Susan Elliott Sim
ses@intdexwis.com

Teaching Statement

Teaching is an essential part of increasing human understanding, not only for the students, but also for the teacher. I have always enjoyed teaching because it forces me to organise my knowledge of an area systematically. It also requires me to examine my understanding more critically because students ask simple questions that challenge implicit assumptions. I also consider teaching a necessary part of research because it communicates results and advances the state of the art by educating both future industrial practitioners and researchers.

I begin any educational interaction, formal or informal, by understanding where students are intellectually, developmentally, and socially. I can subsequently use this understanding to plan out how I will lead them to satisfy or exceed learning objectives. I like to challenge students, respectfully, to come out of their comfort zone, and think in new ways. It is not enough to teach students the answers to today's questions, they also need to learn how to find answers for themselves in the future.

I believe that a teacher's most important job is to provide support and encouragement. Consequently, I have put a great deal of time and effort into mentoring students, especially those from underrepresented groups, whether they belonged to my research group or not. The achievement that I am most proud of is the 2nd Annual Mentoring Award from the Associated Graduate Students in the non-tenured category. My students as a group nominated me without my knowledge and the award was highly competitive. It means a lot to me that my students thought enough of me to put in the effort. In addition, their ability to put together a winning case for me indicates that I am teaching them well.

Classroom Experience

While I was at UCI, I have taught fourteen regular courses in Informatics at the graduate and undergraduate level. I have created one new graduate course and re-designed an existing undergraduate course. I will briefly describe these courses before discussing my teaching techniques.

I am the principal instructor and coordinator responsible for Inf111: Software Tools and Methods. I completely redesigned the course after I arrived at UCI. I update the course on a regular basis in order to stay current with industry standard tools. I teach this course at least once per year and often twice. When the course is taught by an instructor, I ensure continuity by providing them with materials and guidance. This course is required for Informatics majors and it is the only Informatics course that is required in the Computer Science and Engineering Major. Consequently, the enrolment tends to be high (50-100 students) and the background of the students tends to heterogeneous; these factors can make the course difficult to teach. Nevertheless, I have received high teaching ratings, with a median score of 8 out of 9.0 over the last three years. Comments from technical managers in industry and students who have graduated are uniformly positive.

Inf201:Research Methodology for Informatics was a new course that I proposed, designed, and taught for the first time. It is now a core course in the curriculum for all tracks in the Informatics graduate program. All students take it in their first year at UCI and through it they receive a foundation for conducting research. The course covers both conceptual topics, such as philosophy of science, and practical skills, such as conducting a literature search and writing an abstract.

Inf 211: Software Engineering, Inf 217:Software Processes and Inf219:Software Environments were courses that were already in the program, but new to me. Each of these courses typically has an enrolment of 15 students. My median score on teaching evaluations in these classes is an 8.0 out of 9. All of the courses are research-oriented, and require students to engage with the literature and produce original

work. The first course in the list is a broad introduction to the research literature in software engineering. I conduct the latter two courses as seminars that conclude with a research project conducted by pairs of students. In the course on software process, I cover topics such as team dynamics, agile development methods, and resource allocation. My approach is to lead the students through competing streams of research, so they can begin to formulate their own research questions. In the course on software environments, I cover not only the mechanics of implementing interactive software tools, but also the psychological, social, and organizational factors that constrain tool design.

Techniques

I use an eclectic toolbox of techniques for teaching and mentoring students. Every quarter, I like to try at least one new thing. In my classes, I have given demonstrations of software tools in class, written computer programs collaboratively with the students, held small group exercises during lecture, led impromptu discussions, organized formal parliamentary debates, assigned hands-on laboratory exercises, administered surprise quizzes, assigned small weekly assignments, and set larger term projects for students. I want to draw attention to three techniques that I use extensively in my teaching.

Question Asking. I require students in my class to ask questions. Sometimes I pause the class to give students the opportunity to ask questions. At other times, I ask the students to write down questions on index cards and have them circulate the cards so their peers can read each other's questions. I use this technique because being able to ask a question requires active engagement with the class material to identifying a knowledge gap. Pausing for questions provides students with a few moments to start organizing their information in their heads, while they are in the classroom and I am present to help clear up misunderstandings or to take the lesson further. I began to use this technique when I found in my research that programmers were remarkably bad at asking questions when they were given new information [1]. I wanted to address this problem by letting students practice asking questions. Key to the success of this technique is establishing a safe space for lack of knowledge. On a recent teaching evaluation, one student wrote, "Very good teacher, family oriented - in the sense that you don't feel intimidated to ask 'stupid' questions. I enjoy how she is not intimidating, but very friendly and respectable."

