Teaching Philosophy Statement
Danny Dig

My goal is to inspire those around me to find their vocation and perfect their talents. No other career can help me accomplish this goal like an academic career.

My teaching philosophy has developed as I taught software development classes ranging from high school to graduate level, from small undergraduate courses (CS498DD) to large undergraduate courses (CS427/428 at Illinois, CS361 at OSU, OOAD at CU), with traditional audiences as well as over 800 professional programmers at summer schools (e.g., in Portland on Oct 2014), at large companies such as Boeing, and conference tutorials. I believe that *students' motivation is the most important predictor of what they will learn*. The professor can amplify students' motivation by (i) showing the value/utility of the class and (ii) convincing students that the classroom environment is conducive to success. I continuously learn about methods of teaching, and I hope to improve my teaching effectiveness.

I want to be a thinking partner and contribute to the success of SoC’s pioneering Software Development degree and the Master of Software Development program. I am attracted by President Randall’s vision that expresses very well the heart of the U: re-imagine a new experience for students that inspires them with a renewed sense of optimism and impact. I would be honored to serve together in a department that inspired Utah students like Alan Kay to invent ground-breaking computing technologies.

Goals and Methods

1. I believe **the deepest learning occurs in an active learning environment.** Such an environment changes students: they feel ownership for their own education and feel more confident. At Illinois I taught two of the largest undergraduate classes: Software Engineering I (CS427) and Software Engineering II (CS428) with 150 students each. I know that in a large classroom there are many things that can distract students. It is important that every moment in class provides value. Otherwise, students lose interest and become passive consumers. For example, the CS427/428 class had 20% attendance in the past 10 years. To change this, I have tried four new techniques.

(i) I used **iclickers,** a personal response system designed to be used in classroom. My clicker questions checked knowledge but also enabled students to practice critical thinking. The clickers allowed me to constantly poll the students during lecture and adapt my delivery dynamically, based on the topics they had trouble understanding. Unlike passive quizzes where students do not necessarily learn from their own mistakes, active clicker polls create a positive feedback loop. As soon as the students answer my clicker question, I show the response statistics. I celebrate with them when most of them select the correct choice. When I see that many of them pick the wrong choice, I probe deeper. I ask them to reason aloud why they picked a certain choice. They practice critical thinking because they must argue their answer with their peers. This always provokes animated dialogue in the classroom and most of them end up discovering the right answer. Thus, they see value in coming to class and the confidence in their abilities increases.

(ii) I designed **group activities**, where groups of four students work on a small problem during class sessions. The TAs and I are circulating around the classroom to stimulate deeper discussions, and to offer them customized individual attention which is rare to get in a large class. I assign problems that are relevant to their class project. Thus, they see how the lectures and project reinforce each another.

iii) Using video-conferencing, I **interviewed several industry leaders** (e.g., Jack Dorsey - Twitter founder), book authors, lead software architects from well-known products, and famous alumni (e.g., FarmVille creators). The interviewees describe how the class topics apply to the real world, thus students see the utility of the class. Students pose questions to their heroes, thus students feel special. For example, when teaching about the open-source development, I interviewed Robert O'Callahan, one of the lead developers of the Mozilla Firefox. Students drilled down into the relationship with Google, a competitor building the Chrome browser, yet still one of the largest funding sources for Firefox. I was impressed and proud that students asked thoughtful, deep questions. Students were very articulate because they had plenty of opportunities to practice through the iclicker and group activities. In addition, the industry leaders inspire the students to dream of big successes.

(iv) In CS it is easy to lecture for 75 minutes continuously, but deep learning occurs when students themselves wrestle with the content. To **prevent information overload**, I divided my lectures into **15-minute segments** followed by an interactive activity (e.g., clicker poll or group activity). It takes more effort to prepare to teach this way, but these segments are a good reminder for me to let students wrestle with the content. I chose not to be driven by the amount of content, but by the experiences that students need to learn deeply.

These four techniques increased the class attendance from 20% in previous years to 80%. Also, student retention from CS427 into the follow up CS428 increased from 75% to 93%. Students enjoyed the class much more and felt more comfortable. They were more engaged, talked more with each other, and asked insightful questions.

