Thomas Zimmermann – Research Statement

Email: tz@acm.org Web: thomas-zimmermann.com

Software engineering plays a pivotal role in shaping the technological landscape of the modern world, as it drives innovation, supports complex systems, and enables the creation of robust and scalable solutions. I have **15+ years of experience with cutting-edge software engineering research** with a focus on how we can use data to support decision making and improve the creation of software. My technical achievements are impacting daily practices at many of the world's largest software companies and my foundational work continues to shape academic research worldwide. I've published 150+ papers (27K+ citations, h-index: 78) in software engineering's highest impact venues. I am an **IEEE Fellow** for *"contributions to data science in software engineering, research and practice",* an **ACM Fellow** for "*contributions to mining software repositories and defect prediction*", and an **IEEE Edward McCluskey Technical Achievement Award** recipient.

My research philosophy is to focus on software engineering in a practical setting and combine building tools with empirical studies. As a **data scientist at heart**, my journey into the realm of numbers and patterns began in my childhood, where I first discovered the joy of turning raw data into meaningful insights. I continued my passion for data when I started my research career in software engineering. Initially, my work was deeply rooted in the automated mining of software data sources and quantitative methodologies. However, with time, my professional pursuits have evolved to put more focus on the people creating the software and to integrate qualitative methodologies (surveys, interviews) into my research repertoire. This progression has not only enriched my understanding of software development but has also allowed me to contribute more comprehensively to the field of software engineering as well as other fields (e.g., HCI, CSCW, software security, digital games).

In this statement, I will describe my contributions to mining software repositories, my current work on developer productivity and experience, data science and artificial intelligence as well as my plans for future research. Much of the research highlighted in this statement, though often described from a personal perspective, is the product of collaborative work, and I have been fortunate to work with remarkable collaborators throughout my career, each contributing their unique expertise and vision.

Mining Software Repositories

I am best known for **pioneering the Mining Software Repositories** (MSR) field, also known as software analytics. The MSR field analyzes software data from version archives, bug databases, pull requests, and telemetry, to uncover actionable information about software systems, projects, and users. My algorithms to extract, link, and analyze data across software repositories are at the core of many MSR and SE innovations, for example:

- My landmark paper on the automated mining of data from software repositories [TSE05] is considered the founding paper of the MSR field. I demonstrated the potential of analyzing software repositories by building the **first recommender system for related files by automatically mining rules from version control systems**. These recommendations helped developers navigate through software and prevent defects caused by missing changes. The paper has 1,700+ citations and is one of the most cited papers in software engineering. Papers on MSR now account for 20% of submissions to top IEEE/ACM SE conferences (ICSE, FSE).

- My paper on identifying defect-introducing changes [MSR05] introduced **foundational algorithms for the identification of bug fix commits and fix-inducing commits** — commits that likely introduced a bug. With 1,250+ citations, this is the most cited paper in the 20-year history of the MSR conference. The techniques introduced in this paper are now used daily in approaches for bug prediction, detection, and localization to improve software quality. A systematic literature review identified 273 papers that used the SZZ algorithm for their research.

Defect prediction, a subfield of software analytics, is the creation of data science models for identifying potentially defective areas in a software system. These models are commonly used to assess code quality and guide the limited quality assurance resources. A bibliometric assessment published in the Journal of Systems and Software identified me as a **top scholar in defect prediction**. My work on defect prediction had industry-wide implications on software quality. My research is applied at Microsoft (for example, Windows, the largest operating system in the world), at other companies (Google), and in open-source projects.

My technical achievements focused on **removing the limitations of traditional defect prediction models to make them useful in industrial practice**.

- To reduce the need for retraining, I pioneered adaptive defect prediction models, which automatically adapt to the changing contexts of software development. Using a **cache model with different types of localities and replacement strategies**, the FixCache approach selected the 10% of files that accounted for 73%- 95% of future defects [ICSE07]. FixCache was later deployed independently at Google. Adaptive models do not need frequent retraining like static models when the context of software development changes.

- To address the limitations of project-specific models, I investigated to what extent models from one project can predict defects in another project. I **pioneered cross-project defect prediction models, which are useful when little or no training data is available** [ESEC/FSE09]. I showed the feasibility of transferring models with Firefox and Internet Explorer. In a large-scale experiment with 622 experiments, I showed that cross-project defect prediction is a hard, non-trivial problem: only in 3.4% of the experiments, the prediction models could be applied out of the box. Since 2009 cross-project defect prediction has become one of the most active research areas in defect prediction with 200+ published papers.

