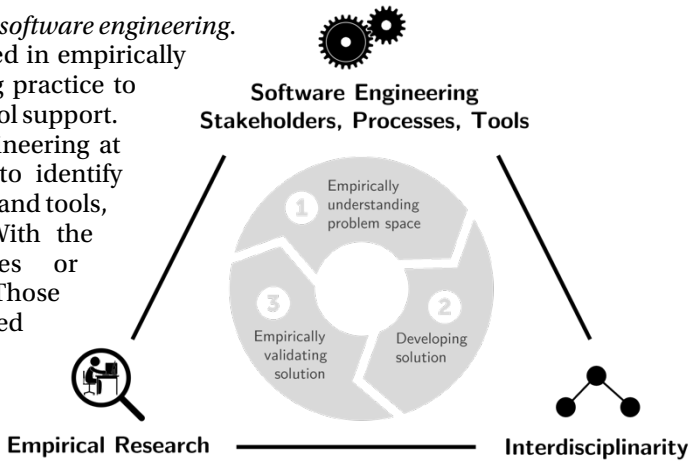
Research Statement

My research falls into what I call the *triangle of empirical software engineering*. From the beginning of my career, I have been interested in empirically studying phenomena observed in software engineering practice to derive actionable insights and requirements for better tool support. Today, as Principal Expert for Empirical Software Engineering at SAP, I regularly utilize empirical research methods to identify problems with existing software development processes and tools, intending to derive guidelines and requirements. With the resulting knowledge, I develop tool prototypes or recommendations to address the identified issues. Those suggested solutions are then again empirically validated and adapted if required. Most research projects I work on have an interdisciplinary angle, including my previous work with connections to psychology, law, social science, and management.



Exemplary for this approach to research is the first project I worked on during my PhD. While working as a software developer in industry, I noticed the widespread usage of informal diagrams without proper tool support. Triggered by this observation, I studied the problem empirically [C1] and then developed tool prototypes addressing the identified issues [S2, S6]. Interdisciplinary topics I worked on include legal implications of code plagiarism [J1] and knowledge transfer from psychology to software engineering [C6]. While *including an interdisciplinary perspective*, the studied phenomena were always *rooted in central software engineering problems* and thoroughly studied empirically before proposing solutions in form of tool prototypes or suggested process adaptations.

The guiding theme of my research has always been the idea that *thoroughly analyzing and understanding the state-of-practice is an essential first step towards improving how software is being developed*. Too often, we still see decisions in software projects being opinion-based rather than data-informed. My considerable industry experience helps me to balance the perspectives of (academic) researchers, software engineering practitioners, as well as other stakeholders. My research focuses on problems that I consider relevant for software practitioners. In addition to that, I put effort into communicating scientific results back to practice by giving talks in user groups, companies, as well as using other channels such as social media.

Besides research projects rooted in software development practice, I am also interested in the *meta-scientific discourse within the software engineering research community*. Early in my PhD research, I noticed potential ethical issues of common practices in software engineering research [S5] as well as a lack of knowledge when it comes to central aspects of empirical research in general [J8, S9]. Therefore, I am part of ACM SIGSOFT Empirical Standards initiative that maintains a continuously evolving collection of guidelines for the various empirical methods being utilized in software engineering research.[1]

I consider myself a *methods pluralist*. To complement qualitative results derived from interviews [C1], observational studies [C2], or open-ended survey questions [C1, C6], I apply data-mining techniques to open-source software projects [J1, J6, W3] or other data sets [C5, J2, S8], including data shared by companies under non-disclosure agreements [C7, C8]. From 2018 until 2021, I further maintained the open dataset *SOTorrent* that other researchers utilized to study the origin, evolution, and usage of Stack Overflow content. This dataset was selected as the official mining challenge of MSR 2019. I am also interested in information visualization and visual analytics [S1, C3], exploring how interactive visualizations can support humans in analyzing data. I regularly develop custom visualizations that have been used in different research projects—also within SAP—to explore data or to derive patterns [S8]. I follow *open science and open data* practices by publishing data, software, analysis scripts, and paper preprints whenever possible.

