Rethinking Productivity in Software Engineering

Edited by
Caitlin Sadowski
Thomas Zimmermann

Apress open

Rethinking Productivity in Software Engineering

Caitlin Sadowski
Mountain View, CA, USA

Thomas Zimmermann
Bellevue, WA, USA

ISBN-13 (pbk): 978-1-4842-4220-9
ISBN-13 (electronic): 978-1-4842-4221-6
https://doi.org/10.1007/978-1-4842-4221-6

Library of Congress Control Number: 2019934471

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: James Markham
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484242209. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

To Mr. Wiggles.

—Caitlin Sadowski

To my parents.

—Thomas Zimmermann

Table of Contents

About the Editors

Dr. Caitlin Sadowski is a software engineer at Google in Mountain View, California, where she aims to understand and improve developer workflows. Currently, she is helping Chrome developers make data-driven decisions as the manager of the Chrome Metrics team. In the past, she made static analysis useful at Google by creating the Tricorder program analysis platform, and then co-founded a team that provides ongoing insight into how developers spend their time and what makes them effective (the Engineering Productivity Research team). She is a committee member of top software engineering and programming language conferences (ICSE, ESEC/FSE, OOPSLA, and PLDI). She has a PhD from the University of California at Santa Cruz where she worked on a variety of research topics related to programming languages, software engineering, and human computer interaction. She enjoys baking with her three-year-old, Naru (otherwise known as Mr. Wiggles).

Dr. Thomas Zimmermann is a senior researcher at Microsoft Research, where he analyzes data for a living. Currently, he works on the productivity of software developers and data scientists at Microsoft. In the past, he analyzed data from digital games, branch structures, and bug reports. He is the co-editor in chief of the Empirical Software Engineering journal and serves on the editorial boards of IEEE Transactions on Software Engineering, IEEE Software, Journal of Systems and Software, and Journal of Software: Evolution and Process. He is a committee member of top software engineering conferences (ICSE, ESEC/FSE, and ASE) and the chairman of ACM SIGSOFT. He previously edited books on recommender systems (Springer) and data science in software engineering (Morgan Kaufmann). He has a PhD from Saarland University where he worked on mining software repositories. He likes movies, enjoys football at -6 degrees Fahrenheit, and collects unicorns.

Acknowledgments

There are many people who made this book possible. We gratefully acknowledge the extensive and professional work of our authors and the Apress team, especially Todd Green, Jill Balzano, and Susan McDermott. Special thanks to the staff and the organizers of Schloss Dagstuhl (`https://www.dagstuhl.de`, where computer scientists meet), who hosted the original meeting that was the genesis of this book. Special thanks also to Jaeheon Yi and Ambrose Feinstein, without whom it would have been impossible to find the time to work on this.

Introduction

Caitlin Sadowski
Thomas Zimmermann

As Marc Andreessen put it, software is eating the world [1], and there is an ever-growing demand on software being built. Despite the immense growth in the number of professional software developers, there is still a shortage. To satisfy this demand, we need more productive software engineers.

Over the past four decades, there has been significant research on understanding and improving the productivity of software developers and teams. A substantial amount of work has examined the meaning of software productivity. Much of this introduced definitions of productivity (many of them!), considered organizational issues associated with productivity, and focused on specific tools and approaches for improving productivity. In fact, most of the seminal work on software productivity is from the 1980s and 1990s (Peopleware, Mythical Man-Month, Personal Software Process).

Why This Book?

Historically, this book began as a weeklong workshop in Dagstuhl, Germany [2]. The motivation for this seminar was that since the 1980s and 1990s many things have changed and that it was time to revisit what makes *modern* software engineers productive.

What has changed since the 1980s and 1990s? Today's software teams and engineers are often global and collaborate across borders and time zones, practice agile software development, frequently use social coding tools such as Stack Overflow and GitHub, and often work on laptops or their own personal devices. Today's software engineers must deal with unprecedented complexity, can build large systems fast in the cloud, can store millions (or even billions) of lines of code in a single repository, and can release software frequently, often multiple times a day. They use on average 11.7 communication channels such as web search, blogs, Q&A sites, and social networking sites [85]; in 1984, the primary communication channels for software engineers were phone calls and in-person meetings [27]. The human-computer interaction (HCI) and computer-supported cooperative work (CSCW) communities have made significant advances in supporting knowledge workers to become more productive that one might also transfer to software engineers. Furthermore, the wide availability of data about software development enables a more sophisticated analysis of software productivity.

