

Equity, Diversity, and Inclusion in Software Engineering

Best Practices and Insights

Edited by

Daniela Damian · Kelly Blincoe
Denae Ford · Alexander Serebrenik
Zainab Masood

Apress
open

Equity, Diversity, and Inclusion in Software Engineering

Best Practices and Insights

**Daniela Damian
Kelly Blincoe
Denaе Ford
Alexander Serebrenik
Zainab Masood**

Apress
open

Equity, Diversity, and Inclusion in Software Engineering: Best Practices and Insights

Daniela Damian
Victoria, BC, Canada

Denae Ford
Redmond, WA, USA

Zainab Masood
Riyadh, Kingdom of Saudi Arabia

ISBN-13 (pbk): 978-1-4842-9650-9
<https://doi.org/10.1007/978-1-4842-9651-6>

Kelly Blincoe
Auckland, New Zealand

Alexander Serebrenik
Eindhoven, Noord-Brabant, The Netherlands

Copyright © 2024 by Daniela Damian, Kelly Blincoe, Denae Ford, Alexander Serebrenik, Zainab Masood

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 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

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.

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: Laura Berendson
Coordinating Editor: Jessica Vakili

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 booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

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 <https://www.apress.com/gp/services/source-code>.

If disposing of this product, please recycle the paper

Funders

ACM Special Interest Group on Software Engineering (SIGSOFT)

Google

*Human Aspects of Software Engineering Lab,
University of Auckland, New Zealand*

Barbora Buhnova, Masaryk University, Czech Republic

Daniela Damian, University of Victoria, Canada

John Grundy, Monash University, Australia

Lucia Happe, Karlsruhe Institute of Technology, Germany

Reed M. Milewicz, Sandia National Laboratories, USA

*Birgit Penzenstadler, Chalmers University of
Technology / Gothenburg University, Sweden, and Lappeenranta
University of Technology, Finland*

*Mary Sánchez-Gordón, Østfold University College,
Norway and EduTech Erasmus+ Project
(609785-EPP-1-2019-1-ES-EPPKA2-CBHE-JP)*

*Alexander Serebrenik, Eindhoven University of Technology,
The Netherlands*

*Cecilia Bastarrica, Nancy Hitschfeld-Kahler and Jocelyn Simmonds,
Universidad de Chile, Chile*

Table of Contents

About the Authors.....	ix
Introduction	xi
Part I: Landscape of Diversity and Inclusion Studies	1
Chapter 1: Roads Ahead to Diversity and Inclusion by Software Engineering	3
Chapter 2: ED&I and SE: Challenges, Progress, and Lessons	17
Chapter 3: The Challenges of Ethnic-Racial Diversity Within the IT Sector	37
Chapter 4: Breaking the Glass Floor for Women in Tech	55
Part II: Inclusive Software: How Inclusion Impacts Products	67
Chapter 5: How Users Perceive the Representation of Non-binary Gender in Software Systems: An Interview Study	69
Chapter 6: Elicitation Revisited for More Inclusive Requirements Engineering	91
Chapter 7: Developers' Perspective of Diverse End User Requirements	105
Chapter 8: UI Development Experiences of Programmers with Visual Impairments in Product Teams	121
Chapter 9: The Role of Ethics in Engineering Fair AI (and Beyond)	135
Chapter 10: Beyond Diversity: Computing for Inclusive Software.....	151
Part III: Diversity and Inclusion in Development Teams	167
Chapter 11: Gender Diversity on Software Development Teams: A Qualitative Study	169
Chapter 12: Exploring Intersectional Perspectives in Software Engineering Through Narratives.....	185

TABLE OF CONTENTS

Chapter 13: Perceptions of Software Developer Inclusion: A Survey at Google ...	207
Chapter 14: How Much Do Women Build Open Source Infrastructure?.....	231
Part IV: Across the Gamut of Opportunities: Initiatives and Interventions	255
Chapter 15: Beyond Classroom: Making a Difference in Diversity in Tech	257
Chapter 16: Toward More Gender-Inclusive Game Jams and Hackathons	275
Chapter 17: Codes of Conduct in Open Source	295
Chapter 18: Did Gerrit's Respectful Code Review Reminders Reduce Comment Toxicity?	309
Chapter 19: Experiences Implementing and Deploying Anonymous Code Review	323
Chapter 20: Mentorship of Women in OSS Projects: A Cross-Disciplinary, Integrative Review.....	337
Chapter 21: Bringing Diversity in Software Engineering Education from the Middle East and Africa	365
Chapter 22: Open Sourcing Diversity, Equity, and Inclusion.....	385
Part V: Challenges and Initiatives to Designing Inclusive Software Engineering Education Environments.....	397
Chapter 23: Rethinking Gender Diversity and Inclusion Initiatives for CS and SE in a University Setting	399
Chapter 24: Economical Accommodations for Neurodivergent Students in Software Engineering Education: Experiences from an Intervention in Four Undergraduate Courses.....	413
Chapter 25: Effective Interventions to Promote Diversity in CS Classroom	429
Chapter 26: Software Engineering Through Community-Engaged Learning and an Inclusive Network.....	449

TABLE OF CONTENTS

Part VI: Methodologies Supporting Studies of Diversity and Inclusion in Software Engineering	467
Chapter 27: How to Measure Diversity Actionably in Technology	469
Chapter 28: How to Ask About Gender Identity of Software Engineers and “Guess” It from the Archival Data	487
Chapter 29: Strategies for Reporting and Centering Marginalized Developer Experiences	507
Index.....	523

About the Authors

Dr. Daniela Damian is Professor of Software Engineering, the ECS-CAPI Chair in Inclusive Science, Technology and Engineering at the University of Victoria, BC, Canada, and a D.L. Parnas Fellow. She studies human and collaborative aspects of software engineering, with a recent focus on diversity and inclusion. Her teaching approach is experiential-based and in close connection with the software practice. Integrating SE research and teaching, her goal is to create a learning environment that prepares students for future real-world software engineering that embraces diversity and takes place in collaborative, inclusive international and multicultural teams.

Dr. Kelly Blincoe is an Associate Professor of Software Engineering at the University of Auckland, New Zealand, in the Department of Electrical, Computer, and Software Engineering and a Rutherford Discovery Fellow. She leads the Human Aspects of Software Engineering Lab (HASEL). Her research is mainly in the human and social aspects of software engineering. Current research topics include software dependencies, software ecosystems, collaborative software development, software requirements engineering, and software developer diversity and inclusion.