My vast methodological experience allowed me to react quickly when more and more developers were forced to work from home due to the COVID-19 pandemic. Together with a colleague from Dalhousie University in Canada, I initiated one of the first global studies on the impact of forced remote work on the productivity and wellbeing of software developers, including an assessment of potential support strategies that organizations can employ [J4]. With colleagues from TU Eindhoven, I further investigated how public media report on the employability of older software developers [J3], followed by an interview study on the challenges and survival strategies of veteran women software developers.

[1] https://github.com/acmsigsoft/EmpiricalStandards

Outlook

Against the above-mentioned background—and my previous work that has been published in highly ranked venues including FSE, ICSE, TSE, and EMSE—I see at least five potential avenues for my future research.

The first avenue is centered around our Google Faculty Research Award. In the corresponding project, we are exploring automated techniques for making software documentation more consistent and inclusive. While we currently focus on improving the natural language used in documentation resources, the more general question is *how software documentation ecosystems can be made more accessible*, not only for newcomers but also for experienced developers seeking information during their daily work. First steps into that direction were our work on the information diffusion between Stack Overflow and other documentation resources via hyperlinks [J2] as well as our work on GitHub Discussions [J6], which we presented internally at GitHub, later being acknowledged to have shaped the development of the tool.[2] At SAP, we currently work on a related project targeting documentation resources for SAP products and services as well as the SAP Community portal. A related general trend is the desire to have everything-as-code, including documentation, diagrams, configuration, and infrastructure definitions. Balancing and supporting the needs and desires of developers, technical writers, UI designers, end-users, and other stakeholders is an important part of the overall project.

A second—interdisciplinary—avenue is based on the idea of developing a holistic view on *software development across the lifespan*. While there is research on programming education for children as well as research on specific challenges of older software developers [J3], there is no holistic framework yet that integrates existing results from various disciplines such as software engineering, learning sciences, and especially cognitive and developmental psychology. The goal is to structure the current body of knowledge along the typical human lifespan, from learning programming as a child to keeping older developers in the workforce. Having such a framework would allow us to identify knowledge gaps, to help steer future research and education, and support software developers throughout their career. The latter aspect is particularly important because role transitions and continuous learning are prevalent in the fast-moving software industry [C6, J3].

A third avenue is part of an emerging research field targeting the *future of work*, in particular how the global pandemic has influenced software development processes and tools. While we contributed initial results based on data from an early phase of the pandemic [J4], there is an ongoing discussion about the "new normal" of work. Some companies are calling employees back to the office while others keep remote or hybrid work modes, each posing specific challenges. The agile manifesto states that "the most efficient and effective method of conveying information [...] is face-to-face conversation",[3] a statement that shows how deeply rooted the agile work culture is in on-site work. I will utilize my own work experience in remote and hybrid teams to explore effective ways of preserving the core values of agile software development in such work settings. One important question for the future of software development is if and how novel devices, sensors, and tools, including AI coding assistants, eye trackers, and novel AR/VR-devices, can support software developers in their daily work.

A fourth avenue emerged while working on a cloud native software development project in industry and has gained more traction recently while working closely with hyperscalers and SAP's cloud platform. The goal is to develop processes and tools supporting *cost-aware architectural decision-making* in software projects. Cloud native software and platform engineers frequently modify Infrastructure as Code (IaC) configuration files in their editor or development environment of choice, without suitable tool support to predict base and usage-related costs resulting from their changes. Cost monitoring tools are available, but usually reside in web frontends and are thus far away from the tools that modern developers and platform engineers use and prefer. A related topic is cloud resource demand management, e.g., suggesting suitable base configurations depending on the envisioned usage scenarios. A vendor-agnostic cost model for compute and storage costs would enable a comparison between different offerings, and could be extended to allow comparison between IaaS, PaaS, and Serverless offerings for specific workloads.

The fifth avenue is an ARC Discovery research proposal on *harnessing natural language processing to explain software defects* that I co-authored (under review). The proposed project aims to develop methods for diagnosing software-intensive systems in terms of defects, vulnerabilities, anti-patterns, and other quality concerns by utilizing natural language associated with these systems.

