Teaching Statement

Already as an undergraduate student, I enjoyed supporting others as a tutor and teaching assistant for programming and IT security classes. Later, while working as a doctoral researcher in the Software Engineering Group at Trier University in Germany, I regularly taught various lectures and seminars on software engineering, programming, and information visualization topics, and further supervised bachelor and master thesis projects. Afterwards, as a Lecturer in the School of Computer Science at the University of Adelaide in Australia, I taught classes and supervised projects in software engineering and research methodology both in face-to-face and remote settings. This diverse experience allowed me to explore different teaching approaches in small and targeted seminar groups as well as in larger foundational lectures and was especially valuable during the move to online teaching due to the COVID-19 pandemic. After moving back to Germany and starting an industry position, I continued teaching at the University of Adelaide remotely. I further gave a guest lecture at the University of Applied Sciences Darmstadt in Germany and have been involved in student supervision at HPI Potsdam.

Approach

Besides building an inclusive learning environment, listening to students' feedback, and adjusting to their different backgrounds, my main goal is to motivate 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 internalize knowledge beyond just passing the corresponding exams. This assumption is supported by our research on software development expertise, where we found that communicating clear visions, directions, and goals is an important factor to keep developers motivated. I am further a certified Scrum Master and have experience working as a senior software developer in industry, which helps me to adapt real-world process models for the project-based courses that I teach.

During my time in Trier, especially in the information visualization classes that 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 according to the student population. During the tutorials, I asked the students about prior knowledge and experience, but also asked about specific struggles they faced with the concepts and techniques that we taught in the corresponding lectures. In classes exclusively consisting of CS majors, I try to provide an interdisciplinary perspective wherever 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, during office hours, or different kinds of online channels—the latter becoming more and more important during/after the pandemic. Offering multiple contact points was especially important at the University of Adelaide where students had very different communication preferences due to their diverse backgrounds.

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 background of students. I gave traditional lectures with corresponding assignments, conducted research seminars and student projects in collaboration with local software companies, combined live programming sessions with supervised assignments, and utilized the flipped classroom concept in a graduate software engineering course in Trier. In that course, students independently studied in an online setting – we used the lecture slots to discuss the content and clarify emerging questions. Those discussions and corresponding assignments led to a joint research paper with three students attending the course, which won the best paper award at scientific workshop in 2018. 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 carefully listening to student feedback and attending teaching methodology courses. This background, and especially my previous experience with the flipped classroom concept, was very useful when we had to move courses online due to the pandemic. In some cases, we pre-recorded lectures and then used the lecture slots for interactive question-and-answer sessions with temporary breakout sessions, similar to what we previously did offline.

A skill that I try to stimulate—especially in postgraduate classes—is critical thinking. For example, we usually included at least two paper critiques as assignments in those courses. We provided a research paper related to the lectures together with instructions on how to critically read and summarize the paper. During the tutorials, we then discussed 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.

To build an inclusive classroom environment, I always establish explicit communication ground rules. I encourage students to share their experiences and perspectives frequently but always in a respectful manner. I am responsible for ensuring that all students feel welcome and valued. Of course, I, as a teacher, need to step in and speak up against any discrimination or bias that I observe in class or beyond. I aim to diversify my course materials so that they represent different perspectives, cultures, and backgrounds. I further consider it essential to accommodate diverse learning styles and potential disabilities of students. Creating accessible presentations and web content should be the default, but specific conditions might require further adjustments. At the University of Trier, for example, we had a student needing special hardware for practical programming assignments because he was almost completely paralyzed. At the University of Adelaide, we had students requiring exam supervision in dedicated rooms due to exam anxiety. It is important to actively and openly look out for students who potentially require adjustments because many disabilities are invisible, and students might be hesitant to reveal them publicly. Besides adjusting materials, teachers must also be flexible regarding their teaching mode. For example, at the University of Adelaide, many Chinese students were rather used to teacher-centered than student-centered instruction or participatory learning. This emerged as a problem during a project-based course because students' tasks included actively eliciting requirements in customer sessions. When I noticed the lack of interactivity, I adapted the sessions of the corresponding groups to provide more guidance, structure, and moderation. This intervention increased student engagement and led to more productive customer sessions.

Experience

I provide a complete list of the classes that I taught and the theses I supervised on the next page.

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. From 2015 until 2019, I was responsible for teaching the empirical software engineering lectures in the postgraduate advanced software engineering course. In the winter term 2016/2017, I revised the curriculum of the undergraduate introduction to software engineering course 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 that I gave at academic conferences. The typical class size of the undergraduate courses I taught in Trier was 40 to 80 students, the postgraduate classes were more focused and thus smaller (usually 15 to 25 students). From 2019 until 2020, at the University of Adelaide, I taught classes about different software engineering topics as well as a course on research methods. Class sizes ranged from 60 to 80 in the postgraduate classes and reached up to 350 students in the undergraduate or mixed courses. In 2020, I revised the curriculum of the undergraduate and postgraduate Software Engineering & Project course in Adelaide, moving it from a waterfall notion of software development towards agile methods, aligning it better with the demands of students as well as project owners, i.e., local South Australian companies that contributed projects. Since September 2022, I have further been involved in supervising a bachelor project group at HPI Potsdam (8 students).

