did not understand what point I was trying to make with the activities. This feedback allowed me to continue the activities while making improvements, such as by more explicitly connecting activity outcomes to learning objectives.

I am interested in applying these principles to teaching both graduate and undergraduate students. Specifically, I am most qualified to teach courses in software engineering and programming languages, but I am willing to teach a range of courses, including human-computer interaction and databases.