Besides the above-mentioned avenues, I am constantly monitoring online software development communities and talk to industry colleagues within SAP and beyond, to learn about their current challenges, complementing my own software development experience. I use those insights to steer my short- and long-term research agenda. I consider such feedback channels essential, because, especially in the last decade, too many shifts in the software industry were studied with a significant delay by software engineering researchers, leading to missed opportunities in providing evidence-based guidelines for practitioners early on.

[2] https://github.blog/2022-07-20-heres-how-academic-research-is-shaping-github-discussions/
[3] https://agilemanifesto.org/principles.html

Publication List

DBLP: https://dblp.uni-trier.de/pid/145/3950.html
Google Scholar: https://scholar.google.com/citations?user=xO09KrYAAAAJ
Website: https://empirical-software.engineering/publications/

★ Among the ten most important publications

Journal Papers (peer-reviewed)

[J9]
18 Million Links in Commit Messages: Purpose, Evolution, and Decay.
Tao Xiao, Sebastian Baltes, Hideaki Hata, Christoph Treude, Raula Gaikovina Kula, Takashi Ishio, and Kenichi Matsumoto.
Empirical Software Engineering (EMSE 2023).
https://empirical-software.engineering/publications/#emse23-commits

[J8]
Sampling in Software Engineering Research: A Critical Review and Guidelines. ★
Sebastian Baltes and Paul Ralph.
Empirical Software Engineering (EMSE 2022).
https://empirical-software.engineering/publications/#emse22-sampling

[J7]
Challenges for Inclusion in Software Engineering: The Case of the Emerging Papua New Guinean Society.
Raula Gaikovina Kula, Christoph Treude, Hideaki Hata, Sebastian Baltes, Igor Steinmacher, Marco Aurelio Gerosa, and Winifred Kula Amini.
IEEE Software (IEEE-SW 2022).
https://empirical-software.engineering/publications/#ieeesw22-bridges

[J6]
GitHub Discussions: An Exploratory Study of Early Adoption. ★
Hideaki Hata, Nicole Novielli, Sebastian Baltes, Raula Gaikovina Kula, and Christoph Treude.
Empirical Software Engineering (EMSE 2021).
https://empirical-software.engineering/publications/#emse21-ghdiscussions

[J5]
On the Diversity and Frequency of Code Related to Mathematical Formulas in Real-World Java Projects.
Oliver Moseler, Felix Lemmer, Sebastian Baltes, and Stephan Diehl.
Journal of Systems & Software (JSS 2021).
https://empirical-software.engineering/publications/#jss21-formulas

[J4]
Pandemic Programming: ★
How COVID-19 Affects Software Developers and How Their Organizations Can Help.
Paul Ralph, Sebastian Baltes, Gianisa Adisaputri, Richard Torkar, Vladimir Kovalenko, Marcos Kalinowski, Nicole Novielli, Shin Yoo, Xavier Devroey, Xin Tan, Minghui Zhou, Burak Turhan, Rashina Hoda, Hideaki Hata, Gregorio Robles, Amin Milani Fard, and Rana Alkadhi.
Empirical Software Engineering (EMSE 2020).
https://empirical-software.engineering/publications#emse20-pandemicprogramming

[J3]
Is 40 the new 60? How Popular Media Portrays the Employability of Older Software Developers. ★
Sebastian Baltes, George Park, and Alexander Serebrenik.
IEEE Software (IEEE-SW 2020).
https://empirical-software.engineering/publications#ieeesw20-ageing

[J2]

Contextual Documentation Referencing on Stack Overflow. ★
Sebastian Baltes, Christoph Treude, and Martin Robillard.
IEEE Transactions on Software Engineering (TSE 2020).
https://empirical-software.engineering/publications#tse20-condor

[J1]

Usage and Attribution of Stack Overflow Code Snippets in GitHub Projects. ★
Sebastian Baltes and Stephan Diehl.
Empirical Software Engineering (EMSE 2019).
https://empirical-software.engineering/publications#emse19-snippets

Conference Full Papers (peer-reviewed)

[C9]