Dr. Denae Ford Robinson is Senior Researcher at Microsoft Research in the SAINTes group and Affiliate Assistant Professor in the Human Centered Design and Engineering Department at the University of Washington. Her research lies at the intersection of human-computer interaction and software engineering. In her work she identifies and dismantles cognitive and social barriers by designing mechanisms to support software developer participation in online socio-technical ecosystems. She is best known for her research on just-in-time mentorship as a mode to empower welcoming engagement in collaborative Q&A for online programming communities including open source software and work to empower marginalized software developers in online communities.

ABOUT THE AUTHORS

Dr. Alexander Serebrenik is Professor of Social Software Engineering at Eindhoven University of Technology (TU/e), the Netherlands. His research goal is to facilitate evolution of software by taking into account social aspects of software development. His work tends to involve theories and methods both from within computer science (e.g., theory of socio-technical coordination; methods from natural language processing, machine learning) and from outside of computer science (e.g., organisational psychology). The underlying idea of his work is that of empiricism, i.e., that addressing software engineering challenges should be grounded in observation and experimentation, and requires a combination of the social and the technical perspectives.

Dr. Zainab Masood is an Assistant Professor of Software Engineering at the Prince Sultan University, Saudi Arabia. Her research interests include secure software development, software development methodologies and practices, and human and socio-technical aspects of software engineering.

Introduction

Creating an inclusive environment where all developers regardless of their identity can feel welcome and exploit their talents is an ethical imperative no company can ignore. With its broad and significant impact on our society, software and its design also must be inclusive of an increasingly diverse end user base and their preferred software usage patterns. It is no surprise that the differences between the people developing software are reflected in how software is developed and what the resulting software looks like. The characteristics of and support for inclusive development teams and inclusive software and the intricate interdependency between the two are emerging as one of the more critical and pressing needs in software research and practice.

Indeed, software organizations have in the last decade been trying to make changes for a more diverse and inclusive software development environment. The push for increased diversity in software has been a public one, from annual diversity reports by some of the world's most visible companies such as Microsoft,¹ Google,² and Facebook³ to large projects such as the Linux Foundation's Software Developer Diversity and Inclusion project [1] that explores, evaluates, and promotes best practices from research and industry to increase diversity and inclusion in software engineering.

At the same time, the scientific literature provides clear evidence that diverse teams are more creative and high-performing [2, 3, 4] and that the software industry still has a diversity problem. Research has shown that diversity is an essential feature of a team – without gender diversity, teams may focus on doing things faster and less on doing new things [5, 6], while teams that benefit from race or nationality diversity also leverage multiple points of view, availability of knowledge and skills, and constructive conflict [7]. However, diversity comes with an increased cost of communication, and successful mechanisms for creating diverse and inclusive software development environments are still to be better studied.

¹www.microsoft.com/en-us/diversity/inside-microsoft/annual-report

²<https://diversity.google/annual-report/>

³<https://diversity.fb.com/read-report/>

INTRODUCTION

Despite these many efforts, diversity remains low. Numbers from the software industry show that less than 25% of software engineers are women. Diversity is also low in regard to many other facets of diversity. For example, in the United States, less than 5% of software engineers are Black and, in New Zealand, only 8% are Māori [8]. Men in same-sex relationships are 12 percentage points less likely to have completed a bachelor's degree in a STEM field compared with men in different-sex relationships [9]. Significant research shows that developers from marginalized groups such as women, racial and ethnic minorities [10, 11], or non-native English speakers face bias [12] and discrimination [13].

While diversity can be described along various dimensions including gender identity, ethnicity, race, age, neurodiversity, or sexual orientation, most of the debate has centered around gender diversity, and gender has usually been interpreted as binary. This book expands beyond this view and considers a range of diversity facets and their impact on software engineering. It also considers intersectionality between the different facets of diversity, since people in intersecting, historically excluded groups will have unique experiences and needs [10, 14].

To the best of our knowledge, this book is one of the first collections of current works on this topic – there are no books explicitly targeting diversity in the context of software engineering to date. A major related work is the *Encyclopedia of Gender and Information Technology* edited by Eileen Trauth, but it covers both a much broader area, information technology rather than software engineering, and a much narrower one, focusing on gender and predominantly on women [15]. Inspired by the aforementioned work, this book provides an updated perspective of the range of diversity and interventions of equity and inclusion in software engineering.

Overall, the book provides an overview of research into the different aspects of diversity and inclusion in software engineering, as well as the tools, methods, and practices proposed to foster diversity, to build inclusive software teams and development environments, as well as inclusive software. It also describes the research challenges and possible methodologies in studying diversity and inclusion in software. For *researchers*, the book presents a state-of-the-art collection of existing studies into many aspects of diversity, methods and tools proposed and tried out in practice, challenges in research, and methodologies supporting studies of diversity and inclusion in software engineering and contributes to a research agenda on this topic for future studies. For *industry practitioners*, the book describes efforts to investigate diversity in

software teams, whether in corporate or open source environments. It also describes empirical evidence about effectiveness of certain methods and approaches to foster diversity and inclusion in software development. For *educators*, the book describes practices and effective changes in computer science/software engineering curricula that were found as effective in engaging learners from marginalized groups and creating inclusive software teams that are diverse and with a heightened ability to develop inclusive software and that relate to educational material useful for training for diversity, equity, and inclusion.

Landscape of Diversity and Inclusion Studies

The space of diversity and inclusion studies is, not surprisingly, diverse: one might consider multiple diversity axes (e.g., gender, age, ethnicity) or their combinations (as individuals located at the intersection of multiple diversity axes do not necessarily know whether their experiences should be attributed to either of the axes or the interplay between them); multiple social, socio-technical, and technical perspectives (e.g., how does team diversity affect communication between the developers and their software, how different stakeholders respond to software changes, or what are gender- and/or age-related biases in the software itself); as well as different tasks, from identifying the problems through proposing solutions to evaluating the solutions and implementing them in software practice.

We start this book with the first part aiming to chart a broad picture of software engineering diversity and inclusion studies. Each one of the chapters constituting this part covers multiple studies related to diversity and inclusion, for example, in Chapter 3, “The Challenges of Ethnic-Racial Diversity Within the IT Sector,” Michele Miranda and Rafael Prikladnicki survey the scientific literature on the topic, while Chapter 4, “Breaking the Glass Floor for Women in Tech,” by Trinkenreich et al. focuses on challenges experienced by women both in companies and in open source projects as well as actions one can take to overcome those. As opposed to these two studies, Chapters 1 and 2, “Roads Ahead to Diversity and Inclusion by Software Engineering” by Rastogi and “ED&I and SE: Challenges, Progress, and Lessons” by Grundy et al., do not focus on specific subdomains and provide a global overview of their work in this domain, challenges, progress made, and lessons learned.

