Teaching Statement

Baris Kasikci

Research Assistant, Ph.D. Candidate

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Mentoring and Teaching Experience

I greatly enjoy teaching. My teaching experience dates back to high school, where I used to work as a part-time tutor in geometry and chemistry. During my undergraduate studies, I worked as a part-time tutor, teaching first year students programming in C. During my industry job, I developed and taught a course on design patterns for the engineers in my team. Throughout this early experience, I appreciated observing the progress of my students and colleagues, which fostered my interest in teaching further.

My teaching experience in graduate school comes from serving as a teaching assistant (TA) for four semesters. I have TAed for an introductory programming course in C++, an undergraduate software engineering course, and a graduate course on principles of computer systems. My main responsibilities for all the courses I TAed for were the preparation and grading of assignments and exams. During this experience, I realized that students appreciate learning by doing, and this observation forms to foundation of my teaching philosophy. For software engineering, I also prepared and delivered two lectures; I led weekly recitation sessions with roughly 25 students; and I served as the head TA responsible for managing the other TAs. Being head TA was a challenging yet rewarding experience: thanks to my head TA experience, I learned to observe the working habits of my fellow TAs and bring forth their strengths to help the entire team of TAs work more efficiently and effectively. For the graduate systems course, I was responsible for conducting discussions with the class on assigned readings of seminal computer systems papers. I greatly enjoyed the participation of the students and our mutually beneficial exchange of ideas. I find graduate courses very rewarding because of the bidirectional nature of learning.

I also find it very gratifying to do research with students. It is rewarding to see the progression of a student from a novice to a collaborating researcher who can contribute to cutting-edge research. I have observed that the following three step strategy helps with this transition:

1. In the early stages, I aim to maximize my communication with the student and provide the necessary hand-holding to get them up to speed quickly. I also make sure to be upfront and clearly define goals and expectations. I have found that helping the student make rapid incremental progress in the early stages is beneficial in terms of the student's morale and helps significantly with the subsequent stages.
2. As the student gradually gains experience and familiarizes her/himself with the process of coming up with unambiguous hypotheses and testing them with experiments, I help them gain more autonomy and explore avenues on their own. Nevertheless, I still make sure the student maintains the goals of the project in focus, and works toward getting quantitative results.
3. In later stages, I expect the student to become an integral part of the project and contribute to new research ideas. I take an inclusive approach, making sure to involve the student in stages of the project involving more exploratory aspects. In return, I expect the student to be critical about our research and help me continuously question our path and decisions.

I had the chance to work with four students during my PhD (two undergraduate, two Master's). In particular, the evaluation of my root cause diagnosis work [1, 2] benefited greatly from the contributions of two of my collaborating students. Another student helped during the early stages of my data race detection work. Together with one student, we pursued the idea of an adversarial compiler that generates code that has a higher probability of encountering failures due to undefined behavior or timing errors. As a first incarnation of this approach, we built a system [3] that increases the probability of a program to encounter a deadlock while preserving program semantics.

Future Teaching Plans

As a faculty member, I would be delighted to teach operating systems, software engineering, and systems programming courses at the graduate and undergraduate level. My main goal in teaching is to distill the core principles behind a subject and convey them to students. Most of my revelations as a student happened when I realized how the abstract concepts I learned in the class *materialized* in real systems. However, I found that seeing such a connection requires time and expertise, which students may not necessarily have. In my teaching, I will aim to make these connections more explicit, with the hope to make the course content truly enjoyable.

I am also keen on developing a hands-on systems course aimed at a late-stage undergraduate level or an early-stage graduate level. I would like this course to span many disciplines (computer architecture, operating systems, programming languages, and applications) and provide a holistic view of computer systems. For this course, I will put an emphasis on building secure and efficient systems drawing on successful examples from real systems. I will adopt a hands-on approach by complementing the course material with practical assignments to build parts of real systems.

I would also be interested in teaching advanced courses on reliability and security. I find that when advanced courses are organized as seminars, exchange of ideas is encouraged and discussions can lead to potential research avenues. Therefore, I would like to teach an advanced class in the form of a seminar.

Finally, I would like to also stress that as a potential future faculty member, I feel that the onus is on academics to help make sure students from communities that are underrepresented in computer science (such as women or racial minorities) are better represented. As a faculty, I would do my part in helping underrepresented communities.

References

[1] B. Kasikci, B. Schubert, C. Pereira, G. Pokam, and G. Candea. Failure sketching: A technique for automated root cause diagnosis of in-production failures. In *Symp. on Operating Systems Principles (SOSP)*, Monterey, CA, October 2015.

[2] B. Kasikci, B. Schubert, C. Pereira, G. Pokam, M. Musuvathi, and G. Candea. Failure sketches: A better way to debug. In *Workshop on Hot Topics in Operating Systems (HotOS)*, Kartause Ittingen, Switzerland, May 2015.

[3] A. Kheradmand, B. Kasikci, and G. Candea. Lockout: Efficient testing for deadlock bugs. In *5th Workshop on Determinism and Correctness in Parallel Programming (WoDet)*, Salt Lake City, UT, March 2014.