"STILL AROUND": Experiences and Survival Strategies of Veteran Women Software Developers. ★
Sterre van Breukelen, Ann Barcomb, Sebastian Baltes, and Alexander Serebrenik.
Proceedings of the 45th International Conference on Software Engineering (ICSE 2023).
Acceptance rate: 26% (209/796).
https://empirical-software.engineering/publications#icse23-still-around

[C8]

Characterizing Search Activities on Stack Overflow.
Jiakun Liu, Sebastian Baltes, Christoph Treude, David Lo, Yun Zhang, and Xin Xia.
Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021).
Acceptance rate: 24% (97/396).
https://empirical-software.engineering/publications/#fse21-search

[C7]

Automated Query Reformulation for Efficient Search Based on Query Logs from Stack Overflow.
Kaibo Cao, Chunyang Chen, Sebastian Baltes, Christoph Treude, and Xiang Chen.
Proceedings of the 43rd International Conference on Software Engineering (ICSE 2021).
Acceptance rate: 23% (138/602).
https://empirical-software.engineering/publications#icse21-reformulation

[C6]

Towards a Theory of Software Development Expertise. ★
Sebastian Baltes and Stephan Diehl.
Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018).
Acceptance rate: 21% (61/289).
https://empirical-software.engineering/publications#fse18-expertise

[C5]

SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts. ★
Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl.
Proceedings of the 15th International Conference on Mining Software Repositories (MSR 2018).
Acceptance rate: 33% (37/113).
https://empirical-software.engineering/publications#msr18-sotorrent

[C4]

Constructing Urban Tourism Space Digitally: A Study of Airbnb Listings in Two Berlin Neighborhoods.
Natalie Stors and Sebastian Baltes.
Proceedings of the ACM on Human-Computer Interaction, Vol. 2, Issue CSCW, Article 166 (PACMHCI/CSCW 2018).
Acceptance rate: 26% (185/722).
https://empirical-software.engineering/publications#cscw18-airbnb

[C3]
Visual Analysis and Coding of Data-Rich User Behavior.
Tanja Blascheck, Fabian Beck, Sebastian Baltes, Thomas Ertl, and Daniel Weiskopf.
Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST 2016).
Acceptance rate: 32% (50/157).
https://empirical-software.engineering/publications#vast16-codingtool

[C2]
Navigate, Understand, Communicate: How Developers Locate Performance Bugs.
Sebastian Baltes, Oliver Moseler, Fabian Beck, and Stephan Diehl.
Proceedings of the 9th International Symposium on Empirical Software Engineering and Measurement (ESEM 2015).
Acceptance rate: 25% (20/81).
https://empirical-software.engineering/publications#esem15-debugging

[C1]
Sketches and Diagrams in Practice. ★
Sebastian Baltes and Stephan Diehl.
Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014).
Acceptance rate: 22% (61/273).
https://empirical-software.engineering/publications#fse14-sketches

Conference Short and Vision Papers (peer-reviewed)

[S12]
Visually Analyzing Company-wide Software Service Dependencies: An Industrial Case Study.
Sebastian Baltes, Brian Pfitzmann, Thomas Kowark, Christoph Treude, and Fabian Beck.
11th IEEE Working Conference on Software Visualization (VISSOFT 2023, to appear).
https://empirical-software.engineering/publications/#vissoft23-service-dependency-viz

[S11]
From Full-fledged ERP Systems Towards Process-centric Business Process Platforms.
Lukas Böhme, Tobias Wuttke, Ralf Teusner, Michael Perscheid, Sebastian Baltes, Christoph Matthies, and Benedict Bender.
29th Americas Conference on Information Systems (AMCIS 2023, to appear).
https://empirical-software.engineering/publications/#amcis23-bpp

[S10]
Applying Information Theory to Software Evolution.
Adriano Torres, Sebastian Baltes, Christoph Treude, and Markus Wagner.
Proceedings of the 2nd International Workshop on Natural-Language-based Software Engineering (NLBSE 2023).
https://empirical-software.engineering/publications/#nlbse23-entropy