Inclusive Software

Lack of diversity and representation in the teams that create and maintain software products can have impacts on the products these teams create. If software products are to be usable by a wide range of diverse people, they must be designed with inclusivity in mind. Having more diverse perspectives involved in the creation of software can help in this regard, but for this to happen, the way we create software needs to be inclusive. In the next part, the chapters focus on the inclusivity of software products themselves and examine the inclusivity of the process of developing software. Hashmati and Penzenstadler, in Chapter 5, “How Users Perceive the Representation of Non-binary Gender in Software Systems: An Interview Study,” discuss how to make software systems more gender-inclusive for the full gender spectrum. Tizard et al. motivate the need for and provide recommendations for more inclusive software requirements elicitation to understand more diverse user needs in Chapter 6, “Elicitation Revisited for More Inclusive Requirements Engineering.” In Chapter 7, “Developers’ Perspective of Diverse End User Requirements,” Grundy et al. examine current practices, software developer perceptions, and challenges of considering diverse user needs. Pandey et al. make recommendations, in Chapter 8, “UI Development Experiences of Programmers with Visual Impairments in Product Teams,” for more accessible software tools from their studies of teams including software developers with visual impairments. Johnson and Smith discuss the importance of considering ethics in the software development process and make recommendations for ensuring ethical decision making in Chapter 9, “The Role of Ethics in Engineering Fair AI (and Beyond).” Finally, Devathasan et al. reflect, in Chapter 10, “Beyond Diversity: Computing for Inclusive Software,” on the ability of students to build inclusive software products by including activities such as empathy-based requirements gathering techniques.

Diversity and Inclusion in Development Teams

Given the surge in research and corporate efforts to study and increase the participation of women and marginalized groups in software development, how diverse and inclusive teams *really* are? What are the perceptions of developers about diversity, its benefits, or about how inclusive their teams are in the development work? While a growing body of research brings evidence about the benefits of diverse teams, with gender-based diversity being most studied, insights into inclusiveness of software teams are still limited.

In this next part, chapters report from studies of perceived diversity and inclusion in development teams in corporate environments, as well as open source projects, or in the industry in general. Hedges and Murphy-Hill (Chapter 13, “Perceptions of Software Developer Inclusion: A Survey at Google”) describe insights from a developer survey at Google that investigated the sentiment around inclusion in development projects as related to experiences in help-seeking or other collaborative behaviors in teams. Kohl and Prikladnicki (Chapter 11, “Gender Diversity on Software Development Teams: A Qualitative Study”) report on perceptions on gender diversity in software engineering, from a more general study, their findings suggesting that gender-diverse workspaces relate to better ideas sharing, better decision making, and creativity and innovation. Similarly, the chapter co-authored by Qiu and colleagues (Chapter 14, “How Much Do Women Build Open Source Infrastructure?”) investigates the representativeness of women in open source development and in particular infrastructure projects, reporting an upward trend in the percentage of women among both highly active (core) and the general repository contributors. Finally, Sánchez-Gordón and Colomo-Palacios (Chapter 12, “Exploring Intersectional Perspectives in Software Engineering Through Narratives”) discuss the largely understudied aspect of intersectionality in software engineering and use narratives collected through ethnographic histories from software practitioners from underrepresented groups to draw attention to what they refer to as “multiple interlocking issues” related to diversity in software development.

Across the Gamut of Opportunities: Initiatives and Interventions

In the world of constant critiques of the low diversity in software engineering settings, it is often quite rare that research can provide a fresh take and showcase intervention and solutions to combat the issue. With this book, we want to showcase that there is room for both the outlined critique and approaches that can be adapted to a range of environments.

In this part, authors cover a range of topics that highlight that we can do more than just study the challenges of diversity, but that we have techniques to help construct, design, and build that can improve diversity in software engineering. This part begins with a chapter by Barbora Buhnova (Chapter 15, “Beyond Classroom: Making a Difference in Diversity in Tech”) on learning about how projects like “Czechitas” have triggered a social movement for ushering in and retaining over 50,000+ women in

INTRODUCTION

Czechia. Gama and colleagues (Chapter 16, “Toward More Gender-Inclusive Game Jams and Hackathons”) also cover how intentional use of in-person experiences of hackathons and game jams can be more inclusive across women and LGBTQ+. In the global landscape of development, Samarah and colleagues (Chapter 21, “Bringing Diversity in Software Engineering Education from the Middle East and Africa”) cover the state of the challenges of engaging software engineering to the Middle East and Africa. Dicker and colleagues (Chapter 19, “Experiences Implementing and Deploying Anonymous Code Review”) compare settings of industrial code review and how developers experience pushback and share results from tools they built to support an anonymous code review process. Likewise, Murphy-Hill and colleagues (Chapter 18, “Did Gerrit’s Respectful Code Review Reminders Reduce Comment Toxicity?”) provide results at reducing code review toxicity in industry settings. This part also covers a range of experiences in OSS from understanding how to enforce codes of conduct by Frluckaj and Howison (Chapter 17, “Codes of Conduct in Open Source”) to recommendation for successful mentoring strategies to onboard new contributors by Jacobs and colleagues (Chapter 20, “Mentorship of Women in OSS Projects: A Cross-Disciplinary, Integrative Review”), and finally Demetris Cheatham (Chapter 22, “Open Sourcing Diversity, Equity, and Inclusion”) outlines how companies like GitHub can use their power to influence equity, diversity, and inclusion by forming partnerships with the organizations and individuals on the ground doing the meaningful work that makes a difference.

Challenges and Initiatives to Designing Inclusive Software Engineering Education Environments

Investing in the study and preparedness of the future generation of software engineers may prove effective in further understanding the barriers of diversity and inclusion but also in trying out some strategies to overcome them in education environments. In addition to supporting software engineers on their career paths once in the software industry, efforts could be made to develop learning environments that are inclusive of all students’ needs as well as train them with a heightened awareness and regard for the benefits of diverse teams and ways in which diverse teams can be inclusive of their members’ unique contributions.