Scaffolding. In construction, scaffolding is used to allow workers to safely access parts of a building that they cannot otherwise reach. In developmental psychology, children often use scaffolding to help them solve problems that they otherwise could not. For example, young children often talk to themselves out loud when figuring out a puzzle; later this self-talk becomes silent and internalized. I use the concept of scaffolding to ensure that students who enter a course with highly divergent backgrounds and skill levels have the same opportunities to master the material. In the undergraduate course on software tools and methods, I use scaffolding extensively. When starting a new unit, I begin by describing a difficulty in software development, which suggests a niche for the tool we are about to study. Next, I give a demo of the tool and take suggestions from students on how I should use it. I then move on to how the tool works conceptually. Students build on this by completing a supervised hands-on laboratory exercise that requires them to use the tool on a simple source code example. By this point, I will have provided a level playing field for students to undertake their individual assignments on a larger source code example. At the graduate level, I have used scaffolding in the software processes course. We start the quarter with readings and lectures, a format that they are familiar with. We then move into a series of formal debates, where each side is defined by a set of 2-4 papers. The "government" has to argue in favor of a resolution that takes an extreme position. The "opposition" argues against the resolution. There are strict time limits on each statement by the speakers and their rebuttals. This format models how to form arguments about positions in research. By the end of several weeks of debate, the students realize the interesting opportunities for research lie in the middle ground. For example, rather than advocating lots of

documentation for software (or no documentation for software), they understand that the real question is how much documentation is enough.

Use of Technology. The current generation of students are at ease with technology in its myriad forms. I try to use this familiarity and ubiquity to my advantage as a teacher. For many years, I have been holding electronic office hours, usually in the evening, when I am available on instant messenger to answer questions. Students tend to have many obligations and responsibilities, and electronic office hours provide one more opportunity for contact with me. They especially appreciate the chat rooms that I hold the day before a test or exam. I have used the web site "Poll Anywhere" to conduct real time surveys during lecture. Students can use their cell phones to text comments, questions, or answers to multiple choice questions to Poll Anywhere and these appear immediately on the web site, which is projected at the front of the class. Recently, I have been experimenting with creating audio and video podcasts of lectures. Making these available did not decrease attendance and students found the podcasts helpful for refreshing their memories when studying. The innovation that I am most proud of is crowdsourcing of tests and exams. About two weeks before a test, I set up a web site where students can submit questions, comment on each other's questions, suggest improvements, and vote on their favorite questions. If there are sufficient high quality questions, the test may consist of questions taken entirely from the site. This technique has been highly effective, because students are engaged in their own education and evaluation, and it gets them asking questions (an activity mentioned above). Students have reported that crowdsourcing has helped them with retention of material, especially when one of their questions gets used on the test.

Research Mentoring

In addition to classroom teaching, I have mentored many students at all levels as part of my research group. I have supervised thirty-one student-quarters of undergraduate research and thirty-nine student-quarters of graduate research. I have also supervised four full-time student-summers of undergraduate research, and eleven full-time student-summers of graduate research. I have supervised six Ph.D. students, served on the examining committee for a further six Ph.D. students. I graduated my first Ph.D. in this year. I have supervised three Master's thesis students (graduating two), and served on the examining committee for four others.

I have a group meeting and one-on-one meetings with each student every week. During the group meeting, each student reports on their progress, describes any difficulties that they encountered, and seeks input from others. Sometimes, we have practice talks or brainstorming sessions. The purpose of the group meeting is to provide students with exposure to research in progress, build community, and allow more senior students to serve as role models for the more junior ones. There are so many frustrating and mysterious aspects of doing research that one of the best ways to learn how to do it is through legitimate peripheral participation. By building ties between the students, they can turn to each other in times of difficulty. During the individual meetings, I can give students personalized attention and provide advice, assistance with writing, or a shoulder to cry on, as appropriate.

References

[1] Ratanotayanon, S. and Sim, S.E., "When Programmers Don't Ask," in *Second International Workshop on Supporting Knowledge Collaboration in Software Development*, Tokyo, Japan, 2006.