Teaching Statement

In "My Teacher is an Alien," a science fiction series that transformed my perspective when I read it as a child, aliens evaluate the human species for destruction as a result of evils that occur on Earth. By having an alien pose as a teacher, the aliens hope to collect data to inform their decision. The hero saves the planet by arguing that the aliens should send *more teachers* to help Earth prepare for membership in the interplanetary council. I believe in the power of teaching to change how people think; to shape the civilization the next generation will form; to empower people from many different backgrounds and perspectives to work together to achieve their goals and make the world a better place.

The human progress that comes from teaching has motivated me to seek many opportunities to teach. As an undergraduate at CMU, I proposed and taught a university figure skating course. As a software engineer at Apple, I recruited and mentored student interns. As a Ph.D. student at CMU, I was a TA for software engineering and constructive logic courses and also recruited and mentored student researchers. Recently, at UMD, I created and taught a course on *usability of programming languages*.

Teaching Philosophy. The power of *abstraction,* the benefits of *efficiency*, and the centrality of *end-users* drive requirements when writing many kinds of software. Students who take my courses will be able to create appropriate abstractions, reason about their performance implications, and consider what implications those abstractions have on their users. For example, when teaching software architecture, I asked students to describe the trade-offs of their design choices, and then gave feedback that helped them consider additional implications of their choices. In software engineering courses, I teach and evaluate teamwork explicitly. I use peer reviews to assess team dynamics since course staff cannot observe all the team meetings. In one case, I attended selected team meetings because one student observed that the other team members were repeatedly interrupting her; while keeping the specific problem confidential, I helped the team establish more effective, equitable collaboration.

Research has shown that students learn best from *effortful practice* [1], in which learners practice challenging tasks intensely. Exercises that have real-world relevance can be particularly motivating. Therefore, I use active learning activities with compelling contexts to reinforce the material. In a lesson on software requirements in a software engineering course, rather than merely discussing methods of requirement elicitation, I asked the students to elicit requirements from me for small computers that SCUBA divers typically carry. To create a software engineering course project, I built a relationship with the campus athletic facilities director and her staff, who wanted software to manage logistics of group exercise lessons. Then, students interacted directly with stakeholders to understand their needs. This approach provided students with effortful practice in an interesting, real-world context.

I use *backward design* [2], which suggests basing decisions about instructional strategies and assessments on learning goals. This approach leads to activities and assessments that focus on the learning objectives, resulting in meaningful evaluations of success. In one course, for example, I graded students according to the depth of insight they demonstrated in their arguments for particular software engineering decisions and only in certain cases according to the quality of the artifacts they produced.

I believe that the best teaching requires continual improvement and reflection. Through the Future Faculty program at CMU, I participated in seminars and received useful feedback on my teaching. I also took a course on CS pedagogy offered by teaching faculty members Charlie Garrod and Michael Hilton. A new research endeavor I am leading concerns how to teach strongly-typed languages most effectively.

Teaching Interests. I am interested in teaching courses on software engineering, programming languages, and human-computer interaction. Leveraging my industrial software engineering experience, I am especially interested in interdisciplinary collaboration that enables students to explore relationships among fields. For example, I would like to create a course in which CS students collaborate with students from other disciplines, such as design and business, so that students learn both to work in heterogeneous

teams as well as to appreciate the value of other ways of thinking. I am also interested in teaching students about software at *scale* — work that rests on lasting foundations of design and debugging techniques but which most students do not learn until they leave academia. I am also excited to teach traditional technical courses, such as compilers and programming languages.

I do not have teaching obligations as a postdoctoral fellow, but I created and taught an online course on *usability of programming languages* to 21 students in spring 2021. To enable the students to practice the material in a context they found compelling, I included a project in which students conducted a usability study on a language design question of their choice. I also leveraged my industry experience to show how language design choices affect practitioners. One student wrote: "I think I gained an interesting perspective from the class regarding programmers, programming languages, and usability research — which was what I was hoping to get out of the class."

Mentoring. I particularly enjoy mentoring students on an individual basis. I have supervised a total of 23 students on various research projects, four of whom later joined Ph.D. programs. I also interviewed, hired, and supervised interns while I was a software engineer at Apple, and I currently supervise a software engineer who works on the Obsidian programming language. I find mentoring particularly rewarding because I can make a significant contribution to people's growth and help them experience the joy of research. I place special emphasis on matching students with projects that they find motivating according to their own interests and goals. When students do not have specific projects in mind, I offer choices they might find interesting, explaining the impact that they could have by working on each project.

Diversity. The lack of diversity in computer science is unjust and threatens our ability to conduct impactful research. Diversity has unfortunately tended to decrease at successive stages; for example, even diverse undergraduate populations yield less-diverse graduate student cohorts. I work to address this by mentoring earlier-stage students, who might not otherwise get involved in research. I have participated both as an organizer and as a mentor in a software engineering Research Experience for Undergraduates (REU) program, which invited undergraduates, particularly ones from under-represented groups, to conduct research over the summer. In the admissions process, I proposed students to recommend to potential research supervisors from among the applicants. I mentored eight different students through this program, four of whom were from under-represented groups, over five summers. Three of these REU students (two from under-represented groups) later joined Ph.D programs. Their work was published in several papers, and two of the students won second place in SPLASH student research competitions. Recently, I mentored four Maryland high school students who were interested in research. Working with earlier-stage students enabled me to recruit a gender-balanced and socioeconomically-diverse group of research assistants, which is a step toward increasing CS diversity in the years to come.

Supporting diversity in teaching goes beyond offering problems and examples that appeal to diverse experiences and interests. It means considering that some students may have jobs or family responsibilities; accepting that *equal* rules are frequently not *equitable*, so special circumstances require flexibility and care; and promoting a *growth mindset* so that all students feel empowered to succeed.

Conclusion. I seek to place the material I teach in context; convey the relevance of *abstraction*, *efficiency*, and *users* in students' work; and become constructive team members and leaders. I convey these ideas through *effortful practice*, which enables students to master the material so that their mastery transcends time, while also ensuring that all students have a positive learning experience.

Teaching is perhaps the most important way we inspire and empower the next generation to be the best they can be. As a teacher, I can make seemingly alien subjects familiar so that future generations can wisely and safely harness and extend computation. I look forward to sharing the beauty and power of computer science with those who will have the power to change the world.

[1] Peter C. Brown, Henry L. Roediger III, and Mark A. McDaniel. Make it Stick: The Science of Successful Learning. The Belknap Press of Harvard University Press. 2014.

[2] Grant P. Wiggins and Jay McTighe. Understanding by Design. Association for Supervision and Curriculum Development. 2005.