This part starts off with Lucia Happe’s (Chapter 25), “Effective Interventions to Promote Diversity in CS Classroom,” that reviews some of the reasons for which women engage with computer science (CS) as well as those for why they would drop out of

CS education and offers some insight into some interventions perceived as effective in targeting and mitigating frustrations in CS education. With a more Latin American focus, Chapter 23, “Rethinking Gender Diversity and Inclusion Initiatives for CS and SE in a University Setting,” by Jocelyn Simmonds and colleagues discusses gender-related initiatives in Chile and brings recommendations for improvement from a qualitative study of the experiences and perceptions of computer science students, staff, and faculty at the University of Chile. The other two chapters describe specific interventions to design inclusive learning environments. In Chapter 26, “Software Engineering Through Community Engaged Learning and an Inclusive Network,” Nawar Arony and colleagues describe the design of and experiences from the INSPIRE: STEM for Social Impact program at the University of Victoria, Canada, where students, in particular women and other underrepresented groups, are empowered to learn and use software and other engineering solutions in approaching sustainability-focused, community-driven problems and in a network of academic and industry mentors. Finally, Liebel and Sigurðardóttir in Chapter 24, “Economical Accommodations for Neurodivergent Students in Software Engineering Education: Experiences from an Intervention in Four Undergraduate Courses,” discuss the obstacles as well as the accommodations and interventions for neurodivergent students they found successful in a number of courses in the undergraduate computer science program at Reykjavik University, Iceland.

Methodologies Supporting Studies of Diversity and Inclusion in Software Engineering

Studying diversity requires utmost care: while methodological rigor is a must for any scientific publication, less rigorous studies targeting individuals from marginalized groups might lead to conclusions harming those individuals or reinforcing pre-existing stereotypes. Moreover, increased societal attention to diversity and inclusion means that studies of these topics are conducted beyond the traditional academic circuit. For example, Stack Overflow⁴ and GitHub⁵ publish yearly reports about developer communities referring to such diversity attributes as gender and location; TokyoDev

⁴<https://survey.stackoverflow.co/2022/>

⁵<https://octoverse.github.com/>

INTRODUCTION

has surveyed 558 international software engineers living in Japan;⁶ Andela studied the importance of team culture and giving back by surveying software developers.⁷

This is why three chapters constituting the last part of the book touch upon topics related to conducting studies of diversity and inclusion in the software engineering realm. Chapter 27, “How to Measure Diversity Actionably in Technology,” by Hamid et al. focuses on the survey instrument to measure diversity in terms of GenderMag personas as well as provides a strategy for validating software engineering surveys in general. In Chapter 28, “How to Ask About Gender Identity of Software Engineers and ‘Guess’ It from the Archival Data,” Serebrenik reviews ways of obtaining information about an individual’s gender identity that often constitute a prerequisite for diversity and inclusion studies. Finally, Chapter 29, “Strategies for Reporting and Centering Marginalized Developer Experiences,” by Ford and Johnson focuses on the interaction between researchers and developers from historically marginalized communities. The chapter provides approaches on how to center research on marginalized perspectives as well as illustrates these approaches by case studies the authors conducted working with blind and low-vision software developers as well as Black and African-American technologists.

We hope that these chapters can benefit both practitioners and academics active in the diversity and inclusion studies of software developers.

Bibliography

- [1] www.linuxfoundation.org/press-release/linux-foundation-focuses-on-science-and-research-to-advance-diversity-and-inclusion-in-software-engineering
- [2] Rodríguez-Pérez, G., Nadri, R., & Nagappan, M. (2021). Perceived diversity in software engineering: a systematic literature review. *Empirical Software Engineering*, 26(5), 1-38.

⁶www.tokyodev.com/insights/2022-developer-survey/

⁷<https://andela.com/insights/software-developer-survey-the-importance-of-culture-and-giving-back/>

- [3] Østergaard, C. R., Timmermans, B., & Kristinsson, K. (2011). Does a different view create something new? The effect of employee diversity on innovation. *Research Policy*, 40(3), 500–509.
- [4] Vasilescu, B., Posnett, D., Ray, B., van den Brand, M. G., Serebrenik, A., Devanbu, P., & Filkov, V. (2015, April). Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 3789–3798).
- [5] Page Scott, E. (2007). How the power of diversity creates better groups, firms, schools and societies. Princeton.
- [6] Albusays, K., Bjorn, P., Dabbish, L., Ford, D., Murphy-Hill, E., Serebrenik, A., & Storey, M. A. (2021). The diversity crisis in software development. *IEEE Software*, 38(2), 19–25.
- [7] Tourani, P., Adams, B., & Serebrenik, A. (2017). Code of conduct in open source projects. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 24–33). IEEE.
- [8] Gruman, G. (2020). IT snapshot: Ethnic diversity in the tech industry. Computer World. Available at www.computerworld.com/article/3567095/it-snapshot-ethnic-diversity-in-the-tech-industry.html
- [9] Sansone, D. & Carpenter, C. S. (2020). Turing's children: Representation of sexual minorities in STEM. *PloS One*, 15(11), e0241596.
- [10] Ross, M., Hazari, Z., Sonnert, G., & Sadler, P. (2020). The intersection of being black and being a woman: Examining the effect of social computing relationships on computer science career choice. *ACM Transactions on Computing Education (TOCE)*, 20(2), 1–15.

INTRODUCTION

- [11] Prana, G. A. A., Ford, D., Rastogi, A., Lo, D., Purandare, R., & Nagappan, N. (2020). Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in OSS. *IEEE Transactions on Software Engineering (TSE)*, doi: 10.1109/TSE.2021.3092813, arXiv preprint at arXiv:2010.00822.
- [12] Davidson, J. L., Naik, R., Mannan, U. A., Azarbakht, A., & Jensen, C. (2014). On older adults in free/open source software: reflections of contributors and community leaders. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 93–100). IEEE.
- [13] Blincoe, K., Springer, O., & Wrobel, M. R. (2019). Perceptions of Gender Diversity's impact on mood in software development teams. *IEEE Software*, 36(5), 51–56.
- [14] Charleston, L. J., George, P. L., Jackson, J. F., Berhanu, J., & Amechi, M. H. (2014). Navigating underrepresented STEM spaces: Experiences of Black women in US computing science higher education programs who actualize success. *Journal of Diversity in Higher Education*, 7(3), 166.
- [15] Trauth, E. M. (ed.). (2006). *Encyclopedia of Gender and Information Technology*. IGI Global.

