Owolabi Legunsen

Teaching Statement

I am very excited by the opportunity to become a teacher and advisor who fosters learning environments in which my students thrive, regardless of their academic backgrounds and chosen career paths. During my PhD, I had the privilege to mentor the research of 22 students, for one of whom I informally co-supervised the MSc thesis. Additionally, I was a teaching assistant in five different classes in two universities (UIUC and UTD), taught by five different professors, with student enrollment in each class ranging from 16 to 204 students. I gave seven guest lectures which gave me the opportunity to practice my teaching philosophy. The classes that I taught were at different levels of instruction, from undergraduate to PhD level, and spanned different topics in software engineering, programming languages, and applied formal methods. I would like to teach classes on these topics, and more, in the future.

Class Design Approach

Strong emphasis on team-oriented hands-on projects. A key component of my classes is the team project, where my students get together to practice what they learn in class, and learn how to work within a team. Team sizes range from two to six, and teams are either formed by the students or assigned. I found team projects to be more effective for student learning when (1) deliverables are in short sprints which closely match the topics being taught in class, and (2) I meet frequently with each group so I can gauge who is participating and what students may be struggling with. A key challenge with team projects is ensuring that the needs of students who may be struggling are not obscured by the performance of stronger team mates. I try to overcome this challenge by ensuring that each student leads the development of at least one project component, and presents at least one component where they were not the lead. I applied team-oriented teaching as a teaching assistant in three classes of different sizes at both UIUC (204 students) and UTD (60 students in one class and 16 students in another).

Balancing Ground-up development with maintenance of existing software. Traditional software engineering courses tend to focus on teaching students how to develop software from scratch. However, in practice, most software engineers spend majority of their time maintaining and extending existing software. Therefore, I strive to expose my students to the challenges of, and effective techniques for *both* developing and maintaining software. For example, the aforementioned 204-student class at UIUC was focused on teaching students how to understand and add functionality to large, existing software. We taught students not just how to understand relevant parts of code in a short time, but also how to elicit/analyze requirements for, and design, develop, integrate, test and deploy the new features. I applied the same principles as a course-project mentor for 13 graduate students in two advanced software engineering courses at UIUC.

Lecturing Approach

During lectures, I strive to maximize student engagement and learning through copious use of simplifying examples, and by relating taught concepts to current industrial and research challenges.

Learning from examples. I gave seven guest lectures during my PhD, in which I found simplifying examples to be very instrumental in conveying the essence of complex concepts. I start my lectures from very basic examples and gradually add complexity. Starting with basic examples can ease introduction to students with no prior exposure to the topic, but risks being boring to more advanced students. Therefore, in each example, I would omit some detail, e.g., a corner case. Then I put up a "quiz" slide and pause for a few minutes to challenge students to figure out what was missing from, or wrong with the example. Lively discussions often ensued which helped me gauge progress and identify any concepts that students may be having trouble understanding.

Relating taught concepts to research and practice. In my lectures I try to show students how taught concepts relate to pressing problems in industry and research. For example, whenever I taught on regression test selection (RTS), I noticed significant increase in students' questions and attention after I mentioned that RTS can help in practice because regression testing at Microsoft, Google and Facebook is now too expensive even for those companies.

Advising Approach

My passion for advising led me to mentor 22 students during my PhD. In my research area, I mentored undergraduate (three), Master's (four) and junior doctoral (two) students. Incidentally, *all four Master's students that I mentored within my research area are women.* I also mentored 13 Master's students in their individual semester-long, research-focused advanced software engineering course project. I have been told that I am very patient with my advisees in terms of helping them find their footing in research. My approach is to initially allow them to explore their own interests within the several topics that I am typically working on at any point in time. Once we find a topic of mutual interest, I then scope the work such that they can start making small but tangible progress immediately, while also building skills for more advanced future steps. I make myself available to my advisees; they always know where to find me physically or online as soon as they run into a problem.

My collaborations with junior students have been quite fulfilling and productive. I have had the fortune of watching each of them grow from figuring out what to do for research in the beginning, to taking ownership and driving their own work in later stages. I also submitted seven papers to top software engineering venues with seven of the nine students that I worked within my advisor's research group—two of three undergrads, and five of six graduate students that I mentored submitted at least one paper with me. Five of these papers are already published, including one paper with two of these students which won the ACM Distinguished Paper at ASE 2016. I also informally co-advised one Master's thesis. One Master's student that I mentored has been accepted to the PhD in CS program at TU Munich, and an undergraduate student that I mentor is currently applying to grad school. The other Master's students that I mentored accepted jobs in industry, and I could not be more proud of their accomplishments.

Teaching Interests

My prior teaching experience covered a wide range of subjects at different levels of instruction (from undergraduate to PhD). I was privileged to work with and learn from a number of amazing professors at UTD (five professors) and UIUC (three professors). I believe that I am qualified to teach undergraduate and graduate classes in software engineering, including classes on software testing, object-oriented programming, requirements engineering, and software architectural design. I am able to teach both the classical concepts in these courses, plus their adaptation to modern-day software engineering. I can also teach undergraduate programming languages, databases, program analysis, program evolution, and formal methods; I taught or excelled in many of these classes in the past.

At the graduate level, I would like to organize discussion seminars on recent research and industrial trends in software engineering, formal methods, and program analysis. Regardless of the course, I strive to teach students technical and critical reasoning skills that can help them excel in both academia and industry. As a researcher and teacher, I look forward to having the unique opportunity to create an environment that equips and enables my students to pursue their ambitions.