Teaching Philosophy and Interests

Michael Ekstrand

Computer scientists are familiar with requirements. They establish what a program should do and may help us understand how it fits into the broader world. In software engineering classes, we teach our students to identify and document the requirements for a program or system, and let these requirements drive the design. Lightweight development methods also emphasize understanding what it is that the program needs to do in the form of documentation or tests.

I find this concept to also be a useful guide to my teaching. 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.

Applying Requirements

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, often modeled after basic Unix text utilities, in addition to working on several more substantial projects (optionally with a partner).

I don't have hard data on how effective these assignments were, but students seemed to be able to complete them reasonably. Several of them commented favorably about the assignment structure, including small programs, on the teaching evaluations (one student even wanted more small programs).

Online Education

In addition to traditional classroom experience, I also have experience designing and teaching a MOOC. With my adviser Joe Konstan, I am co-teaching *Introduction to Recommender Systems*, offered via Coursera. 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.

We dual-offered this course as a for-credit graduate-level course with an inverted classroom format: students watched the Coursera videos and completed their assignments online, coming to normal class periods if they wanted for discussion and answering questions. For-credit students also had direct access to the instructors and TA via e-mail and office hours. We used the opportunity to study the relative learning outcomes between these different formats, in a paper to appear in the first ACM Learning @ Scale conference.

I think MOOCs are an interesting experiment in education that we don't yet know how best to use and deploy. One reason I want to be a professor provide an excellent education to a wide range of students; I am excited to see what we learn in the coming years about how best to use traditional classrooms, small- and large-scale online education, and other instructional techniques to that end.

Other Experience

I have also worked with and mentored a number of undergraduate, masters', and junior Ph.D students in the course of my studies. In particular, many of them have contributed to the LensKit project, where I have supervised their programming work and helped them to conduct research with it, write papers describing the results, and apply for research funding. Requirements and learning objectives have also been a major component of how I relate with them: understanding what their goals are for their research experience and working to find projects and a path to get them from their current skill and experience level to where they want to be.

Teaching Interests

I am interested in teaching a wide range of topics. My research and teaching experience most directly prepare me to teach human-computer interaction (including collaborative and social computing and user experience design), recommender systems, machine learning/data mining, and information retrieval, as well as introductory computer science courses. I also have a strong general computer science background, particularly in programming languages and theory; I would be able to teach programming languages, compilers, theory, and web/network programming, as well as software engineering.

I have a particular interest in developing courses on recommendation and personalization technology, both from a human-centered research perspective and on the engineering details of how to build and deploy large-scale, data-intensive software systems for users.

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.