Book Editors

Information about the book authors/editors and links to their web pages are listed here:

- **Daniela Damian:** <https://danieladamian.ca>
- **Kelly Blincoe:** <https://kblincoe.github.io/>
- **Denae Ford:** <https://denaeford.me/>
- **Alexander Serebrenik:** www.win.tue.nl/~aserebre/
- **Zainab Masood:** <https://zmasood.github.io/>

PART I

Landscape of Diversity and Inclusion Studies

CHAPTER 1

Roads Ahead to Diversity and Inclusion by Software Engineering

Ayushi Rastogi, University of Groningen, The Netherlands.*

Software has a pivotal role in human lives. It is instrumental in transforming dreams into reality, such as the role of software in self-driving cars and space explorations. Software is also fueling our worst nightmares. Channeling the spread of misinformation, inciting communal violence, and making life-threatening decisions for humankind are examples of how software harms society. In the twenty-first century, software is the most potent tool with humankind that nurtures and ruptures the social fabric of modern human civilization.

Software is a creation and a reflection of humans. From inception to implementation and maintenance, an individual or a group works and makes decisions on software. This process infuses the technical details of software development with the experiences and worldviews of the individuals and groups involved in its development. The resulting software is not just for the people and by the people but also a reflection of the people involved in its development.

Software is made with various intentions and serves a range of purposes. Even with the best intentions, software can unintendedly harm humankind, especially marginalized communities.

One such example is a face recognition system. Face recognition systems help humankind automatically process people's information, individually and as a crowd. These systems, however, are shown to harm marginalized communities (e.g., black population¹) by systematic misclassification, placing them in a disadvantaged position.

Explorations on diversity and inclusion focus on appropriately handling the distinguishing characteristics that systematically place an individual or group(s) in a disadvantaged position. These characteristics can be visible and perceived (e.g., gender [21], age [10], nationality [18], and race [15]) and subtle and not visible otherwise (e.g., beliefs and background [9]) [20]. In simpler terms, promoting diversity starts with understanding when to distinguish and when not to. For instance, one way to promote diversity is to not distinguish people on gender for job application. In another scenario, promoting diversity means creating special provisions for the visually challenged. Inclusion goes further into making everyone feel included.

This chapter offers the author's perspective on promoting diversity and inclusion and discusses the role of software engineering research. The chapter introduces the unique position of the software industry in exaggerating diversity and inclusion issues and the unique opportunity to solve a problem of historical relevance. The author describes the two paths research can pursue, at the software and human levels, to mitigate diversity and inclusion issues and the scope of each exploration. Finally, the chapter characterizes the software engineering research focused on diversity and inclusion (with examples) and proposes a roadmap for improvement.

This chapter is written for various audiences. Software engineering research can pursue the two paths presented in this chapter to promote diversity and inclusion. The software industry (including software-defined enterprises) can use it to overview the current state of research and potential solution spaces. For the broad computer science audience and other fields interested in exploring diversity and inclusion, this chapter introduces the unique position of software systems in exacerbating diversity and inclusion issues and providing a unique solution space. While there are many paths to explore diversity and inclusion issues (e.g., policy-making bodies and government), the insights presented in this chapter are limited to what we know from the scientific literature in software engineering research. For the interested, it will be worthwhile exploring the directions mentioned previously.

¹ www.wired.com/story/best-algorithms-struggle-recognize-black-faces-equally/

A Unique Opportunity

Software engineering is in a unique position regarding diversity and inclusion. Software, on the one hand, is exaggerating diversity and inclusion issues at an unprecedented pace. On the other hand, it offers a unique solution space that never existed earlier.

Diversity and inclusion issues exist historically. Some early documented references to diversity and inclusion issues date to the 1960s [24], although the issues have existed for much longer. These issues have percolated in society for a long time and have many manifestations. For instance, Chapter 6, “Elicitation Revisited for More Inclusive Requirements Engineering,” describes diversity and inclusion issues in requirements engineering and offers advice for more inclusive requirements engineering. Another manifestation is during software development, where there is a systematic difference in evaluation depending on the gender [21] and geographical location [18] of code contributors. Part 3 of this book offers detailed insights into diversity and inclusion in development teams. Further, some issues affect a specific subpopulation (e.g., a region). In contrast, others can involve a wider population (e.g., multiple countries) [17].

With software, diversity and inclusion issues are widespread and bigger. As reflections of people, software imbibes people’s characteristics and cues that introduce biases and fairness issues. For example, a recent study shows that setting gender to “female” resulted in fewer high-paying job ads than setting the gender to “male” [4]. A similar exploration in software engineering research showed that the chances of women’s contributions being accepted are higher when their gender identity is unknown [21]. The software picks cues from its environment as well. With machine learning software, the software picks cues and amplifies what it learns, which worsens existing issues. Software systems can now inflict more harm than they would have had in the past.

The software also offers an unprecedented opportunity and a solution space to mitigate this problem of historical relevance. This unique position comes from the characteristics of software (as opposed to humans), the role of software as an intermediary in our daily lives, and the historical data generated during its development and use.

Software offers a way to enforce rules where humans fail and subconsciously introduce biases [13]. The second opportunity comes from the position of software in modern human civilization. As the invisible air around us, the software is everywhere, giving it the unique opportunity and capability to influence communications, beliefs, experience, and more. If software’s position is leveraged wisely, it can contribute to reintroducing lost values.

Software engineering offers another opportunity in the form of data to investigate and reflect on software and its development. During software development, traces of development activities are captured. Examples are version control systems, mailing lists, and issue-tracking systems. For decades, these logs with rich insights on activities, actors, and events have helped us improve software and its development [11, 19]. These traces have recently shown us an evidence-based path to investigate diversity and inclusion issues, including measuring the extent of the problem [17, 18, 21] and potential solution spaces [17]. For more initiatives in practice and education, refer to Parts 4 and 5 in the book.

This opportunity is unprecedented in history for various reasons. We now have data on actions and behavior contributing to diversity and inclusion issues. For the first time, we can study the decisions behind closed doors. The qualitative analysis helped us understand perceptions and experiences. Now with the data, we can substantiate the claims. Software data can help us go beyond perceptions, which might sometimes not reflect reality. Unlike experiments, which are yet another powerful tool to study behavior, to some extent, data mitigates the changes in behavior resulting from observation – also referred to as the Hawthorne effect.

Today, data-driven explorations promise to offer objective insights as long as the data reasonably reflect actual events. Analyzing activity log data in its current form offers a space where the phenomenon seen is closer to the actual event. However, it remains a valid risk that once these data sources are used for analysis, it changes the actions and behavior of the people. Another challenge is the nonavailability or limited availability of data. Given the topic's sensitive nature, there is limited to no data to study diversity and inclusion.

