

# 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  
<https://doi.org/10.1007/978-1-4842-4221-6>

ISBN-13 (electronic): 978-1-4842-4221-6

Library of Congress Control Number: 2019934471

Copyright © The Author(s) and Editor(s) 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.



**Open Access** This book is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this book or parts of it.

The images or other third party material in this book are included in the book's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the book's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Chapter 9 is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>). For further details, see license information in the chapter.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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](http://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](mailto:orders-ny@springer-sbm.com), or visit [www.springeronline.com](http://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](mailto:rights@apress.com), or visit [www.apress.com/rights-permissions](http://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](http://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](http://www.apress.com/9781484242209). For more detailed information, please visit [www.apress.com/source-code](http://www.apress.com/source-code).

Printed on acid-free paper

*To Mr. Wiggles.*

—*Caitlin Sadowski*

*To my parents.*

—*Thomas Zimmermann*

# Table of Contents

<b>About the Editors</b> .....	<b>xvii</b>
<b>Acknowledgments</b> .....	<b>xix</b>
<b>Introduction</b> .....	<b>xxi</b>
<b>Part I: Measuring Productivity: No Silver Bullet</b> .....	<b>1</b>
<b>Chapter 1: The Mythical 10x Programmer</b> .....	<b>3</b>
Some Work Time Variability Data .....	3
Insisting on Homogeneity.....	4
Deciding What We Even Mean.....	5
Uninsisting on Homogeneity .....	6
Questioning the Base Population .....	6
It's Not Only About Development Effort.....	7
Are Slower Programmers Just More Careful? .....	7
Secondary Factors Can Be Important .....	8
The Productivity Definition Revisited .....	9
How Would Real People Work? .....	9
So What? .....	10
Key Ideas.....	10
References.....	11
<b>Chapter 2: No Single Metric Captures Productivity</b> .....	<b>13</b>
What's Wrong with Measuring Individual Performers?.....	14
Why Do People Want to Measure Developer Productivity? .....	14

TABLE OF CONTENTS

- What's Inherently Wrong with a Single Productivity Metric? ..... 15
  - Productivity Is Broad ..... 15
  - Flattening/Combining Components of a Single Aspect Is Challenging ..... 16
  - Confounding Factors ..... 16
- What Do We Do Instead at Google? ..... 17
- Key Ideas..... 19
- References..... 19
- Chapter 3: Why We Should Not Measure Productivity ..... 21**
  - Unintended Consequences ..... 22
  - Explaining Productivity..... 23
  - Dealing with Change ..... 24
  - Managers as Measurers ..... 25
  - Key Ideas..... 26
- Part II: Introduction to Productivity..... 27**
- Chapter 4: Defining Productivity in Software Engineering ..... 29**
  - A Short History of Software Productivity..... 30
  - Terminology in the General Literature..... 32
    - Productivity..... 32
    - Profitability ..... 33
    - Performance ..... 33
    - Efficiency and Effectiveness..... 33
    - Influence of Quality..... 33
  - An Integrated Definition of Software Productivity ..... 34
  - Summary..... 36
  - Key Ideas..... 36
  - Acknowledgements..... 37
  - References ..... 37

<b>Chapter 5: A Software Development Productivity Framework .....</b>	<b>39</b>
Productivity Dimensions in Software Development .....	40
Velocity .....	40
Quality .....	40
Satisfaction.....	41
Lenses.....	41
The Productivity Framework in Action: Articulating Goals, Questions, and Metrics.....	42
Example 1: Improving Productivity Through an Intervention .....	43
Example 2: Understanding How Meetings Impact Productivity.....	44
Caveats .....	45
Key Ideas.....	46
References.....	46
<b>Chapter 6: Individual, Team, Organization, and Market: Four Lenses of Productivity .....</b>	<b>49</b>
The Individual.....	49
The Team.....	50
The Organization .....	51
The Market.....	52
Full-Spectrum Productivity .....	53
Key Ideas.....	53
References.....	53
<b>Chapter 7: Software Productivity Through the Lens of Knowledge Work.....</b>	<b>57</b>
A Brief History of Knowledge Work .....	57
Techniques for Measuring Productivity.....	58
Outcome-Oriented Techniques .....	58
Process-Oriented Techniques.....	59
People-Oriented Techniques.....	60
Multi-oriented Techniques.....	60

TABLE OF CONTENTS

- Drivers That Influence Productivity ..... 61
- Software Developers vs. Knowledge Workers: Similar or Different? ..... 63
- Summary..... 64
- Key Ideas..... 64
- References ..... 65
  
