Teaching Statement
Daniel Fryer

Teaching Experience & Interests

My teaching interests largely follow from my research interests, with a focus on systems programming and dependability. At the undergraduate level I am comfortable teaching in the areas of programming, software engineering, operating systems, architecture, databases, video game programming, and networking. At the graduate level I would also be interested in offering topics or seminar courses on storage, operating systems and virtualization, and dependable systems.

As a graduate student I had the privilege of teaching the introductory operating systems course at the University of Toronto. While I had previously delivered tutorial content on the mechanics of operating system programming and debugging, this gave me the opportunity to deliver conceptual content through lectures, modify and develop assignments, design fair evaluation strategies and manage teaching assistants. While the coursework was challenging, I was pleased that outcomes were in line with previous iterations of the course.

I have also been privileged to hold a mentoring role for other students conducting research. Several students (graduate and undergraduate) contributed to the development and testing of Recon, my thesis project. While the undergraduate projects required careful management of both scope and related work in order to achieve success (e.g. developing a failure injection tool for the file system, or implementing a predefined set of consistency checks), graduate students were usually engaged while identifying the direction for their own research to take. I find that the most effective working relationships depend on cultivating a habit of frequent communication by sharing problems and challenges from my own work and encouraging them to do the same.

Teaching Philosophy

My primary objective in teaching is to stimulate curiosity about a subject. In order to find a problem interesting, students must first understand why the problem exists in the first place. For instance, successfully teaching about virtual memory requires the students to be engaged in thinking about how a program they might write uses memory, and then to consider the case of coexisting applications and the problems this raises, before appreciating the tradeoffs of virtualization. For students to engage and think critically, ensuring that problems are well-motivated is as important as explaining the solution. Working with a consistent, concrete and relevant example scenario is one way to avoid losing the point – for example, in an operating systems course, keeping a particular application (e.g., a web browser) in mind in order to explain the necessity of different operating system services.

My secondary objective is to prepare students for challenges they might meet in the workplace, future coursework, or in research. Assignment material should reinforce and demonstrate course concepts, but it also becomes a confidence building exercise as students dive in to a new skillset. In this respect, excellent students need as much attention and mentorship as
struggling students in order to meet their potential. I believe that assignments should contain an explicit delineation between basic and advanced (or bonus) material in order to engage advanced students without penalizing the majority.

I also believe in the value of group projects for developing real-world interpersonal skills. However, the assignment of group work must be done carefully. In particular, while motivated students do learn from each other, the collaboration is most effective when students have similar familiarity with the material. When assigning group work, affordances should be made for effective group formation, conflict resolution, and fair grading. Additionally, some students benefit from assistance in group formation.

Promoting student engagement during a class is important, providing natural breaks for students to process freshly-delivered information. One piece of student feedback that I found encouraging was that my use of the blackboard during tutorials helped pace the flow of information at a digestible rate, resulting in well-attended tutorials. Aside from asking or answering questions verbally (which benefits the more outgoing students) another effective approach is to have all students writing speculative answers and turning their work in at the end of class for participation marks. This tightens the feedback loop of instructor self-evaluation; rather than waiting for assignment or midterm results, weak areas can be reinforced in the next lecture.

Finally, I appreciate the value of sensitivity in interacting with students. In particular, I've listened to students share ways that casual assumptions about the hobbies and interests of computer science students leave them feeling like outsiders. Additionally, the experience of mature and nontraditional students can be negatively affected by assumptions about social networks within the class. An effective instructor does not avoid these issues but instead recognizes when their assumptions should be called into question or made to be explicitly non-normative.