Tianyi Zhang | Teaching Statement

My philosophy about teaching, advising, and working with students is inherited from my great-grand academic advisor, Nico Habermann: "*Focus on the students, since graduating great students means you will produce great research, while focusing on the research may or may not produce great students.*" The opportunity of teaching students and igniting their curiosity about computing is what drives me. As an instructor and a mentor, my goal is to help them learn, improve, and grow into computer scientists and engineers——critical and independent thinkers with solid knowledge and skills.

Teaching Experience and Strategies

I have prepared myself through many teaching experiences since graduate school. I have been the teaching assistant for the Software Engineering classes in both UCLA and UT Austin.[1] I received the Excellent Teaching award from the ECE department at UT Austin in Fall 2013. I have also given guest lectures on *peer code reviews*, *software mining and visualization*, and *interactive program synthesis* at UCLA and University of Michigan. These experiences have shaped my teaching approach, centered around the following strategies:

#1. Active Learning. I will structure my lectures with active learning activities to engage students. In particular, I will adopt the **think-pair-share** method in my classes, which I learned from my PhD advisor and have practiced in my TA sessions. For each lecture, I will prepare a set of questions and hands-on exercises in the slides. When a question is prompted during the lecture, students will THINK to themselves first. Then, they will be instructed to discuss with a person sitting near them (PAIR). While students are discussing, I will circulate to identify pairs that are struggling and offer quick tutoring. Finally, the pairs will SHARE their discussion with the entire class.

In graduate seminar classes, I plan to **build a MiniPC paper review experience** where students get assigned to two or three papers, write reviews, and have a roundtable discussion as in a real PC meeting. While being a TA of the graduate Software Engineering class at UCLA (CS230), I have set up such a MiniPC using EasyChair and assisted the instructor to organize and moderate in-class discussions. Students in CS230 enjoyed the experience of serving on a simulated PC meeting. Compared with only asking students to read and present papers, MiniPC not only encouraged more active discussions among students but also fostered deeper thinking of the value and contribution of a research paper. It provides a more hands-on experience about how research is assessed in academia.

#2. Relating Concepts and Principles to Practices and Research. I will develop a curriculum that not only covers theoretical foundations and principles in Computer Science but also relates them to skills and practices in industry. In UCLA CS130 Software Engineering, I have covered widely-used SE practices in my TA sessions, such as peer code reviews and continuous integration. Furthermore, I will also incorporate some practical research techniques into my undergraduate classes, such as Hoare Logic, SMT solving, and static program analysis. By sharing those systematic design, analysis, testing, and verification methods, I hope that my students can not only get a good job as an entry level software engineer, but also succeed in the future as a lead, principal engineer and architect.

#3. Creating an Inclusive Classroom. From UT Austin to UCLA to Harvard, my awareness and appreciation of cross-race and cross-cultural understanding grow continually. I am committed to creating an environment where all students feel equally valued. While interacting with students with different backgrounds, learning styles, and abilities, one effective way I found is to proactively reach out to them and listen to their needs. I have observed that classroom discussions can be easily dominated by students who have more programming background, students who have easy access to resources, students who have related research experience, etc. Instead of having more privileged students dominating all classroom discussions, I will ensure my classroom includes a variety of voices from different subject positions of race, ethnicity, gender, and genre. For example, in think-pair-share, I will ask student pairs to share in a round-robin fashion, so everyone gets a chance to share their thoughts at least once through a lecture. I will also place some easy questions at the beginning of a lecture to get students speaking early, so they would feel comfortable speaking later.

#4. Soliciting Feedback from Students. It is critical for me to know how students are doing in my class, so I can create an environment effective for teaching and learning. I will ask students to give me feedback on their learning in both informal and formal ways. I will hand out index cards at the end of several lectures and ask students what questions they still have and what else they wish to learn. Furthermore, I will include an anonymous midterm survey about how my teaching strategies are helping or hindering their learning. As the semester goes, I will adapt my teaching strategies based on students' feedback and include contents students wish to learn either in

[1] As my PhD advisor moved from UT Austin to UCLA in 2014, I transferred to UCLA together with her.

my own lectures or in TA's discussion sessions.

Mentoring Experience and Strategies

I have mentored 14 students at three universities, including UT Austin, UCLA, and Harvard. I am also currently serving as the formal co-advisor of a PhD student from Macquarie University, Australia. 9 of these students have published research or demonstration papers in top-tier conferences including ICSE, ESEC/FSE, UIST, and PerCom (4 students as first authors, 5 as co-authors, and 8 papers in total).

The experience of working with these many students made me well aware that ***each student is different***. In academia, it is a common mistake for advisors to expect students to work or act like the advisors themselves in research. When I mentored the first few students during my PhD, I made this mistake and learned from it. It is critical to keep such differences in mind and find the most effective way to communicate and collaborate with students. One thing I have been practicing since then is to always share my working style first and get to know their preferences, e.g., learning styles, work-life balance, how independent they want to be, etc. By doing this, we can establish mutual understanding and set clear expectations at the very beginning, and then adapt over time.

As a mentor, my major goal is to help students ***grow into independent researchers***. Besides providing technical training and research advice, I want to help students develop their own taste and vision in research. These are what I learned from my PhD program, and I am excited to provide my students with similar experiences to help them succeed in their future careers. In particular, I have been using a conceptual framework called ***Design Arguments*** to teach my students how to analyze, select, and evaluate research ideas. I learned it from my postdoc supervisor at Harvard and found it very effective for mentally debugging my own ideas.[2] Since then, I have taught it to all my students and asked them to organize their ideas following the design argument framework. The resulting design arguments then serve as a common ground for both me and the students to work on in the next meetings, in which I can show how I assess the ideas and also give concrete advice to improve them.

Teaching Interests

At the undergraduate level, I am interested in teaching software engineering and human-computer interaction courses on topics like software design and implementation, program analysis, software testing, and usable interface design. I am also interested in teaching introduction-level programming classes, especially for non-CS majors. I can also teach undergraduate courses in programming languages and compilers, which I have extensively studied or excelled as a student. At the graduate level, I would like to teach both (1) lecture-driven courses that cover advanced yet practical software engineering, programming languages, and HCI techniques, and (2) discussion seminars on recent research and industry trends. For example, I would like to develop graduate courses on program synthesis and end-user programming, data science in software engineering, usable machine learning, and software engineering for machine learning.

[2] https://pl-hci-seminar.seas.harvard.edu/design_arguments.pdf