Developer Productivity and Experience

I am also known for my empirical studies of software development practices in industry and open source. My research focused on aspects such as developer productivity and satisfaction, work-life balance, task management, perceptions, and experiences with creating software systems. For example, I have studied perceptions of productivity [FSE14], retrospections [CSCW17], software engineering managers [TSE19], developer satisfaction [TSE21a], reflective goal setting [TSE21b], physical work environments [TSE21c], and objectives and key results in software teams [ICSE-SEIP24].

SPACE framework. I am one of the co-creators of the SPACE framework, a conceptual model developed to understand and measure developer productivity. SPACE is **based on decades of productivity research.** SPACE identifies five dimensions of productivity: Satisfaction, Performance, Activity, Communication, and Efficiency. The SPACE framework moves beyond traditional, narrow measures of productivity (like lines of code) and encompasses a broader, more holistic view of what impacts a developer's work and productivity. It's particularly relevant in modern software engineering, where teamwork, well-being, and a diverse range of activities are integral to successful project outcomes. The paper on the SPACE framework is the **sixth most frequently downloaded paper among 750K papers in the ACM Digital Library.** [QUEUE21]

Impact of work from home on productivity. I led several projects at Microsoft on how the COVID-19 pandemic and work from home policy affected productivity, and social connectedness of employees. The initial research identified a list of challenges, benefits, and best practices and quantified their impact on productivity. The work highlighted that there was a dichotomy of developer experiences influenced by many different factors [TOSEM21]. Thousands of employees started to work for Microsoft during the pandemic. Through interviews and a survey, our team identified a list of recommendations that teams can follow to onboard new hires more efficiently [ICSE-SEIP21] and to work more efficiently as a team [ICSE21, ACM SIGSOFT Distinguished Paper]. The findings from this research were prominently featured within Microsoft, included in the Microsoft New Future of Work reports, and impacted thought leadership across the company.

Data Science, Artificial Intelligence and Software Engineering

I spearheaded data science projects related to several Microsoft software products. My work influenced the development practices of some of the world's most important software systems at Microsoft and Google. For Windows, which is built by 4,000 engineers and used by 1.5 billion people, my work has been influential in speeding up development by 8.9 days through improved branch structures [FSE12]. For Xbox (48 million members), my analytics models of player behavior and engagement have improved the retention of millions of video game players [CHI13]. For example, I invented a technique to create engagement profiles of 1.2 million players of Forza Motorsport 5, a popular Xbox car racing game, and highlighted patterns associated with active gameplay. Subsequent Forza titles used the profiles to improve the game design. The analysis was repeated for other Microsoft games such as Killer Instinct, Solitaire, Minecraft, and Halo Reach (3 million players). [CHIPLAY15].

Artificial intelligence (AI) has been another central component of my research. My work has been **leveraging AI techniques for software engineering tasks**. For example, I used language models such as GPT 3 and 3.5 to recommend root causes and mitigation steps for cloud incidents. More than 70% of the on-call engineers gave a rating of three or above (out of 5) for the usefulness of recommendations in a real-time production setting. This is an important technology to swiftly resolve disruptions, maintain service reliability, and prevent future occurrences of similar issues for cloud services [ICSE23].

I have also **focused on the engineering of AI-powered software applications**. I published a highly influential case study on software engineering for machine learning [ICSE-SEIP19, IEEE Software Best SEIP Paper, 830 citations]. This paper explored the integration of AI capabilities into software and services. It reported on a study observing Microsoft software teams as they develop AI-powered applications and identified a nine-stage workflow process as well as a maturity model. The paper discussed the unique challenges faced

in machine learning applications compared to traditional software engineering, such as complex data management, model customization, and difficulty in handling AI components as distinct modules. This work helped design a process for responsible AI in Microsoft software products.

Finally, I worked on **empirical studies of how software developers use AI-powered tools** such as GitHub Copilot. GitHub Copilot is an AI-powered code completion tool that uniquely assists programmers by suggesting entire lines or blocks of code in a wide range of programming languages, learning from the context provided in existing code. My work on analyzing GitHub user behavior and feedback helped shape the GitHub Copilot product launch and informed features and pricing strategy [QUEUE22].