Two Solution Spaces

Software and humans are in a vicious cycle in which they constantly learn and influence each other. To break this cycle, changes are required at the level of software and people, the two solution spaces to promote diversity and inclusion. Changes at either of the two levels will influence one another and create a cycle of changes promoting diversity and inclusion. For example, a recommender system that offers equal job opportunities to people from different demographics will foster diversity. In the long run, this is likely to change marginalized communities' social status and contribute to changing the prevalent notions responsible for biases.

Making software systems fairer can start at any stage of software development. For software systems yet to be developed, discussions on fairness should start at the inception of an idea, leading its way into its design, implementation, testing, and use. For software systems in use, it is still necessary to gauge how the outcomes of a software system influence people.

During ideation, fairness refers to the awareness of audiences and their various needs. The better we understand the needs, the more prepared we are to propose a solution that likely works for all. However, there remains an unforeseeable future that is hard to quantify. For example, creating software for a visually challenged audience requires understanding their needs. One such need is comprehending an image. See Chapter 6, “Elicitation Revisited for More Inclusive Requirements Engineering,” for insights on how requirements can be made more inclusive.

Fairness in design choices is about balancing the stakes by making trade-offs among competing and often conflicting priorities. In the preceding example, to design a software system for a visually abled and disabled audience, there is a need to consider presenting the same information in different formats. For example, offering an alternative text for an image makes the content of an image comprehensible to a visually disabled person.

During development, the individual and group experiences and decisions are codified in source code, potentially introducing fairness issues. These issues can come through three channels: (a) algorithms making decisions, (b) machine learning components as abstract algorithms, and (c) the data from or to which it is learned or applied.

Algorithms as decision-makers, unless designed to meet the requirements of any, are likely to miss the needs of many. These differences emerge as bias against subpopulations. Machine learning components are a more complex alternative to algorithms in making decisions, and often it is unclear what rules they follow. Sometimes issues creep in from the data it is trained on and, otherwise, the data it is applied to. Even when an algorithm is unbiased, it can introduce unfairness if the data it is applied to has inherent biases.

Testing for fairness comes into play when a system is ready for use or in use. Testing for fairness looks into attributes (e.g., gender and ethnicity – also referred to as sensitive or protected attributes) to assess how software systems behave for one or more protected attributes [5]. Finally, who has access to software or its feature is another source contributing to fairness issues during deployment.

Another way to break the vicious cycle starts with the people involved in its development. As the driving force, humans have a significant role in shaping software. Therefore, with diversity considerations, it is likely to create software that caters to the needs of many. This starts with creating and promoting a fair development environment (including its process) and a diverse and inclusive workforce.

People are shaping software in various capacities. Some of these roles might be more visible, for example, developer and tester. Other roles might be more subtle, such as user and business requirements analysts shaping the features implemented in software. As a user, people consciously or subconsciously drive the need for software or its features. An architect envisions what the software may look like, and developers bring the vision to reality. There are more roles than the ones mentioned here. Collectively, these individual and group dynamics, if meaningfully controlled, can improve the software system and spearhead solving the fairness issues at its source.

The two solution spaces have their utility and purpose. Fixing software systems for fairness is a reasonable short-term goal, although it has its share of issues. First, fixing fairness issues is challenging in the current landscape of somewhat homogenous teams. It is unreasonable to expect teams with a limited understanding of the problem to devise a solution that works for all. The meaningfulness of such a solution needs to be investigated.

The other alternative is making participating stakeholders in software development diverse and devising initiatives and processes that foster inclusion. This solution space is even more challenging since we are discussing issues that have percolated for centuries and are now deep-rooted. Any attempts at solving the problem from either end will go a long way in mitigating diversity and inclusion issues.

Studies in Software Engineering

Explorations on diversity and inclusion in software engineering can be best described based on (a) the objective of a study, (b) the context, (c) the characteristics of stakeholders, and (d) the choice of data and method for exploration.

Objective: The objective of a study can be (a) to identify a problem, (b) to characterize the state of practice, and (c) to propose or adopt solutions. Today, some studies identify and report a problem relating to diversity and inclusion. For instance, a study by Terrell et al. shows evidence of gender bias by eliminating other potential explanations for an observation [21]. More similar studies offer insights pointing to diversity issues (e.g., geographic disparity in code evaluation [18]).

Studies characterize the state of practice to describe how things work in practice. This includes investigations into interactions among diversity and inclusion goals and software development activity, process, and outcomes. These explorations form the basis for understanding problem space and exploring appropriate solutions. For instance, a study on gender diversity showed that gender diversity correlates to improved productivity [23].

The solution space often takes inspiration from other fields for ideas likely to work in software engineering, for instance, anonymized peer reviews as a potential solution for mitigating biases in software engineering [14]. Another example is the GenderMag tool designed to identify diversity issues in software [3].

Context: The context for explorations on diversity and inclusion can be characterized as open source, industry-sponsored open source software systems, commercial software systems, and software-defined enterprises. Other contextual factors are the phase of software development and specific development activity. Here, the phase of software development refers to requirements elicitation, design, development, testing, and maintenance [20]. In each of these software development phases, activities such as code review [18], debugging [8], and pair programming [7] are studied to investigate diversity and inclusion issues.

Stakeholders: Each exploration addresses the problems or needs of one or more stakeholders. A stakeholder is characterized by features (visible or invisible) that uniquely define an individual or a group. These features can be identifiable, cognitive, changing over time, and role-based. Identifiable features generally refer to gender [21], ethnicity [15], and culture [1]. A particular class of identifiable characteristics includes specially abled individuals, for example, visually challenged [12]. Cognitive features include aspects such as ethics [2] and personality [6]. Features that change with time are age [10], experience [10], and status as a newcomer [16]. Finally, people in various roles include users, developers, and managers.

Data and methods: There are quantitative explorations based on traces of development activities inferred from archival data and its associated metadata (e.g., [19]). There are qualitative explorations (in the form of surveys and interviews) soliciting the perceptions and experiences of participating stakeholders (e.g., [22]). Other explorations follow a mixed-methods approach, use ethnography, or conduct experiments [20]. Ultimately, these solution spaces generate awareness and offer recommendations and tools that help identify and understand a problem and propose solutions.

The Road Ahead

Data: To understand diversity and inclusion issues, we need data for analysis. While the software industry may have information on some attributes relevant to diversity studies, it is only the case for some. For instance, asking about an employee's ethnicity at work might not be legal. Care must still be taken when we have access to the data that can offer insights since observation will likely change the behavior.