2. I believe my role as a teacher is to **inspire students to find their own voice.** When students find their unique talents and how they can contribute to the society, they are happy during their academic life and feel satisfied during the professional life. For example, one summer I was mentoring one undergrad intern who struggled at the beginning of the internship. Once I helped him tap into his passion for functional programming, he was able to come up with a novel research idea, mature it for submission to a top conference, and contribute it to the official release of the NetBeans development environment used by millions of developers. I was gratified to be an enabler and a maximizer of his talents.

Some undergrads are excellent candidates for graduate school, though they do not think about themselves that way. To inspire them to find their own passion for research, I have mentored 19 undergrads during summer internships; 9 of them are now in graduate school and 2 more are applying. I organized four faculty-grad-undergrad panels to encourage CS undergraduates to continue with graduate school. I enjoyed organizing such events at other departments on campus (e.g., ITI summer internships at Illinois), other schools (e.g., Rose-Hulman), and conferences (e.g., the 2-day Inspirations event at SPLASH'14 conference, attracting **30% undergrad females**).

Not every student will continue with graduate-level research. To inspire those students who choose industrial careers, I require they gain experience with large pieces of high quality open-source code. This way they can see the beauty of well-crafted code and emulate it. When teaching CS361/427/CS498DD, rather than asking students to design toy projects, I have them **contribute to large open-source projects**. Real open-source projects are relevant to students' lives and these projects increase students' motivation. Students learn how to make valuable changes in large projects, thus they learn the essential skills for being successful in today's software engineering jobs. When teaching SE, my students worked with real-world customers (e.g., building the library chat system used now at UIUC library), and **deploy into Apple and Google app store**.

I also organize/moderate panels with industry leaders to help students prepare for jobs. In the most recent panel I organized, representatives from AKQA, Columbia Sportswear, Hewlett-Packard, Tripwire and Vadio answered questions about what to expect when entering the workforce and how best to prepare. The event drew 250 students. The event was well received. One student who attended the event said: "*Being able to get firsthand advice from professionals in the industry is invaluable to me. It not only gives me ideas of what I would like my goals to be beyond college but it helps me to be pro-active about my future right now*."

3. As a teacher, I strive to educate students to **grasp the principles**. Long after students have forgotten the trivia, they can remember and apply the principles, even in different contexts.

Principles become more memorable when I support them by real-world stories, experiences, and/or case studies. In addition, the stories enforce the relevance of the class to the real-world. To relate the principles to students, I ask them to share how principles can be applied to problems they faced in the class project. The homework assignments that I designed require students to apply the principles to very different contexts than the ones presented in class. The questions I ask in exams are short essays that require students to think creatively and apply the principles to different problems. These questions take more time to write and grade (there are many correct answers, and even more wrong answers), but they are a worthy investment: they engage the student to learn new things even from exam questions.

4. I want to help my students improve their **interpersonal skills**: to be successful in a **team**, to **lead** and **influence**. The complexity and scale of the problems in computer science today outweigh the capabilities of one individual. I want to teach my students how they can contribute their own unique talents to enrich the team.

To promote collaboration, I require that all programming assignments be done in a pair-programming style: the partners alternate between playing the "Driver" and the "Navigator" roles. The research [McDowell et al. CACM Vol 49(8), SIGCSE Vol 34(2), Williams and Kessler – Conf. Soft. Eng. Education'00] shows that pair programming is effective in computer science education.

In the Software Engineering course I put students in 8-member projects, a sub-optimal size (too large), but this teaches them how to handle the communication overhead and resolve conflicts.

At the end of each team project, I encourage each team to conduct a project retrospective using several exercises from the seminal book *Project Retrospectives* by Norman Kerth. The retrospectives show students how much more effective they are in a team and it gives them a sense of accomplishment and success.