- Part III: The Context of Productivity..... 67**
  
- Chapter 8: Factors That Influence Productivity: A Checklist ..... 69**
- Introduction..... 69
- A Brief History of Productivity Factors Research ..... 70
- The List of Technical Factors..... 70
  - Product Factors ..... 70
  - Process Factors ..... 72
  - Development Environment ..... 73
- The List of Soft Factors..... 74
  - Corporate Culture ..... 74
  - Team Culture ..... 75
  - Individual Skills and Experiences ..... 76
  - Work Environment..... 78
  - Project ..... 79
- Summary..... 79
- Key Ideas..... 80
- Acknowledgments ..... 80
- Appendix: Review Design..... 80
- References..... 81
  
- Chapter 9: How Do Interruptions Affect Productivity? ..... 85**
- Introduction..... 85
- Controlled Experiments..... 86
  - What Is the Aim of an Experiment? ..... 87
  - A Typical Interruptions Experiment..... 87

How Is Disruptiveness of an Interruption Measured?.....	88
Interruptions Cause Errors.....	89
Moving Controlled Experiments Out of the Lab .....	90
Summary: Controlled Experiments .....	91
Cognitive Models.....	91
What Are Cognitive Models?.....	92
What Can Cognitive Models Predict About the Impact of Interruptions on Productivity? .....	93
Summary: Cognitive Models.....	94
Observational Studies .....	94
Observational Studies of the Workplace.....	94
Benefits and Detriments of Interruptions .....	95
Stress, Individual Differences, and Interruptions.....	96
Productivity.....	96
Strategies for Dealing with Interruptions .....	97
Summary: Observational Studies .....	97
Key Insights.....	98
Key Ideas.....	99
Acknowledgments .....	99
References.....	99
<b>Chapter 10: Happiness and the Productivity of Software Engineers .....</b>	<b>109</b>
Why the Industry Should Strive for Happy Developers .....	110
What Is Happiness, and How Do We Measure It?.....	110
Scientific Grounds of Happy and Productive Developers .....	111
How Happy Are Software Developers? .....	112
What Makes Developers Unhappy? .....	113
What Happens When Developers Are Happy (or Unhappy)? .....	114
Are Happy Developers More Productive? .....	118
Potential Impacts of Happiness on Other Outcomes.....	120
What Does the Future Hold? .....	121
Further Reading .....	121



TABLE OF CONTENTS

Key Ideas..... 122

References..... 122

**Chapter 11: Dark Agile: Perceiving People As Assets, Not Humans ..... 125**

Revisiting the Agile Manifesto..... 125

Agile in Global Outsourcing Setups..... 126

Tracking Work to Increase Productivity..... 127

Daily Stand-Up Meeting to Monitor Productivity..... 128

Stressful Work Environment..... 128

Cost of Productivity ..... 129

Open Questions for Productivity in Software Engineering ..... 131

Key Ideas..... 132

Acknowledgments ..... 132

References..... 133

**Part IV: Measuring Productivity in Practice ..... 135**

**Chapter 12: Developers’ Diverging Perceptions of Productivity ..... 137**

Quantifying Productivity: Measuring vs. Perceptions..... 137

Studying Software Developers’ Productivity Perceptions..... 138

The Cost of Context Switching ..... 139

A Productive Workday in a Developer’s Life..... 139

Developers Expect Different Measures for Quantifying Productivity ..... 140

Characterizing Software Developers by Perceptions of Productivity ..... 141

Opportunities for Improving Developer Productivity ..... 143

Key Ideas..... 145

References..... 145

**Chapter 13: Human-Centered Methods to Boost Productivity ..... 147**

Key Ideas..... 155

References..... 155

<b>Chapter 14: Using Biometric Sensors to Measure Productivity.....</b>	<b>159</b>
Operationalizing Productivity for Measurement.....	159
What the Eye Says About Focus.....	160
Observing Attention with EEG .....	161
Measuring Rumination.....	163
Moving Forward .....	164
Key Ideas.....	165
References.....	165
<b>Chapter 15: How Team Awareness Influences Perceptions of Developer Productivity .....</b>	<b>169</b>
Introduction.....	169
Awareness and Productivity.....	171
Enabling Awareness in Collaborative Software Development.....	172
Aggregating Awareness Information into Numbers .....	173
Aggregating Awareness Information into Text .....	174
Rethinking Productivity and Team Awareness .....	175
Key ideas.....	177
References.....	177
<b>Chapter 16: Software Engineering Dashboards: Types, Risks, and Future .....</b>	<b>179</b>
Introduction.....	179
Dashboards in Software Engineering.....	181
Developer Activity.....	181
Team Performance .....	182
Project Monitoring and Performance.....	183
Community Health.....	184
Summary .....	184
Risks of Using Dashboards .....	185

