MICHAEL BERNSTEIN TEACHING STATEMENT

I developed my identity as a teacher through ten years as an amateur comedy writer. Comedy is ruthless work: twenty ideas produce one good joke. Others never find a joke as funny as you do, and watching your punchline fall on a silent or bewildered audience is its own special kind of torture.

Teaching has many parallels to writing comedy. They both require predicting what listeners will think at every moment: what they know, what they suspect, and what will be a surprise. In teaching, however, the climax is not a punchline: it is an idea. I draw several pedagogical techniques from my experiences.

1. *Write meaning into every action*. Ideas, like jokes, need to be set up or else they will not stick. When teaching, every action needs to move students one deliberate step closer to the goal.
2. *Plan the narrative arc.* Students find it difficult to absorb ideas when critical concepts and trivial details are both delivered with equal emphasis. I design micro- and macro-level narrative arcs into my lectures to mark important ideas, I link new insights to old ones, and I use vocal prosody for emphasis.
3. *Focus on creative recombination*. Like comedy inspiration, the source of academic insights is often the combination of known facts in unexpected ways. For example, students might combine known facts into a proof or repurpose other designs for their own project. This is a skill, and it needs to be practiced.

TEACHING EXPERIENCE

I put these techniques to work primarily in the classroom. I have been teaching university students since I joined the staff of Stanford's introductory computer science course as a sophomore. Since then, I have taught areas ranging from human-computer interaction to introductory programming. In addition, I created a design studio course at MIT and taught it for the past three years.

My core teaching experience is in human-computer interaction. I have been fortunate to help teach classes in human-computer interaction at two universities: with Professor Rob Miller at MIT, and with Professor Terry Winograd at Stanford University. Based on these experiences, I draw on multiple approaches to teaching user interface design: Stanford's focus on the design process, and MIT's emphasis on interface engineering. I guided teams in both classes through multi-week design projects.

My most defining teaching experience was creating an MIT class called Interactive Technology Design. I have led and taught the class for the past three years. When I arrived at MIT, there was only one human-computer interaction class in the curriculum. Sensing a gap, I created an intensive two-week design studio course that runs during MIT's winter inter-semester. The class focuses on teaching students the skills of brainstorming and rapid prototyping. For example, I have taken students on needfinding trips to the Boston Public Library, then given them one hour to brainstorm and assemble a prototype solution based on their observations. To build creative recombination skills, I drill students to brainstorm ideas under time pressure and push teams to generate tens or hundreds of ideas before identifying the best one. I believe that these skills are critical to any engineering or design education. Students found the class engaging and useful—alumni now help teach the class, and new students join because of word-of-mouth from previous years. The class has included students from computer science, the Media Lab, aero/astro, and business. It is rated 6.6 / 7 in anonymous MIT evaluations, and other classes have since adapted my classroom materials.

Experience can be enhanced by structured training, so I also enrolled in MIT's graduate student teaching certificate program. The program consisted of seven teaching workshops, readings and homework from the education literature, and one-on-one feedback on a teaching session. I have integrated these techniques into my teaching.

MENTORSHIP

Mentorship is similar to improv comedy: a mentor adjusts moment-to-moment based on responses from students. I have had the pleasure of mentoring six undergraduates and masters students in research. These students have contributed to almost every project I pursued. Three of the students worked with me on summer-long internships, coding and designing alongside me. The other students worked on senior theses and other independent projects, giving me an opportunity to explore more hands-off mentorship styles.

Peer collaboration is critical to success in research. I seek out collaborations with other computer scientists and academics. As a result, I have now co-authored with database researchers and media artists, and I have collaborated with machine learning researchers and social scientists.

COMMUNITY

Peers play an important role in editing and improving ideas, both in academia and comedy. I worked to expand this community of human-computer interaction scholars at MIT and in Boston. I began with a weekly reading group that drew in participants from computer science, aero/astro, the Media Lab, and surrounding universities. As the reading group grew, I took on responsibilities organizing MIT's human-computer interaction seminar series, chairing a local workshop, and starting a Boston-area university consortium for human-computer interaction researchers. Creating a design studio course furthered these goals. I look forward to fostering such a community as a faculty member.

EXAMPLE COURSES

Undergraduate —

1. *Human-computer Interaction:* A practice-oriented course that teaches the fundamentals of human-computer interaction and design. Includes needfinding, brainstorming, prototyping, and evaluation. Students work in teams on a multi-week design and implementation project to exercise course topics.
2. *Information Networks:* Understanding and engineering large information networks like the web. Topics include web architecture, network analysis of authority, centrality, and diffusion, and parallel data frameworks.
3. *Interaction Design Studio:* Teaches ideation and rapid prototyping for interactive technology. Introduces fabrication and prototyping tools, as well as brainstorming, sketching, and critique. Homework consists of structured brainstorming as well as teamwork on design and implementation projects.

Graduate —

1. *Crowd Computing:* Introduction to crowdsourcing and human computation. Focuses on algorithms and techniques for coordinating crowd work, as well as empirical studies of collective intelligence.
2. *Social Computing:* The study of large-scale social systems. Combines theory, social science, and design. Capstone project challenges students to create a successful social site.
3. *Research Topics in Human-Computer Interaction:* Foundational and recent advances in human-computer interaction. Readings from the literature are paired with assignments to recreate or extend well-known work. Topics include input, interaction, end-user programming, social and crowd computing, design, computer-supported cooperative work, ICT4D, and ubiquitous computing.