Austin Henley — Teaching Statement

austinhenley@microsoft

The most invigorating moments in my career are when I have the opportunity to **teach**. This has been true while working as a program manager at Microsoft, as a tenure-track professor at the University of Tennessee, while writing for my technical blog that has been viewed over 3,000,000 times, and even as a PhD student when I mentored students and taught my first course. I want *more* of these opportunities.

My experience in the classroom involved teaching undergraduate and graduate software engineering courses. They focused on applying industry-inspired development processes to build graphical user interfaces with modern technology (e.g., Electron apps made with Node.js) in a group setting. A primary goal was to give students a portfolio that they can demonstrate at job interviews as well as give them the confidence to succeed once they do begin their career. Through these formative years in the classroom, I was able to establish my personal approach to teaching, find what is effective for me, and reflect on what I can improve on to be a more successful instructor.

Classroom Methods

To incorporate evidence-based teaching practices, I invest considerable energy into educational psychology research. As a graduate student I took numerous graduate psychology courses that introduced me to concepts such as active learning, cognitive load theory, self-explanation, and the spacing effect. I operationalize these techniques by (1) breaking lectures up into 20-minute micro lectures, (2) involving a hands-on activity nearly every lecture, (3) exposing students to course material a minimum of three times that is deliberately spaced out across two weeks, and (4) providing opportunities of reflection and peer teaching.

In fact, I have found the peer teaching component to be incredibly effective at improving engagement. The contributing student pedagogy emphasizes students taking ownership in the course by contributing to the course materials and teaching one another while deemphasizing the traditional one-way lecture. I even give the option for student groups to give their own micro-lectures and in-class activities. I took this to an extreme for one course where we, the students and myself, investigated the CPython implementation while sharing everything we found during class sessions, and I took part as a student and a facilitator. My takeaway from that semester is that students are far more open and engaged when they view me as a collaborator rather than an evaluator and authority.

Lessons Learned

After teaching my second year of our undergraduate software engineering course, I received my career lowest score on student evaluations, a 3.9/5.0, compared to a 4.8/5.0 the year prior. I took the feedback personally. My evaluation scores and feedback did go back to far above the university average the next semester, but it took me considerable time to make peace with the negative feedback and to learn from it.

What happened? My first cohort of students gave me glowing feedback that they *loved* the in-class activities and the project, and they suggested that I reduce the long lectures and exams to

make room for more activities. So, I did. But what I did not consider is that each cohort of students has different needs. I immediately noticed that my approach was not working for the second cohort: students resisted engaging in the projects, they were spaced out in a far larger auditorium, the time of day had everyone feeling tired, and the sense of community did not come effortlessly this time. I made the mistake of attempting to push through with my semester plan without adjusting.

This scenario was further complicated by my teaching assistant management. Other professors told me I was spending too much time on my courses, so I relied on my teaching assistants more this semester. But once again, I leaned too far into addressing the feedback. Because I did not properly support my teaching assistants, the students felt as though they were given too harsh of grades, not enough qualitative feedback, and inconsistent instructions.

What did I learn from this? I have to adapt to the students and every semester will be different. It is great to react to direct feedback, but I should investigate the core problem, iterate quickly to try multiple solutions, and not be afraid to change or undo a change. Moreover, I learned teaching assistants are a *phenomenal* resource when given a constructive environment. For example, now I prefer to let TAs work collaboratively, rather than splitting them to portions of the class, and I establish a tight feedback loop between them and myself so we can solve problems together and they can see my decision making.

Beyond the Classroom

There are far, far more opportunities to teach than only in a formal classroom setting. In fact, in the first year of being a professor, I realized that teaching one course per semester was not enough. I then started my technical blog about software engineering, academia, and product design that has been viewed over 3,000,000 times over the last 4 years. Notable posts include *"Challenging projects every programmer should try"*, *"Lessons from my PhD"*, and *"Let's make a Teeny Tiny compiler"*. I also regularly volunteered to teach independent study courses where I would simulate a startup environment for small groups of students to identify a problem, a market, and iterate on solutions. These sessions were so exciting that I often would get involved in the building of an app with the students, which inspired many of my blog posts.

Although writing blog posts and tutorials scales well for a large audience, it does not provide the same feedback loop and human interaction as a classroom. After joining Microsoft, I co-organized a cross-organizational storytelling initiative to foster technical presentation skills, and then I created a storytelling mentorship ring to work closely with a small group of early-career researchers and engineers on a regular basis.

These teaching pursuits are just the tip of the iceberg though. I am continuing to look for ways that I can have more impact on more people, perhaps through an online interactive "book" or through creative videos to engage people who never intended to learn computer science. But these efforts are not mutually exclusive with teaching a traditional course, and I cannot wait to be back in the classroom.

Course interests: *software engineering, GUI programming, developer tools, empirical methods, web development, mobile app development*