TABLE OF CONTENTS

Rethinking Dashboards in Software Engineering..... 188

Key Ideas..... 189

References..... 189

**Chapter 17: The COSMIC Method for Measuring the Work-Output  
Component of Productivity ..... 191**

Measurement of Functional Size ..... 192

The COSMIC Method ..... 193

Discussion of the COSMIC Model..... 195

Correlation of COSMIC Sizes with Development Effort ..... 199

Automated COSMIC Size Measurement..... 201

Conclusions..... 202

Key Ideas..... 202

References..... 203

**Chapter 18: Benchmarking: Comparing Apples to Apples..... 205**

Introduction..... 205

The Use of Standards..... 206

    Functional Size Measurement..... 206

Reasons for Benchmarking..... 208

A Standard Way of Benchmarking ..... 209

Normalizing..... 210

Sources of Benchmark Data ..... 211

    ISBSG Repository..... 211

    Internal Benchmark Data Repository..... 212

    Benchmarking in Practice ..... 212

False Incentives ..... 214

Summary..... 214

Key Ideas..... 215

Further Reading ..... 216

<b>Part V: Best Practices for Productivity .....</b>	<b>219</b>
<b>Chapter 19: Removing Software Development Waste to Improve Productivity ....</b>	<b>221</b>
Introduction.....	221
Taxonomy of Software Development Waste.....	222
Building the Wrong Feature or Product .....	223
Mismanaging the Backlog .....	224
Rework .....	225
Unnecessarily Complicated or Complex Solutions .....	226
Extraneous Cognitive Load .....	227
Psychological Distress.....	228
Knowledge Loss .....	229
Waiting/Multitasking.....	230
Ineffective Communication.....	231
Additional Wastes in Pre-agile Projects.....	232
Discussion.....	234
Not All Problems Are Wastes .....	234
Reducing Waste .....	235
Conclusion .....	238
Key Ideas.....	239
References.....	239
<b>Chapter 20: Organizational Maturity: The Elephant Affecting Productivity .....</b>	<b>241</b>
Background.....	241
The Process Maturity Framework.....	242
The Impact of Maturity on Productivity and Quality.....	245
Updating Maturity Practices for an Agile-DevOps Environment.....	246
Summary.....	248
Key Ideas.....	248
References.....	248

TABLE OF CONTENTS

- Chapter 21: Does Pair Programming Pay Off? ..... 251**
  - Introduction: Highly Productive Programming ..... 251
  - Studying Pair Programming ..... 252
  - Software Development As Knowledge Work ..... 253
  - What Actually Matters in Industrial Pair Programming ..... 254
    - Constellation A: System Knowledge Advantage..... 255
    - Constellation B: Collective System Knowledge Gap ..... 256
    - Constellation C: Complementary Knowledge..... 256
  - So, Again: Does Pair Programming Pay Off? ..... 257
  - Key Ideas..... 258
  - References ..... 258
  
- Chapter 22: Fitbit for Developers: Self-Monitoring at Work ..... 261**
  - Self-Monitoring to Quantify Our Lives..... 261
  - Self-Monitoring Software Developers' Work..... 262
  - Supporting Various Individual Needs Through Personalization ..... 264
  - Self-Reporting Increases Developers' Awareness About Efficiency..... 265
  - Retrospection About Work Increases Developers' Self-Awareness ..... 265
  - Actionable Insights Foster Productive Behavior Changes..... 266
  - Increasing Team Awareness and Solving Privacy Concerns ..... 267
  - Fostering Sustainable Behaviors at Work ..... 268
  - Key Ideas..... 269
  - References ..... 269
  
- Chapter 23: Reducing Interruptions at Work with FlowLight ..... 271**
  - The Cost of Interruptions at Work ..... 271
  - FlowLight: A Light to Indicate When to Interrupt..... 272
  - Evaluation and Benefits of FlowLight..... 273
  - Key Success Factors of FlowLight ..... 274
    - Pay Attention to Users ..... 274
    - Focus on Simplicity ..... 275

Pay Attention to Privacy Concerns.....	276
Focus on Value First, Not on Accuracy.....	276
Let Users Surprise You.....	277
Summary.....	277
Get Your Own FlowLight.....	277
Key Ideas.....	278
References.....	278
<b>Chapter 24: Enabling Productive Software Development by Improving Information Flow .....</b>	<b>281</b>
Mylyn: Improving Information Flow for the Individual Software Developer .....	282
Tasktop Sync: Improving Information Flow for the Development Team .....	285
Tasktop Integration Hub: Improving Information Flow for a Software Development Organization.....	288
Takeaways .....	290
Key Ideas.....	291
References.....	291
<b>Chapter 25: Mindfulness as a Potential Tool for Productivity.....</b>	<b>293</b>
A Definition of Mindfulness.....	293
Mindfulness for Productivity? .....	294
Cognitive Benefits of Mindfulness .....	295
Mindfulness and Emotional Intelligence .....	296
Pitfalls of Mindfulness .....	297
Mindfulness Breaks .....	298
Conclusion .....	299
Key Ideas.....	300
References.....	300
<b>Index.....</b>	<b>303</b>

# 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

## INTRODUCTION

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.



**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 Jaspán 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 Vogezang 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 mindset 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>

## INTRODUCTION

- [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.