Future work

The emergence of software and AI-driven technologies has transformed industries across the world and reshaped the way we work, communicate, and live. This paradigm shift represents a unique opportunity in history to engage in software research, as it enables us to explore novel applications, address complex challenges, and unlock untapped potential across various domains. In 2011, Marc Andreessen famously said that "Software is eating the world", and now in 2024, AI is eating the world as well. My future research will focus on several aspects at the **intersection of AI and software** and their profound impact on society, shaping the future of technology and innovation.

- **How we consume software is changing.** AI is fundamentally altering how we consume and interact with software, moving away from static products to dynamic, evolving services that are hidden behind a common interaction model. With AI, software will also be capable of increased personalization and adaptation to user needs. It will be important to understand how people are using software in this new world. I have started some work in this direction by investigating trust in AI-powered tools and their recommendations. This led to the PICSE framework for trust in software tools which identified five dimensions of trust [ICSE-SEIP23].
- **Democratization of software development**. AI will democratize software development, making the creation of software accessible to a broader range of people, regardless of their technical background. By automating repetitive tasks, offering code suggestions, and simplifying development processes, AI tools will lower the barriers to entry, empowering more people to participate in software creation and innovation.
- **AI and software for social good**. With my colleague Denae Ford Robinson, I investigated the motivations and challenges of contributing to open source for social good [ICSE21]. AI holds immense potential for software applications aimed at social good, such as addressing healthcare disparities, optimizing resource allocation, and aiding disaster response. By harnessing AI's capabilities, we can develop innovative solutions that have a positive impact on society, improving the quality of life for communities worldwide.

I'm frequently asked about the possibility of AI replacing software engineers. While AI is a disruption and the role of software engineers will change in the future, AI won't replace software engineers. Instead, it will augment their capabilities by automating routine tasks, enabling engineers to focus on more complex and creative aspects of software development. Software engineers will need to adapt by acquiring skills in AI, data science, and continuous learning to thrive in this evolving technological landscape. With my future research, I hope to maintain a positive impact on the ongoing transformation of software by AI.

References

[CHI13] Jeff Huang, Thomas Zimmermann, Nachiappan Nagappan, Charles Harrison, Bruce C. Phillips: Mastering the art of war: how patterns of gameplay influence skill in Halo. ACM SIGCHI Conference on Human Factors in Computing Systems (CHI) 2013: 695-704. **"Best of CHI" Honorable Mention Award.**

[CHIPLAY15] Erik Harpstead, Thomas Zimmermann, Nachiappan Nagappan, Jose J. Guajardo, Ryan Cooper, Tyson Solberg, Dan Greenawalt: What Drives People: Creating Engagement Profiles of Players from Game Log Data. ACM Annual Symposium on Computer-Human Interaction in Play (CHI PLAY) 2015: 369-379

[CSCW17] André N. Meyer, Gail C. Murphy, Thomas Zimmermann, Thomas Fritz: Retrospecting on Work and Productivity: A Study on Self-Monitoring Software Developers' Work. Proc. ACM Hum. Comput. Interact. 1(CSCW): 79:1-79:24 (2017)

[ESEC/FSE09] Thomas Zimmermann, Nachiappan Nagappan, Harald Gall, Emanuel Giger, Brendan Murphy: Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. European Software Engineering Conference and ACM SIGSOFT Intl. Symposium on Foundations of Software Engineering (ESEC/FSE) 2009: 91-100. **ESEC/FSE Test of Time Award.**

[FSE12] Christian Bird, Thomas Zimmermann: Assessing the value of branches with what-if analysis. ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE) 2012: 11 pages. **ACM SIGSOFT Distinguished Paper Award.**

[FSE14] André N. Meyer, Thomas Fritz, Gail C. Murphy, Thomas Zimmermann: Software developers' perceptions of productivity. ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE) 2014: 19-29

[ICSE07] Sunghun Kim, Thomas Zimmermann, E. James Whitehead Jr., Andreas Zeller: Predicting Faults from Cached History. IEEE/ACM International Conf. on Software Engineering (ICSE) 2007: 489-498. **ACM SIGSOFT Distinguished Paper Award.**

