Teaching Statement

Michael Ekstrand

Much of my approach to teaching is guided by requirements, a concept familiar to computer scientists and software engineers. For each course, assignment, lecture, etc., what are its learning objectives? What does it need to accomplish? Good software design and good education design have a lot in common.

At a high level, when designing a course, I first try to identify the purposes the course serves. What do the department, the students, and the teachers of classes that list it as a prerequisite expect it to provide? I also strive to let requirements guide my teaching at lower levels. What does each lab or class period need to accomplish? What does this assignment need to test or practice? What kind of assignment or activity would support a particular requirement?

I put this into practice when I taught CSCI 1902 (UMN's CS2 course) in the summer of 2012. This class concludes students' primary training in the act of programming itself; later classes teach particular applications and techniques, assuming that students who have passed 1902 can do the mechanics of programming with standard data structures. They also expect students to be familiar with core vocabulary and concepts.

I designed the course to meet those objectives: to equip the students to go about the practice of programming as they studied more advanced topics or work on internships and side projects. This worked itself out at many levels: prioritizing material to help students make good software decisions, teaching concepts so as to make good habits more natural, etc. I also added a new feature: small weekly programming assignments. Previous offerings I had TAed for gave students 5–6 major assignments. Several involved writing components that fit into a broader program written by the professor or TA; this is valuable, but students don't get much practice with the process of writing a program for some task.

I intended the small programming assignments to provide this practice. Each one could be completed in about 20 or so lines of Java, and I aimed for students to be able to complete each one in an afternoon. By the end of the course, each student had written 7 complete programs on their own, in addition to working on several more substantial projects in pairs.

I have also co-taught a MOOC, *Introduction to Recommender Systems*, with Joseph Konstan. The class has had 28,000 students registered, with almost 3000 of them still engaged at some level halfway through the course. Requirements and learning objectives again played a large role in our development of this class, particularly in developing assignments and assessments that can scale up to thousands of students but still provide useful practice and learning opportunities.

My experience and training most directly equip me to teach introductory computer science, HCI (including collaborative/social computing), machine learning, information retrieval, data mining, and artificial intelligence. I also have a solid general CS background and a good grasp of algorithms, theory, and general programming practice, and would be able to teach classes in these areas and a variety of application areas if needed.

I have been programming for many years, love the craft, and believe that computers have an incredible potential for good in our society. I hope to encourage my students to passionately pursue excellence as they harness computing to make the world a better place.