Some studies derive information such as gender [21], geographic location [18], and ethnicity from signals [15], but this comes with its limits. These automated solutions miss nuances and sometimes reinforce stereotypes running in society. Therefore, diversity and inclusion must be investigated with great care. For more on ethics, refer to Chapter 9, "The Role of Ethics in Engineering Fair AI (and Beyond)."

Intersectionality: Most explorations in software engineering have focused on understanding problem space in one dimension. It needs to be made clear how improving diversity in one aspect influences another. For instance, if we are trying to improve gender diversity, are we also improving diversity in other forms?

A common notion is that improving diversity in any form promotes diversity in other forms, but preliminary evidence suggests the opposite. For instance, a recent study showed that teams diverse in gender are not necessarily diverse in geographical location and vice versa [17].

This raises an even more important question: whose problem do we solve? This question goes beyond the discussion on gender and location into deeper spaces where solving the problem for one person creates a problem for another. It only gets complicated from here, with software now having an uncodified part (courtesy of machine learning) that continues to evolve. This ever-changing element further complicates our understanding of fairness, diversity, and inclusion.

One way ahead is investigating how solving a problem for one subgroup influences another. This way, we facilitate making informed choices and trade-offs otherwise. This information is relevant since it generates awareness of what we can improve and acceptance of what we cannot.

While we previously discussed limitations relating to data, these issues increase multifold for investigating trade-offs. From the given data sources, it is often difficult to gauge the influence of all the characteristics that interact. Other alternatives, such as conducting experiments on humans, are challenging. The closest we have is conducting experiments on the student population. Still, studying a problem on a student population in many situations is not feasible. For instance, understanding how global software engineering teams collaborate might be challenging to replicate based on student population, even if globally distributed. Chapter 12, Exploring Intersectional Perspectives in Software Engineering Through Narratives,” presents explorations on intersectionality using narratives.

Ethics: Most problems arise because of the topic’s sensitive nature. With studies on diversity and inclusion, considering the ethics of doing a study becomes as important as the objective of bringing positive change in society. Since these goals can often be conflicting, studying and meaningfully exploring the choices is hard.

Diversity and inclusion: Improving diversity is essential; doing it meaningfully is more. Most explorations have been trying to understand the problem space and how to quantify it. Future exploration should now focus on finding solutions but also on looking into inclusion. This is as important since there is no meaning to improving diversity without doing it meaningfully or without making attempts for inclusion, which is a much more complex problem. Chapter 10, “Beyond Diversity: Computing for Inclusive Software,” provides more details on the subject.

Sustainable solutions: Another aspect to consider is the sustainability of a solution. Most current solutions are explored in isolation. In practice, however, for a solution to be sustainable, it must balance business needs and the needs of society. Only then are we minimizing the potential for harm while maximizing the interests of software, society, and industry.

Inspiring other fields: Amid all these challenges and limitations in software engineering, we still have something that can inspire research in other fields: software and data. We hope that what software engineering can do can inspire other fields to gather a closer understanding of their subject. Future research can explore novel ways to enable us to study relevant problems to solve diversity and inclusion issues. This could come either from software solutions or guidelines for building software solutions.

Takeaways

- Software is a creation and reflection of humans.
 - For a better society, have better software. For better software, have a better software development team.
 - Unless designed for the needs of any, a software system will miss the needs of many.
 - Development activity traces are our unique opportunity to understand diversity and inclusion issues in the wild.
 - Improving diversity is important; doing it meaningfully is more.
 - Sustainable solutions to improve diversity and inclusion are pragmatic and account for business needs.
-

Bibliography

- [1] Greg Anderson, Mark Keith, Conan Albrecht, Alex Spruill, and Clayton Pettit. Optimizing software team performance with cultural differences. 2019.
- [2] Fatma Bas, ak Aydemir and Fabiano Dalpiaz. A roadmap for ethics-aware software engineering. In *Proceedings of the International Workshop on Software Fairness*, 15–21, 2018.
- [3] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.
- [4] Clementine Collett, Livia Gouvea Gomes, Gina Neff, et al. *The Effects of AI on the Working Lives of Women*. UNESCO Publishing, 2022.
- [5] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 498–510, 2017.

- [6] Abdul Rehman Gila, Jafreezal Jaafa, Mazni Omar, and Muhammad Zahid Tunio. Impact of personality and gender diversity on software development teams' performance. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, 261–265, IEEE, 2014.
- [7] Omar S. Gómez, Martín Solari, César Jesús Pardo Calvache, and A. Carolina Ledezma. A controlled experiment on productivity of pair programming gender combinations: Preliminary results. In *CIBSE*, 679–692, 2017.
- [8] Valentina Grigoreanu, James Brundage, Eric Bahna, Margaret Burnett, Paul ElRif, and Jeffrey Snover. Males and females script debugging strategies. In *End-User Development: 2nd International Symposium, IS-EUD 2009, Siegen, Germany, March 2–4, 2009, Proceedings* 2, 205–224, Springer, 2009.
- [9] Karen A. Jehn, Gregory B. Northcraft, and Margaret A. Neale. Why differences make a difference: A field study of diversity, conflict and performance in workgroups. *Administrative Science Quarterly*, 44(4):741–763, 1999.
- [10] Wiesław Kopeć, Bartłomiej Balcerzak, Radosław Nielek, Grzegorz Kowalik, Adam Wierzbicki, and Fabio Casati. Older adults and hackathons: a qualitative study. In *Proceedings of the 40th International Conference on Software Engineering*, 2018.
- [11] Elvan Kula, Ayushi Rastogi, Hennie Huijgens, Arie van Deursen, and Georgios Gousios. Releasing fast and slow: an exploratory case study at ING. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 785–795, 2019.
- [12] Sean Mealin and Emerson Murphy-Hill. An exploratory study of blind software developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 71–74, IEEE, 2012.

