

EMERY BERGER

Teaching Statement

My goal in teaching is to provide students with a deep understanding of the issues underlying programming language and software system design and implementation, preparing them for future research and/or development in either an academic or industrial setting. I believe in engaging students in the classroom with energetic lectures and interactive classes, challenging them with exciting coursework and projects that crystallize their understanding, and exposing them to current research problems.

1 Course-Related Initiatives

Below is a chronology of my teaching activities.

| Semester | Course | Level | Students | Comment |
|-------------|---|------------|----------|---------------|
| Fall 2002 | Robust Software Systems (CMPSCI 691P) | Graduate | 11 | New |
| Spring 2003 | Advanced Compiler Techniques (CMPSCI 710) | Graduate | 15 | Fully revised |
| Fall 2003 | Operating Systems (CMPSCI 377) | Undergrad | 41 | Fully revised |
| Spring 2004 | Advanced Compiler Techniques (CMPSCI 710) | Graduate | 5 | |
| Fall 2004 | Operating Systems (CMPSCI 377) | Undergrad. | 57 | |
| Fall 2004 | Topics in Runtime Systems (CMPSCI 691S) | Graduate | 8 | New |
| Fall 2005 | Operating Systems (CMPSCI 377) | Undergrad. | 40 | |
| Spring 2006 | Parallel and Concurrent Programming (CMPSCI 691W) | Graduate | 16 | New |
| Spring 2007 | Operating Systems (CMPSCI 377) | Undergrad. | 42 | Fully revised |

I have thoroughly revamped the required undergraduate *Operating Systems* course (CMPSCI 377), which I have now taught four times, as well as the graduate *Advanced Compilers* course (CMPSCI 710), which I have taught twice. Since 2002, I have also taught a number of graduate seminar courses covering recent results in my research areas (*Robust Software Systems*, *Topics in Runtime Systems*, and *Parallel and Concurrent Programming*).

My teaching has been well-received, both at the graduate and undergraduate levels. In student evaluations of overall effectiveness, I have received course evaluations of 4.63 (on a scale of 1 to 5) for my graduate courses, and my most recent undergraduate course evaluation was 4.46. I was recently awarded a *Lilly Teaching Fellowship*, an award given annually to eight junior faculty members across the campus, in recognition of teaching excellence. As part of the Lilly program, each Lilly Fellow creates a new class or thoroughly overhauls an existing one. I fully revised our CMPSCI 377 *Operating Systems* course as my Lilly Project, building on the initial revision I made to that course in 2003, and beginning the evolution of this course towards being the foundational software systems course in our new undergraduate curriculum.

1.1 Undergraduate Education

I am most proud of my work on the undergraduate *Operating Systems* course, which I have taught almost every year since joining the department. I feel that this is the class that has most developed my skill as a teacher. I have incorporated an *active learning* approach in my course that has been successful at engaging students and ensuring that they develop a deep understanding of the material.

To engage the students, I make extensive use both of *in-class exercises* and technology. I use PowerPoint slides in my lectures, but these are not the typical bulleted lists of material. I push the animation capabilities of PowerPoint to their limit, and use highly graphical, engaging slides that make it easier for students to grasp algorithms by observing them in progress. The students love the slides, and they have been adapted for use at Tufts, UT-Austin, UConn, and MIT.

After illustrating algorithms on-screen, I then give the students in-class exercises that require them to demonstrate their understanding of the concepts or algorithms just covered in lecture. After letting them work on the exercise individually for a period of time, I have them check their results with their neighbors. This approach fosters a collaborative atmosphere: most students have no inhibitions about correcting or being corrected by their peers. These are low-stakes exercises: the students turn them in at the end of class, and as long as they have done some work, I give them one point towards their participation grade. I have found this combination of engaging lectures and in-class learning, with a small amount of credit as an added incentive, to be a remarkably effective way to keep interest and attendance high.

Last semester, as part of my Lilly Fellowship, I thoroughly overhauled the operating systems course, whose content has become less relevant to mainstream computer science over time. While the kinds of applications we run on computers has radically changed, the existing curriculum has not kept pace with these changes. In particular, the complexity of software systems has grown enormously in recent years. It is now commonplace to use client-side applications that run inside a web browser (itself a complex, multithreaded application generally written in C++). These applications are written in managed languages that execute on complex runtime systems with garbage collection and dynamic compilation. In turn, they communicate with web services, clients across the Internet, or a distributed service running across a cluster. Effective programming of these applications requires understanding of a broad range of material, much of it far beyond the material taught in a traditional operating systems course.

