Teaching Statement

Nathaniel Nystrom

Over my graduate career, I have had many opportunities to teach and have found it to be a highly rewarding experience. I enjoy inspiring students to pursue their interests in more depth and to develop interests in new areas. I find the best way to understand material is to explain it to others.

At Cornell, I have been a teaching assistant for a graduate-level software engineering course, for a graduate-level programming language course, and twice for an undergraduate-level compilers course. I was also the instructor of a four-week short course on Unix. Earlier, at Purdue, I was a teaching assistant for an undergraduate software engineering course and an undergraduate web programming course.

When I taught the Unix short course, I went through the material taught in the previous few versions of the course, and reorganized it to focus on getting the students up and running faster, and dropped some topics that seemed out-of-date. Dropping some topics freed up a lecture at the end of the course to address assorted topics that students had specific questions on. I also put emphasis on finding and reading the Unix documentation.

In lectures, I demoed each topic to the class using the projector and my laptop and summarized it on the board. Using the projector, if a student asked a question about a particular command or program, I could demonstrate the answer right there. I encouraged questions and tried to make the lectures as interactive as possible. Before the first lecture, I made an effort to learn students’ names using the “mugshots” provided with the list of enrolled students. Putting lecture notes online before, or just after class, provided reference material they could use for assignments and in the future.

There were several short homework assignments to help reinforce the lectures. Since the course was pass–fail, I was very lenient with respect to grading and deadlines. Grading was a way to provide feedback and alternative solutions on assignments, rather than to evaluate or rank the students’ performance.

For the software engineering course at Cornell, I taught a recitation section once a week with another graduate student. The lectures given by the professor were largely theoretical; the recitations covered practical aspects of software engineering aimed at helping the students with their projects. Conveying some of my experiences working in industry enhanced the course.

While a T.A. for the graduate programming language semantics course, I created and graded assignments and exam questions, and ran a review session for the midterm exam. For programming assignments, much of the code was provided, allowing the students to concentrate on a core language feature or transformation based on the lectures given in class.

I was also twice a teaching assistant for Cornell’s undergraduate compilers course. The course provides a nice blend of theory and practice, and is a great software engineering course. Students work in groups of 3–5 to develop a complete compiler over a semester. Group projects provide experience that will valuable in industry. In this course, I developed and graded assignments and exam questions. The second time I T.A.ed the course, I wrote the midterm exam and filled in for the professor when he was out of town. I also help to design the source language for the compiler project.

My philosophy is to teach students core principles, motivated with practical examples. Some people learn better from examples and some learn better by deriving results from the underlying theory, I find using both provides the best way to get the material across.
Students should be encouraged to write and give talks. In the programming language course I T.A.ed, students took turns taking lecture notes to be polished up and distributed to the rest of the class. This gave the students practice with technical writing and with explaining an unfamiliar topic. I ran a programming languages discussion group at Cornell, in which students—myself included—gave talks on their research, on recent papers, or on whatever else happened to interest them. I also participated in several seminars throughout my graduate student career, usually giving one or two talks per semester in each seminar. The feedback on these talks improved my own presentations and I think other students would benefit from the experience as well.

Recently, I have been working with another graduate student, Xin Qi, on nested inheritance. With my advisor, I have been helping to supervise Xin’s research, and together, we’ve designed and implemented a programming language and have submitted a paper. Mentoring another student has been a great experience and look forward to doing more of it.

I eagerly anticipate embarking on an academic career. I would like to teach classes on compilers, programming languages, software engineering, and also lower-level programming, computer architecture, and systems classes. I also look forward to collaborating with other faculty members on keeping the curriculum up-to-date to better prepare students for their careers.