My Software Engineering undergrad classes (CS427, CS428 at Illinois, CS361 at OSU, CS 4448 at CU) are regarded by undergrads as some of the most work-intensive classes. However, I receive dozens of emails from students (typically a couple of years after they graduate) telling me how useful my class was for the transition to professional software engineering. Below are just a few samples:

"*Being one of my favorite mentors, I just wanted to write to share some exciting news with you: I am thrilled and honored to be one of the 5 speakers representing Salesforce.com to speak at the Grace Hopper Celebration of Women in Computing this October …. Thanks so much for your help and guidance in my software engineering project which greatly helped (and is still helping now) in every walk of my career.* " [class of 2012, now at Salesforce]

"*The biggest thing that the Software Engineering class did for me was to give me a sense of the Real World (tm). Much of the work was thought-provoking and exciting. Software Engineering and the Senior Projects both gave a sense of how to work in teams, how to engineer solutions to specific technical challenges. I think having a balance of deep technical issues (most of the classwork) and real-life challenges and approaches is critical to being successful in the field (software development) MOST of us ended up*." [software architect, Chicago Mercantile Exchange]

"*I've been working as Quality Engineer for the past 6 months now, and I must say: the lessons learned in Software Engineering I and II have been invaluable to say the least. … The knowledge I gained on development methodologies in particular have made the transition into the working world extremely easy*." [Alum 2013, Startup in the Bay Area]

Teaching Interests

My extensive experience as a CS faculty for on-campus, distance learning, and industrial courses gives me the confidence that I can teach both introductory and advanced CS topics. I am already prepared to teach several existing courses in Software Engineering and Programming Languages (CS – 3500, 3505, 3550, 4000, 4011, 4500, 4230, 4530, 4550, 5470, 5510, 6010, 6011, 6015, 6018, 6019, 6800), Introductory courses (CS - 1030, 1400, 1410, 1420, 2100), and develop new courses. For example, at CU Boulder I am developing a new grad course, Software Development for IoT Systems, where students work on projects with campus partners. One team works with the CU's Chief Data Officer to develop a "sense of belonging" community sensor, a novel IoT service that our Office of Data Analytics plans to use across the whole campus. Unlike qualitative surveys that suffer from selection bias and single point-in-time data, the new IoT community sensor will have more data points and contextual data and will help the campus implement data-driven approaches to improve student success and retention. I am also interested to develop other grad courses: Software Design, Software Evolution, Program Analysis and Transformation, thus bringing grad-level SE classes to Utah students so they have equal opportunities with students in the MSD program. I am most excited to develop a new course on how to retrofit ML libraries in large legacy software systems and existing messy datasets, as this is the reality that students will face in their jobs.

I am impressed about the pioneering spirit in education at Utah's SoC and the desire to align with the Utah Legislator to create software development degrees and certificates. I would like to be a thinking partner and contribute to (i) the growth of the Software Development B.S degree that launched in Fall'22, (ii) the Master of Software Development program led by Matt Flatt, with whom I've had several conversations on this topic, and (iii) Stackable Certificates in existing areas like Software Development, and launching new areas, e.g., IoT. I am excited about **adapting education to Gen Zers, whose learning style is Just-in-Time, as opposed to the Just-in-Case style of my generation**.

I also plan to continue sessions with junior and senior undergrads to **inspire** them to apply for graduate school. For graduate students I will continue to develop seminars like Practice for Academic Job Talks, where they get feedback on their job talks from a broader CS audience – one like that they will face in a real job talk.

To constantly sharpen my teaching skills, I worked with scholars from the Center for Teaching Excellence at Illinois, OSU, and CU. I will continue to seek support from mentors in my goal of becoming an effective, responsive teacher.

Conclusions

I am aligned with the strategic vision of President Randall for the U to reimagine how to Inspire, Innovate, and Impact. I would be honored to join forces with others at the U to revolutionize the experience of students and create a sense of optimism, belonging, and contribution to the society. I am dedicated to produce well rounded software engineers with knowledge and skills in ethics, leadership, business, and communication. This is well aligned with CoE's CLEAR program. I made a priority to invest myself in people, not products. I will continue to inspire students, engage them in active learning, help them learn the principles not the trivia, and develop their interpersonal skills. More than anything, I want to help the U **raise tomorrow's technology leaders**. Together we can provide tremendous value to all Utahns.