Student Supervision

In Trier, I supervised and mentored nine bachelor, master, and diploma students writing their theses in 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 have 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 have pursued PhD projects themselves. In Adelaide, I continued to apply and refine my supervision skills while adding the new experience of supervising students remotely. I successfully co-supervised two master students in collaboration with a colleague at TU Eindhoven, leading to a publication in IEEE Software (2020) and one at ICSE (2023) receiving an ACM SIGSOFT Distinguished Paper award. I further co-supervised one master student in Adelaide, one master student at the University of Victoria, as well

as several smaller seminar-like student projects in Adelaide. I continue to be involved in student supervision at the University of Adelaide, where I currently supervise one graduate student. I am further mentoring one PhD student and supervising one master's student at HPI Potsdam.

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 undergraduate or postgraduate class on software engineering topics, including an *Introduction to Software Engineering* and a postgraduate class on *Advanced Topics in Software Engineering*. More targeted classes might include *Cloud-native Software Development, Software Architecture,* or *Social Software Engineering,* the latter focusing on human aspects in software development. Project-based courses and seminars should complement those classes, in particular a course like the *Software Engineering Project* course I taught at the University of Adelaide. Moreover, I could teach an *Information Visualization* class that is suitable for a broad audience but could also focus on software visualization for CS majors. Finally, I could offer a *Research Methods for Computer Scientists* course to prepare students for their thesis projects. It could be extended to a whole class on *Empirical Software Engineering* or alternatively *Software Analytics*, depending on whether the course should be broad or rather focused on quantitative methods.

List of Courses

Summer Semester 2021 (University of Applied Sciences Darmstadt):

- Guest lecture on ***Empirical Software Engineering*** in the Software Engineering course (Bachelor) at University of Applied Sciences Darmstadt.

Semester 2020-2 (University of Adelaide):

- ***Software Engineering Project*** (Undergraduate/Postgraduate): Coordinated a major revision of the course, adopting agile process models for the student projects. Further delivered lectures on software process models, Scrum, and software development tools.

Semester 2020-1 (University of Adelaide):

- ***Engineering Software as Services I*** (Undergraduate): Lectures on the architecture of SaaS applications, an introduction to Ruby, Ruby on Rails, and BDD.
- ***Software Process Improvement*** (Undergraduate/Postgraduate): Lectures on software development process models, software metrics, and estimation techniques.
- ***Research Methods in Software Engineering and Computer Science*** (Postgraduate): Lectures on research design, research methods (qualitative, quantitative, mixed-methods), descriptive statistics, clustering, measurement, reporting empirical results, sampling, controlled experiments, ethnography and interview studies, grounded theory, case studies, and surveys.

Semester 2019-2 (University of Adelaide):

- ***Introduction to Software Engineering*** (Undergraduate): Lectures on software evolution, security engineering, software testing, project management and planning, quality management, and configuration management.
- ***Engineering Software as Services II*** (Undergraduate): Lectures on JavaScript and design patterns.
- ***Software Engineering Project*** (Undergraduate/Postgraduate): Lectures on project management, software process models, software development tools, risk management, software quality management, configuration management. Further coordinated two student project groups.

Summer Semester 2019 (University of Trier):

- ***Information Visualization*** (Master): Responsible for practical assignments and tutorials, covering general properties of visualizations, infographics, UML, multivariate and timeseries data, control flow graphs, visualizations of software architectures, implementation of an algorithm visualization and a tree map in Java, paper critiques.

Winter Semester 2018/2019 (University of Trier):

- ***Advanced Software Engineering*** (Master): Lectures on continuous integration, static analysis tools, and empirical software engineering as well as corresponding assignments.

- ***Research Seminar Software Engineering*** (Bachelor/Master): Topic selection and marking.
- Two guest lectures in the ***Research Methods in Software Engineering*** course (Master) at University of Stuttgart.

Summer Semester 2018 (University of Trier):

- ***Programming II*** (Bachelor): Responsible for programming assignments in Java, covering topics such as GUI programming, concurrency, data structures, I/O, serialization, reflection, Java lambda expressions and the Stream API.
- ***Information Visualization*** (Master): See Summer Semester 2019.

Winter Semester 2017/2018 (University of Trier):

- ***Advanced Software Engineering*** (Master): See Winter Semester 2018/2019.
- ***Research Internship Software Engineering*** (Master): Supervised students working on independent research projects.
- ***Research Seminar Software Engineering*** (Bachelor/Master): Topic selection and marking.