I taught the revised course in Spring 2007. It will eventually occupy a permanent place in our undergraduate curriculum as the foundational systems course, to be retitled *Software Systems*. The course now integrates the various areas of software systems—from architecture to distributed systems—and focuses on the cross-cutting concepts and foundational design principles. The goal is that students who take this course will understand and be able to develop modern large-scale applications. The course projects reflect this goal, and are organized around the implementation of a limited Google-like service from the ground up.

1.2 Graduate Education

In Spring 2003, I fully revised the *Advanced Compiler Techniques* course. This course had not been taught at UMass in a number of years. The previous class materials focused on traditional, FORTRAN-style array optimizations using a C-based compiler infrastructure built at UMass.

I shifted the focus of the course to analyses and optimizations appropriate for object-oriented programming languages such as Java. These languages require aggressive optimization in order to provide reasonable performance, and present a wide range of challenges and opportunities for compiler research, including dynamic re-compilation, object devirtualization, and cooperation or coordination with garbage collection. I also extended the scope of the course to go beyond optimization: it now includes topics like static analysis for error detection, an active area of research.

I based the lectures and associated PowerPoint slides, homeworks, and course projects on a Java compiler infrastructure that we are currently using at UMass, the open source Jikes Research Virtual Machine (RVM) originally developed at IBM TJ Watson. This course and associated materials are now linked from IBM's web page listing "Course Information Using the Jikes RVM"; my lecture slides have been downloaded over 5,000 times. I have now taught this course twice.

I have also taught a range of seminar courses that synthesize recent research in a variety of areas. Each course primarily involved lectures by me, with student presentations of papers and semester-long student projects towards the end of the course.

My first seminar course, *Robust Software Systems*, covered the numerous issues involved in building robust, high-performance software systems, spanning topics from algorithms and architecture to programming languages, operating systems, and software architectures.

In *Topics in Runtime Systems*, I addressed the "glue" that connects programming languages and operating systems. These systems have grown in importance as programming languages increasingly rely on runtime systems to support memory management / garbage collection, concurrency, and distributed execution. This seminar was organized around key literature in the area; students were required to submit summaries of each paper prior to class, in the form of a conference review.

Last semester, I developed a course on *Parallel and Concurrent Programming*, which covers a wide range of programming paradigms, parallel programming systems and languages. I have integrated a simpler form of some of this material into the revised CMPSCI 377 course described above. This course was quite popular: in addition to the 16 students officially enrolled, there were a number of postdocs, faculty members, and auditing grad students in attendance. I plan to offer much of this material again in a seminar next semester.

2 Supervision of Graduate and Undergraduate Students

I enjoy working with and mentoring both graduate and undergraduate students. I typically meet with my students at least once weekly, in one-on-one and project meetings. My office door is always open, and my students frequently drop by to discuss research.

I have graduated one Ph.D. student, Matthew Hertz, who is now a faculty member at Canisius College. I am currently supervising four Ph.D. students: Divya Krishnan, Tongping Liu, Gene Novark, and Ting Yang (co-supervised with Eliot Moss). I encourage my students to do research internships early in their careers, and have sent students every year to places like Microsoft Research, IBM T.J. Watson, Azul Systems, Google, and Intel.

I have supervised or co-supervised a number of synthesis projects, and have served or am currently serving on the doctoral committees of six students (John Cavazos, Abhishek Chandra, Asjad Khan, Naren Sachindran, Bhuvan Uргаonkar, and Ed Walters). I have so far graduated four M.S. students.

The best Computer Science undergraduates at UMass are amazingly good. I have actively incorporated undergraduates in my research, using my operating systems course as a recruiting tool. I have so far published four papers at top venues (USENIX, FAST, SenSys) with undergraduate authors. These students played key roles in the research, and in two cases, the research project was led by the undergrad. This semester, I will have four undergraduate students working with me on a variety of projects. I meet with these students weekly, and each is assigned a graduate student mentor from my group.

Several of the undergraduate students I have worked with have gone on to our “Baystate Scholar” program, a M.S. program reserved for strong UMass CS students. I encourage students who decide to continue to do a Ph.D. to go elsewhere and experience other departments, which is The Right Thing To Do. For example, one of these students (Jim Cipar), who started working with Mark Corner and me as an undergrad and continued as a Master’s student here at UMass, is now in the Ph.D. program at CMU.

I have also co-advised undergraduate senior thesis projects with my colleague Scott Kaplan at nearby Amherst College (Ana Mocanu, Isuru Senevi, and Laura Strickman). While these students are being advised primarily by Prof. Kaplan, we have involved them in our joint research efforts. I also supervised Virginie Guionnet, a visiting undergraduate student from the University of La Rochelle (France).