[S9]
Paving the Way for Mature Secondary Research: The Seven Types of Literature Review.
Paul Ralph and Sebastian Baltes.
Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022).
Acceptance rate: 25% (7/28).
https://empirical-software.engineering/publications/#fse22-literature-reviews

[S8]
Code Duplication on Stack Overflow.
Sebastian Baltes and Christoph Treude.
Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020).
Acceptance rate: 30% (28/93).
https://empirical-software.engineering/publications#icse20-clones

[S7]
SOTorrent: Studying the Origin, Evolution, and Usage of Stack Overflow Code Snippets.
Sebastian Baltes, Christoph Treude, and Stephan Diehl.
Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019).
Acceptance rate: 33% (1/3).
https://empirical-software.engineering/publications#msr19-sotorrent

[S6]
Round-Trip Sketches: Supporting the Lifecycle of Software Development Sketches from Analog to Digital and Back.
Sebastian Baltes, Fabrice Hollerich, and Stephan Diehl.
2017 IEEE Working Conference on Software Visualization (VISSOFT 2017).
Acceptance rate: 59% (10/17).
https://empirical-software.engineering/publications#vissoft17-livelysketches

[S5]
Worse Than Spam: Issues In Sampling Software Developers.
Sebastian Baltes and Stephan Diehl.
Proceedings of the 10th International Symposium on Empirical Software Engineering and Measurement (ESEM 2016).
Acceptance rate: 37% (23/61).
https://empirical-software.engineering/publications#esem16-sampling

[S4]
Effects of Sketching on Program Comprehension (Research Plan).
Sebastian Baltes and Stefan Wagner.
Proceedings of the 17th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2016).
Acceptance rate: 38% (5/13).
https://empirical-software.engineering/publications#xp16-sketching-experiment

[S3]
VisualCues: Visually Explaining Source Code in Computer Science Education.
Benjamin Biegel, Sebastian Baltes, Bob Prevos, and Stephan Diehl.
Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2015).
Acceptance rate: 48% (36/75).
https://empirical-software.engineering/publications#vlhcc15-visualcues

[S2]
Linking Sketches and Diagrams to Source Code Artifacts.
Sebastian Baltes, Peter Schmitz, and Stephan Diehl.
Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014 Research Demos).
Acceptance rate: 65% (15/23).
https://empirical-software.engineering/publications#fse14-sketchlink

[S1]
RegViz: Visual Debugging of Regular Expressions.
Fabian Beck, Stefan Gulan, Benjamin Biegel, Sebastian Baltes, and Daniel Weiskopf.
Proceedings of the 36th International Conference on Software Engineering (ICSE 2014 NIER).
Acceptance rate: 24% (35/146).
https://empirical-software.engineering/publications#icse14-regviz

Workshop Papers and Extended Abstracts (peer-reviewed)

[W4]
An Annotated Dataset of Stack Overflow Post Edits.
Sebastian Baltes and Markus Wagner.
Genetic and Evolutionary Computation Conference Companion Proceedings (GECCO 2020 Companion), 9th Genetic Improvement Workshop.
https://empirical-software.engineering/publications#geccogi20-soedits

[W3]
(No) Influence of Continuous Integration on the Commit Activity in GitHub Projects.
Sebastian Baltes, Jascha Knack, Daniel Anastasiou, Ralf Tymann, and Stephan Diehl.
Proceedings of the 4th International Workshop on Software Analytics (SWAN 2018).
Acceptance rate: 64% (7/11).
https://empirical-software.engineering/publications#swan18-ci

[W2]
Attribution Required: Stack Overflow Code Snippets in GitHub Projects.
Sebastian Baltes, Richard Kiefer, and Stephan Diehl.
Proceedings of the 39th International Conference on Software Engineering Companion (ICSE 2017).
https://empirical-software.engineering/publications#icse17-snippets

[W1]
CodeBasket: Making Developers' Mental Model Visible and Explorable.
Benjamin Biegel, Sebastian Baltes, Ivan Scarpellini, and Stephan Diehl.
Proceedings of the 2nd International Workshop on Context for Software Development (CSD 2015).
https://empirical-software.engineering/publications#csd15-codebasket