Summer Semester 2017 (University of Trier):

- ***Programming II*** (Bachelor): See Summer Semester 2018.
- ***Information Visualization*** (Master): See Summer Semester 2019.

Winter Semester 2016/2017 (University of Trier):

- ***Software Engineering*** (Bachelor): Redesign of course lectures and assignments, delivering lectures as well as tutorials, covering topics such as requirements engineering, GUIs, UI/UX, modelling, object-oriented design, and software architecture.
- ***Independent Studies in Software Engineering*** (Master): Supervised students working in a flipped-classroom setting on software engineering topics.
- ***Research Internship Software Engineering*** (Master): Supervised students working on independent research projects.

Summer Semester 2016 (University of Trier):

- ***Programming II*** (Bachelor): See Summer Semester 2018.
- ***Information Visualization*** (Master): See Summer Semester 2019.
- ***Research Seminar Software Engineering*** (Bachelor/Master): Topic selection and marking.

Winter Semester 2015/2016 (University of Trier):

- ***Software Engineering*** (Bachelor): Responsible for assignments covering paper critiques, GUIs, UI/UX, requirements engineering, static and dynamic modeling with UML, design pattern, test coverage, and software metrics.
- ***Advanced Software Engineering*** (Master): Lecture on empirical software engineering as well as corresponding assignments.
- ***Research Internship Software Engineering*** (Master): Supervised students working on independent research projects.

Summer Semester 2015 (University of Trier):

- ***Information Visualization*** (Master): See Summer Semester 2019.
- ***Research Seminar Software Engineering*** (Bachelor/Master): Topic selection and marking.

Winter Semester 2014/2015 (University of Trier):

- ***Software Engineering*** (Bachelor): See Winter Semester 2015/2016.
- ***Advanced Software Engineering*** (Master): Tutorials covering different aspects of empirical software engineering.
- ***Research Internship Software Engineering*** (Master): Supervised students working on independent research projects.

Summer Semester 2014 (University of Trier):

- ***Information Visualization*** (Master): See Summer Semester 2019.
- ***Research Seminar HCI/UX*** (Bachelor/Master): Topic selection and marking.

Winter Semester 2013/2014 (University of Trier):

- ***Software Engineering*** (Bachelor): See Winter Semester 2015/2016.
- ***Advanced Software Engineering*** (Master): See Winter Semester 2014/2015.
- ***Research Seminar Software Engineering*** (Bachelor/Master): Topic selection and marking.

Summer Semester 2013 (University of Trier):

- ***Research Seminar Software Engineering*** (Bachelor/Master): Topic selection and marking.

List of Supervised Theses

Maria Christina Kirchner - *Stack Overflow Code Snippet Selection: An Experiment on the Effects of Source Code Comments and Answer Scoring* (Master's thesis, Uni Innsbruck, Austria, 2022)

Sterre van Breukelen - *The Survivorship of Older Women in Software Development: An Intersection between Age and Gender* (Master's thesis, TU Eindhoven, Netherlands, 2022)

Nimmi Weeraddana - *How Solution Snippets are Presented in Stack Overflow and How those Solution Snippets Need to be Adapted for Reuse* (Master's thesis, University of Victoria, Canada, 2022)

Tingsheng Lai - *Using Machine Learning to Classify Programming-related Online Snippets* (Master's thesis, University of Adelaide, Australia, 2020)

George Park - *Age(ing) in Software Development* (Master's thesis, TU Eindhoven, Netherlands, 2019)

Lorik Dumani - *Reconstructing and Linking the Version History of Stack Overflow Posts* (Master's thesis, University of Trier, Germany, 2017)

Richard Kiefer - *Stack Overflow Code Snippets in GitHub Repositories: Referenced and Unreferenced Occurrences* (Master's thesis, University of Trier, Germany, 2017)

Mert Demir - *Diskussionsverhalten englisch- und japanischsprachiger Entwickler auf Q&A-Seiten im Vergleich: Eine explorative Analyse am Beispiel von Stack Overflow* (Bachelor's thesis, University of Trier, Germany, 2017)

Bob Prevos - *Skizzieren mit Hilfe animierter Zeichnungen* (Master's thesis, University of Trier, Germany, 2016)

Fabrice Hollerich - *LivelySketches: Lifecycle Support für Skizzen* (Master's thesis, University of Trier, Germany, 2016)

Oliver Moseler: *Profiling mit Skizzen* (Master's thesis, University of Trier, Germany, 2015)

Pascal Robert - *Visuelle Worthaüfigkeitsanalyse mit THREE.js* (Master's thesis, University of Trier, Germany, 2015)

Sascha Rudolph - *Berechnung und Visualisierung ähnlicher Ordnerpaare in einem Verzeichnisbaum* (Diploma thesis, University of Trier, Germany, 2015)

Peter Schmitz - *Sketchlink Plugin: Improving software documentation and comprehension by linking source code to relevant sketches and utilizing them for navigation tasks* (Diploma thesis, University of Trier, Germany, 2014)