The goal of this seminar was to rethink, discuss, and address open issues of productivity in software development and figure out how to measure and foster productive behavior of software developers. Specifically, the discussion at the seminar focused on the following questions:

- What does productivity mean for individuals, teams, and organizations?
- What are the dimensions and factors of productivity?
- What are the purposes and implications of measuring productivity?
- What are the grand challenges in research on productivity?

This book explores what productivity means for modern software development. The chapters were written by participants at the Dagstuhl seminar (see Figure 1), plus numerous other experts. Our goal is to summarize and distribute their combined experience, wisdom, and understanding about software productivity.

© SCHLOSS DAGSTUHL – LZI GMBH
licensed under Creative Commons License CC BY-NC-ND

Figure 1. *The attendees of the Dagstuhl seminar called "Rethinking Productivity in Software Engineering" in March 2017. The two editors of this book are in the second row on the right hand side.*

About This Book

This book is organized into five topic areas. We begin with a set of essays outlining challenges with measuring productivity ("Measuring Productivity: No Silver Bullet"). This is followed by essays focused on breaking down productivity into its components ("Introduction to Productivity") and essays that identify productivity factors and how they may give a different perspective on productivity ("The Context of Productivity"). Even though productivity is difficult to measure in general, we include specific case studies focused on measuring some aspect of productivity ("Measuring Productivity in Practice"). We finish with a series of essays on interventions that do work to improve productivity ("Best Practices for Productivity").

Measuring Productivity: No Silver Bullet

Are some programmers indeed ten times more productive than others, as some people claim? Lutz Prechelt digs into the data to address this question in Chapter 1. Ciera Jaspan and Caitlin Sadowski then explain what is inherently wrong with focusing on a single productivity metric (and what you can do instead) in Chapter 2. Amy J. Ko describes a thought experiment identifying the unintended consequences of measuring productivity in Chapter 3.

An Introduction to Productivity

We begin this part with an overview of ways that productivity has been defined in the past with Chapter 4 by Stefan Wagner and Florian Deissenboeck. In Chapter 5, Caitlin Sadowski, Margaret-Anne Storey, and Robert Feldt describe a framework for breaking down productivity into three dimensions: quality, velocity, and satisfaction—and how to apply that framework when considering productivity metrics. Amy J. Ko then describes how it is important to consider productivity in context through a particular lens in Chapter 6. Emerson Murphy-Hill and Stefan Wagner conclude this introduction to productivity concepts with an overview of productivity research in a related context (knowledge work) in Chapter 7.

The Context of Productivity

There are many different factors that may affect the productivity of software engineers. Stefan Wagner and Emerson Murphy-Hill overview the space of these factors in Chapter 8. We do a deep dive into two of these factors in the following two chapters: Duncan Brumby, Christian Janssen, and Gloria Mark provide an overview of research on interruptions in Chapter 9, and then Daniel Graziotin and Fabian Fagerholm discuss research about the relationship between happiness and productivity in Chapter 10. We end this part with Pernille Bjørn's cautionary tale about the importance of considering social factors for productivity in Chapter 11.

Measuring Productivity in Practice

André N. Meyer, Gail C. Murphy, Thomas Fritz, and Thomas Zimmermann dig into the varying ways developers perceive productivity and the implications for self-reported productivity measurement in Chapter 12. Brad A. Myers, Amy J. Ko, Thomas D. LaToza, and YoungSeok Yoon then discuss how qualitative research methods can aid in understanding productivity challenges or improvements in Chapter 13. Marieke van Vugt then overviews the benefits and limitations of using eye trackers and electroencephalography (EEG) scans to measure productivity in Chapter 14. Christoph Treude and Fernando Figueira Filho discuss the importance of awareness of what is going on in the larger team (team awareness) for productivity and investigate how team awareness can be measured in Chapter 15. In Chapter 16, Margaret-Anne Storey and Christoph Treude overview benefits and challenges of presenting productivity metrics in dashboards.