[ICSE21] Courtney Miller, Paige Rodeghero, Margaret-Anne D. Storey, Denae Ford, Thomas Zimmermann: "How Was Your Weekend?" Software Development Teams Working From Home During COVID-19. IEEE/ACM International Conf. on Software Engineering (ICSE) 2021: 624-636. **ACM SIGSOFT Distinguished Paper Award.**

[ICSE21] Yu Huang, Denae Ford, Thomas Zimmermann: Leaving My Fingerprints: Motivations and Challenges of Contributing to OSS for Social Good. IEEE/ACM International Conf. on Software Engineering (ICSE) 2021: 1020-1032

[ICSE23] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, Saravan Rajmohan: Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models. IEEE/ACM International Conf. on Software Engineering (ICSE) 2023: 1737-1749

[ICSE-SEIP19] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald C. Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, Thomas Zimmermann: Software engineering for machine learning: A case study. IEEE/ACM International Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2019: 291-300. **IEEE Software Best SEIP Paper Award.**

[ICSE-SEIP21] Paige Rodeghero, Thomas Zimmermann, Brian Houck, Denae Ford: Please Turn Your Cameras on: Remote Onboarding of Software Developers During a Pandemic. IEEE/ACM Intl. Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2021: 41-50

[ICSE-SEIP23] Brittany Johnson, Christian Bird, Denae Ford, Nicole Forsgren, Thomas Zimmermann: Make Your Tools Sparkle with Trust: The PICSE Framework for Trust in Software Tools. IEEE/ACM International Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2023: 409-419

[ICSE-SEIP24] Jenna L. Butler, Thomas Zimmermann, Christian Bird: Objectives and Key Results in Software Teams: Challenges, Opportunities and Impact on Development. IEEE/ACM Intl. Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2024.

[MSR05] Jacek Sliwerski, Thomas Zimmermann, Andreas Zeller: When do changes induce fixes? IEEE/ACM International Workshop on Mining Software Repositories (MSR) 2005. 5 pages. **MSR Most Influential Paper Award.**

[QUEUE21] Nicole Forsgren, Margaret-Anne D. Storey, Chandra Shekhar Maddila, Thomas Zimmermann, Brian Houck, Jenna L. Butler: The SPACE of Developer Productivity: There's more to it than you think. ACM Queue 19(1): 20-48 (2021). **Sixth most frequently downloaded paper in the ACM DL among 750K+ papers.**

[QUEUE22] Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, Idan Gazit: Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. ACM Queue 20(6): 35-57 (2022)

[TOSEM21] Denae Ford, Margaret-Anne D. Storey, Thomas Zimmermann, Christian Bird, Sonia Jaffe, Chandra Shekhar Maddila, Jenna L. Butler, Brian Houck, Nachiappan Nagappan: A Tale of Two Cities: Software Developers Working from Home during the COVID-19 Pandemic. ACM Trans. Softw. Eng. Methodol. 31(2): 27:1-27:37 (2022)

[TSE05] Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, Andreas Zeller: Mining Version Histories to Guide Software Changes. IEEE Trans. Software Eng. 31(6): 429-445 (2005). **ICSE Most Influential Paper Award for the ICSE 2004 paper.**

[TSE19] Eirini Kalliamvakou, Christian Bird, Thomas Zimmermann, Andrew Begel, Robert DeLine, Daniel M. Germán: What Makes a Great Manager of Software Engineers? IEEE Trans. Software Eng. 45(1): 87-106 (2019)

[TSE21a] Margaret-Anne D. Storey, Thomas Zimmermann, Christian Bird, Jacek Czerwonka, Brendan Murphy, Eirini Kalliamvakou: Towards a Theory of Software Developer Job Satisfaction and Perceived Productivity. IEEE Trans. Soft. Eng. 47(10): 2125-2142 (2021)

[TSE21b] André N. Meyer, Gail C. Murphy, Thomas Zimmermann, Thomas Fritz: Enabling Good Work Habits in Software Developers through Reflective Goal-Setting. IEEE Trans. Software Eng. 47(9): 1872-1885 (2021)

[TSE21c] Brittany Johnson, Thomas Zimmermann, Christian Bird: The Effect of Work Environments on Productivity and Satisfaction of Software Engineers. IEEE Trans. Software Eng. 47(4): 736-757 (2021)