Teaching Statement

Sebastian Baltes

Already as a bachelor student, I enjoyed helping other students as a tutor and teaching assistant for programming and IT security classes. Since I started working as a researcher in the Software Engineering Group of Trier University, I regularly taught various lectures and seminars on software engineering, programming, and information visualization topics. This experience allowed me to explore different teaching approaches in small and targeted seminar groups as well as in larger foundational lectures.

Approach

Besides building an inclusive learning environment, listening to students' feedback, and being responsive to their different backgrounds, my main goal is motivating why the taught knowledge is relevant and how it can be applied—either in a research or industry context. I am convinced that providing a clear vision for possible future applications helps students to internalize knowledge beyond just passing the corresponding exams. This assumption is supported by our recent research on software development expertise, where we found that communicating clear visions, directions, and goals is an important factor to keep developers motivated.[1]

Especially in the information visualization classes I taught, students had diverse backgrounds and many of them were studying in non-CS programs like digital humanities or computational linguistics. To suit the needs of both CS and non-CS students, I always adapted the assignments in response to the programs they were studying in. During the tutorials, I asked the students about prior knowledge and experience, but also asked about specific struggles they face with the concepts and techniques that we taught in the corresponding lectures. In classes exclusively consisting of CS majors, I still provide an interdisciplinary perspective where possible to build awareness for connections to other disciplines, but also to motivate "thinking outside the box". It is important to always be open to feedback and provide and advertise the different channels that students can use to contact the instructor, be it email, face-to-face communication in class or during office hours, or different kinds of social media.

For me, there is no "one size fits all" approach to teaching, which is why I applied different teaching models depending on class size, topic, and the background of students. I gave traditional lectures with corresponding assignments, conducted research seminars in collaboration with two local software companies, combined live programming sessions with supervised assignments, and applied the flipped classroom concept. We used the latter approach in a class for master students who wanted to specialize in software engineering but needed additional course credits. The students independently studied the online materials, and, during the lecture slots, we discussed the content and emerging questions. Those discussions and the assignments for the empirical software engineering block led to a joint research paper with three students attending the course, which recently won the best paper award at a workshop.[2] In general, to inform students about current research directions, I link the courses I teach as much as possible to my research projects as well as recently published related work. Moreover, I am constantly working on improving my teaching skills, for example by attending teaching methodology courses offered at my university.

A skill that I try to stimulate especially in master classes is critical thinking. For example, we usually include at least two paper critiques as assignments in those courses. We provide a research paper related to the lectures together with instructions how to critically read and summarize the paper. During the tutorials, we then discuss the main contributions of the paper, but also its limitations and possible threats to validity. Since the latter aspects are sometimes hidden in the papers or not mentioned at all, students learn to carefully reflect on the (proclaimed) contributions of a research paper and the context in which those contributions apply.

Experience

Complete lists of the classes I taught and the theses I supervised are available on my website.[3]

Lectures and Seminars

Since 2013, I regularly taught seminars, research internships, and tutorials on software engineering, information visualization, and programming, both for bachelor and for master students at Trier University. Since 2015, I am responsible for teaching the empirical software engineering lectures in our advanced software engineering course for master students. In the winter term 2016/2017, I revised the curriculum of our software engineering course for bachelor students and gave the corresponding lectures. In January 2019, I gave two guest lectures at the University

of Stuttgart, where I combined the above-mentioned empirical software engineering lectures with talks I gave at academic conferences. The typical class size of the undergraduate courses I taught was 40 to 80 students, the graduate classes were more focused and thus smaller (usually 15 to 25 students).

Theses

I supervised and mentored nine bachelor, master, and diploma students writing their theses at our group. Their projects were closely linked to my research, but focused on a particular (sub-)topic that the students were interested in. One of those projects was carried out in collaboration with a local software company. I always illustrated the students how their projects are embedded in a larger context, how they can contribute, and what they can learn. Five theses finally resulted in peer-reviewed publications, one in a tool prototype used in a company. The students were particularly proud to see their work being acknowledged and valued, either through publication or through usage in real-world projects. I am still in (loose) contact with most of my former students and always enjoy getting feedback on how their research projects helped them in their later job—but also which aspects of my mentoring can be improved. I am particularly proud that two of the master students I supervised now pursue PhD projects themselves.

Course Preferences

Based on my experience and expertise, I would be happy to teach any introductory programming class—also for non-CS majors—as well as any class on software engineering topics. Moreover, I could teach an information visualization class that is suitable for a broad audience but can also be focused on software visualization for CS majors. Finally, I can offer an introduction to research design and methodology to prepare students for their thesis projects, which could be extended to a whole class focused on empirical software engineering.

---

[1] https://empirical-software.engineering/publications#fse18-expertise
[2] https://empirical-software.engineering/publications#swan18-ci
[3] https://empirical-software.engineering/teaching/