Teaching philosophy and experiences

I strive not merely to deliver knowledge but to build students’ expertise and critical thought. To prepare myself to be an effective educator, I studied pedagogical techniques in the *Foundations of Engineering Education* course offered by VT’s Engineering Education department. I learned that motivation is critical to learning [1], and that active and inductive learning experiences can encourage deeper understanding than traditional lectures [2,3]. I have found these principles effective both when I educate engineers and when I train researchers. And my students agree, by giving me strong ratings on student evaluations and by winning research awards.

In the classroom, I have been an instructor of record for two undergraduate courses:

- **CS 1064** Introduction to Programming (75 students, non-majors, Spring 2019)
- **CS 3114** Algorithms and Data Structures (70 students, CS majors, Fall 2019)

and a teaching assistant for three courses:

- **CS 3604** Professionalism in Computing (CS majors, Fall 2015)
- **CS/ECE 5510** Multiprocessor Programming (Grad-level, Fall 2017)
- **ENGE 1644** Global STEM Practice (general engineering majors, Spring 2017 and Spring 2018)

In the laboratory, I have mentored 10 students in computer science research.

Teaching in the classroom

Keeping students motivated is key to successful learning outcomes, so I apply the eMpowerment-Usefulness-Success-Interest-Caring (MUSIC) model of motivation in my teaching [1]. Let me share some examples.

To increase eMpowerment, I make my classroom interactive, interspersing instruction with think-pair-share exercises to engage my students and learn their misconceptions. I also eMpower my students by inviting monthly feedback to assess the effectiveness of my teaching and make micro-adjustments in the course. To help students see the Usefulness of the material, I begin many lectures with a relevant story from my years as an engineer at IBM. To show students that I Care, I learn the names. And to increase Interest, I incorporate examples that appeal to a broad audience with diverse life experiences.

With a longer class period (CS 1064 was 75 minutes/session), I facilitate inductive learning by splitting the in-class time between lecture and in-class laboratory exercises. This division gives my students a chance to cement their understanding of the deductive lecture presentation through an active learning activity with plenty of time for questions. I developed a variety of exercises, from pair programming to debugging to comparing algorithm runtimes. Students said this was effective: “He makes you think critically” and “was really good about learning his students’ names...he explained things differently depending on how we had interacted on past subjects”.

When I have shorter classes (CS 3114 was 50 minutes/session), I rely more heavily on think-pair-share [3] to efficiently give students a chance to try topics out for themselves. My purpose with think-pair-share is to invite my students into active learning, rather than just listening to me lecture. I commonly begin a topic with an opening question and the instruction to “Ponder!”, and then roam the room for a few minutes to coach students through their reasoning. I also introduce visualizations and activities to help students build mental models about the material. For example, to illustrate binsort I sorted a deck of cards in linear time, then had volunteers fling cards into the air to illustrate bogosort. In one of my feedback surveys, a student shared that “I really liked the visual demos we did, that will help me remember better” – precisely my goal!

Mentoring in the laboratory

While a graduate student at Virginia Tech, I have helped train ten researchers: one PhD student, three master’s students, four undergraduate researchers, and two high school students. Many of these students contributed significantly to research projects, and three of my mentees received awards for their research.

One of my mentees was James Donohue M.Sc., University of Bradford, UK. James learned of my research into regexes and asked me to serve as his de facto master’s thesis advisor for his final year of graduate school. We exchanged 150 emails, in which I gave feedback on his research ideas, helped him scope his project, advised on his experimental design, and coached him in writing up his thesis. Another of my mentees, Louis Michael, extended James’s findings, and the three of us collaborated on an award-winning paper at ASE’19.

In addition to James and Louis, I have mentored eight other students and published with three of them.

As I do in the classroom, I structure my mentoring using active learning. My mentees developed as researchers by
conducting research, not by hearing me lecture about it. When I work with a mentee, I typically “give them their head”, even when they pursue a risky course of action. When analyses work, we celebrate; when experiments fail, we discuss what went wrong, why, and what to try next. Through this process I identify the strengths and weaknesses of my mentees and equip them with the resources they need to improve. For example, where needed, I share resources on software design or technical writing and focus my coaching in these areas.

Teaching amidst diversity

Diversity is a strength. A diverse engineering team can design and build with all of their consumers in mind, and a diverse research team can consider a wider range of problems and phenomena. It is to the detriment of the computing field and the world we serve that computer science is dominated by white and Asian men. By encouraging and supporting diverse students through my instruction and research mentoring, I hope to shape the next generations of computer scientists to more closely resemble society.

For example, as an instructor I try to create a friendly learning environment that welcomes differences. I refer to my students as “folks”, not “guys”, and I design exercises that will interest a range of students including underrepresented minorities. Modifying my behaviors in these ways costs me nothing, but warming the “chilly climate” of engineering can help with the engagement and retention of these students [4].

Teaching plans

My research, coursework, and professional experience have prepared me to teach a range of undergraduate courses, such as algorithms, automata theory, operating systems, ethics, and capstone courses. I can teach graduate-level coursework in systems, security, and software engineering. As the organizer of the VT Systems Reading Group since 2017, I have spent years leading a seminar-style “class” in these areas.

In addition to teaching existing courses, I would be interested in developing a course in global software engineering. At IBM I worked with coworkers in different countries and from different cultures, which introduced additional complexity into the engineering process. I share these experiences with first-year students in the Global STEM Practice course (ENGE 1644). I would like to go further by developing a course in which teams of students from international universities build software together. This course would prepare students for the realities of computing in industry and give them a more global outlook on the role that engineers play in human society.

I am excited to build a research laboratory. I plan to offer my students weekly one-on-one training meetings, weekly lab meetings, and monthly social gatherings. These meetings will give my students personalized training, support from the larger team, and a sense of community. I will involve both graduate and undergraduate students, to increase the lab’s bandwidth, offer mature graduate students mentoring opportunities, and let undergraduates test whether graduate school would be a good fit.

References