Some organizations perform productivity benchmarking using International Organization for Standardization (ISO) standard methods; the final two chapters give a perspective into this world. Charles Symons overviews one such measurement (COSMIC) in Chapter 17. Frank Vogelezang and Harold van Heeringen describe a case study of how organizations use a benchmarking method like COSMIC in Chapter 18.

Best Practices for Productivity

There are too many "best practices" for improving the productivity of software engineers to include in this book, so we give an overview of different interventions that provide a variety of perspectives into what such an intervention could look like. Todd Sedano, Paul Ralph, and Cécile Péraire describe how changing the mind-set from "improving productivity" to "reducing waste" can make productivity improvements tractable in Chapter 19. Bill Curtis describes the importance of having clear, mature processes in Chapter 20. In Chapter 21, Franz Zieris and Lutz Prechelt give an answer to the question of whether pair programming pays off.

There are also tool-supported interventions to improve productivity. The benefits and challenges of self-tracking for productivity are described by André N. Meyer, Thomas Fritz, and Thomas Zimmermann in Chapter 22. Manuela Züger, André N. Meyer, Thomas Fritz, and David Shepherd present a system to surface information about when to interrupt software engineers in Chapter 23. In Chapter 24, Gail C. Murphy, Mik Kersten, Robert Elves, and Nicole Bryan review an evolution of technologies focused on improving the access and flow of information between the humans and tools involved in creating software systems. Lastly, Marieke van Vugt focuses inward and overviews the role of mindfulness in productivity in Chapter 25.

The Future of Software Productivity

While these essays were written by experts, they are hardly complete. Software development is always changing, and there is a lot we don't know yet about software productivity. At the Dagstuhl seminar, the attendees identified several open questions and grand challenges. The three main grand challenges are building a body of knowledge about what we know about software productivity, improving the measurement of productivity, and affecting and improving software productivity through interventions.

Building a Body of Knowledge About Software Productivity

The following are the next steps towards building a body of knowledge about software productivity:

- Develop a theoretical framework for productivity.
- Define *laws or rules of productivity* similar to the laws of software evolution. For example, a happier developer is a more productive developer; a participatory culture in a team is more productive.
- Examine the difference of software development to all other kinds of knowledge workers and learn what is unique about software development and what is not.
- Develop a mapping from questions on productivity to a methodology of studying it.

Improving the Measurement of Productivity

The following are the next steps for improving the measurement of productivity:

- Collect examples of where measuring productivity was done well with good outcomes. Distill the insights and guidelines from this collection.
- Develop an approach that can track "everything" at every moment, including detailed data across a company; biometric data from individuals; and data on aspects such as satisfaction, mood, fatigue, and motivation. Use the data to profile development work and productivity. *Obviously, it will be hard (if not impossible) to get the privacy right for an approach like this.*

Improve the Productivity of Software Engineers

The following are the next steps for improving the productivity of software engineers:

- Understand how to support and facilitate productivity.
- Conduct a multitude of comparative studies on productivity at different companies and on different interventions.

Exciting times are ahead. We hope you enjoy this book!

References

[1] Marc Andreessen. Why Software Is Eating The World. Wall Street Journal 2011. https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

[2] Thomas Fritz, Gloria Mark, Gail C. Murphy, Thomas Zimmermann. Rethinking Productivity in Software Engineering (Dagstuhl Seminar 17102). Dagstuhl Reports, Volume 7, Number 3, March 2017, pages 19–26. http://dx.doi.org/10.4230/DagRep.7.3.19

[3] M.-A. Storey, A. Zagalsky, F. F. Filho, L. Singer, and D. M. German. How social and communication channels shape and challenge a participatory culture in software development. IEEE Transactions on Software Engineering, 43(2):185–204, 2017.

[4] T. DeMarco and T. Lister. Programmer performance and the effects of the workplace. In Proceedings of the 8th international conference on Software engineering, pages 268–272. IEEE Computer Society Press, 1985.