

Teaching Statement

Nathaniel Nystrom

<http://www.nanocow.com/nystrom>

I have had several 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.

Most of my experience was as a teaching assistant for several courses at Cornell and at Purdue. I was a teaching assistant for both graduate-level courses (software engineering, programming languages) and for undergraduate-level courses (compilers, web programming). My primary role was to hold office hours—answering student questions and helping on their assignments—to run review sessions, and to grade assignments and exams. I have also been the instructor of a recitation section and the instructor for a short course on Unix. Since leaving graduate school, I have also given tutorials on the X10 programming language.

As the instructor for a Unix short course at Cornell, I was responsible for giving lectures, writing assignments, grading. Before starting the 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course, and reorganized it to focus on getting the students up-and-running faster, and dropped some topics that seemed out-of-date.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 material allowed for a bit more depth on the remaining topics and freed up time to answer more student questions in class. I also put emphasis on finding and reading the Unix documentation.

In lectures, I demoed each topic to the class, projecting a terminal on the screen, and summarized the topic on the board. Using the projector, if a student asked a question about a particular command or program, I could demonstrate the answer immediately. 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 class, or as soon as possible 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 Unix course was pass–fail, I was very lenient with respect to grading and deadlines. I used grading as a way to provide feedback and alternative solutions on assignments, rather than to rank the students' performance.

For the software engineering course at Cornell, I taught with another graduate student a once-per-week recitation section. 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. Before entering graduate school at

Cornell, I had been a software engineer at Hewlett-Packard. I believe that conveying my experiences working in industry enhanced the course.

Students should, if practical, be encouraged to work in groups on large projects. Software development is rarely a solitary task, and group projects are a way to expose students to this experience. Both the software engineering courses and compilers courses required students to work in groups. Students work in groups to develop a complete optimizing compiler over a semester. I prefer this approach over having students work alone to develop a less complete compiler. Making the project large enough that a single person cannot complete it alone is essential. Groups of 3–5 seem to work best: larger groups tend to be overwhelmed by the communication overhead; smaller groups have trouble completing the 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 to give talks. In the programming language course I TAed, 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. In the software engineering and compiler courses, students were required to write documentation.

I also ran an informal programming languages seminar at Cornell, in which students, myself included, gave talks on their research, on recent papers, or on whatever else happened to interest them. These kinds of seminars are an essential part of graduate-student education, requiring students to give a few talks per semester (or else forfeit the free food). Professional development courses for graduate students should also be utilized.

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