- [13] Samim Mirhosseini and Chris Parnin. Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 84–94, IEEE, 2017.
- [14] Emerson Murphy-Hill, Jillian Dicker, Margaret Morrow Hodges, Carolyn D. Egelman, Ciera Jaspan, Lan Cheng, Elizabeth Kammer, Ben Holtz, Matthew A. Jorde, Andrea Knight Dolan, et al. Engineering impacts of anonymous author code review: A field experiment. *IEEE Transactions on Software Engineering*, 48(7):2495–2509, 2021.
- [15] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. On the relationship between the developers perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering*, 48(8):2955–2968, 2021.
- [16] Hema Susmita Padala, Christopher Mendez, Felipe Fronchetti, Igor Steinmacher, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Margaret Burnett, et al. How gender-biased tools shape newcomer experiences in OSS projects. *IEEE Transactions on Software Engineering*, 48(1):241–259, 2020.
- [17] Gede Artha Azriadi Prana, Denae Ford, Ayushi Rastogi, David Lo, Rahul Purandare, and Nachiappan Nagappan. Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in OSS. *IEEE Transactions on Software Engineering*, 2021.
- [18] Ayushi Rastogi, Nachiappan Nagappan, Georgios Gousios, and André van der Hoek. Relationship between geographical location and evaluation of developer contributions in GitHub. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–8, 2018.

- [19] Ayushi Rastogi, Suresh Thummalapenta, Thomas Zimmermann, Nachiappan Nagappan, and Jacek Czerwonka. Ramp-up journey of new hires: Tug of war of aids and impediments. In *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–10, IEEE, 2015.
- [20] Gema Rodríguez-Pérez, Reza Nadri, and Meiyappan Nagappan. Perceived diversity in software engineering: a systematic literature review. *Empirical Software Engineering*, 26(5):1–38, 2021.
- [21] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson R Murphy-Hill, and Chris Parnin. Gender bias in open source: Pull request acceptance of women versus men. *PeerJ Prepr.*, 4:e1733, 2016.
- [22] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Perceptions of diversity on GitHub: A user survey. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, 50–56, IEEE, 2015.
- [23] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 3789–3798, 2015.
- [24] Stanton William. *The Leopards' Spots: Scientific Attitudes Towards Race in America. 1815–1859*, 1960.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 2

ED&I and SE: Challenges, Progress, and Lessons

John Grundy, Monash University, Australia.*

Tanjila Kanij, Swinburne University of Technology, Australia.

Rashina Hoda, Monash University, Australia.

Hourieh Khalajzadeh, Deakin University, Australia.

Anuradha Madugalla, Monash University, Australia.

Jennifer McIntosh, Monash University, Australia.

Software is developed by humans to address human needs. Many human aspects impact end users of software, for example, sight differences; cognitive differences; user age, cultural background, language, and language proficiency; living and working conditions; and so on. Many human aspects also impact software engineers and teams, such as differences in gender, culture, language, emotions, motivation, experience, etc. While end users of software represent the wider human population with its variations, typically software developers form a more unique group: highly educated, highly proficient in English, very comfortable with technology, highly paid, and, unfortunately due to continued trends, mostly male. In order to build software solutions for the wider end user populations, it is critical for software developers to empathize with and deeply understand their end user differences and how that impacts software usage.

In this chapter, we provide an overview of a few of our recent studies identifying some of the key challenges of these human aspects impacting software engineers and software end users. We follow this with a summary of a few of our recent studies addressing how some human aspects impact software engineering (SE), including end user age and mental and physical challenges and developer emotions and gender.

We report some of the key lessons that we have learned to date from conducting these studies and present a brief research roadmap for better addressing these human aspects in SE.

Human Aspects and Software Engineering

Several human aspects impacting SE have been studied. In the following, we briefly review some of these, but this is by no means a comprehensive list – we have developed a preliminary taxonomy of end user human aspects [11] and one of developer human aspects impacting requirements engineering (RE) [14].

Gender: Gender bias toward software end users, for example, assuming users are male – this can include the use of noninclusive language, terminology, treatment, and assumptions about users based on their gender [42, 45]. Gender bias also impacts software engineers [Chapter 4 -Breaking the Glass Floor for Women in Tech, Chapter 5 - How Users Perceive the Representation of Non-binary Gender in Software Systems: An Interview Study, Chapter 11- Gender Diversity on Software Development Teams: A Qualitative Study] [22]. The GenderMag framework offers an approach to de-bias user interfaces [4].

Age: Impact of age on usage and/or age bias in design of software systems, for example, elderly users desiring interaction and terminology respecting their life experiences – this includes the different needs that young vs. older users may have, the different contexts of use for their software solutions, and wrong assumptions about differently aged users. We developed AgeMag to address age bias in ecommerce [33], and various guidelines exist for aging user design [36].

Emotions: People react differently to software, for example, some feel over-constrained and anxious when using smart home technology – some users react positively and others negatively to exactly the same solution. Different emotional reactions can have a profound impact on acceptance and adoption of software applications, explored in several recent work on emotions and requirements engineering from both end user and developer emotion perspectives [1, 5].

Physical or mental challenges: Both developers and end users may have a wide range of physical challenges, for example, a color-blind user with fine motor skill control limitations may struggle to use many mobile apps – challenges include color-blindness, sight challenges, hearing challenges, coordination challenges, and impacts caused by chronic disease such as stroke, obesity, cardiac arrest, infection, etc. These may be due to a range of causes, and their impact on different developers and users ranges

from inconvenient to making software impossible to use. Various design guidelines have been developed to help different challenges [19], though much less in the mental challenge area [Chapter 24 - Economical Accommodations for Neurodivergent Students in Software Engineering Education: Experiences from an Intervention in Four Undergraduate Courses] [32].

Spoken language: Spoken and written language differs among end users, for example, an app assuming high English proficiency and using technical jargon will confuse many end users – spoken and written language, language complexity, dialogue style, and language proficiency all may differ. Development teams may also use different languages, jargon, and terminology [21]. Some work has explored internationalization of software and simplification of text [41].

Socio-economic status: Different users have differing abilities to pay for and use technology, for example, a homeless person may only be able to afford a very low-end handset – this includes access to technology, affordability of solutions including software, network costs, etc. Their living and work environments may also greatly impact software use [12, 44]. Some users have no work, are under-employed or in precarious work, and have differing incomes, and all may have a significant impact on the design and deployment of software, especially in health and services domains. Limited guidance and tools address these issues arising from digital inequity [32, 35].

Culture and ethnicity: There is a need in development teams and in software developed to address cultural differences, for example, a global software team with mixed cultural work practices needs to respect differing approaches to authority – considerations include cultural practices, assumptions, behaviors, accepted and unaccepted practices, biases against particular users of software, and power structures in teams and groups. Limited work has been done on requirements and design approaches to incorporate culture and respect different ethnicities of users [Chapter 3 - The Challenges of Ethnic-Racial Diversity Within the IT Sector, Chapter 21 - Bringing Diversity in Software Engineering Education from the Middle East and Africa] [3, 47].

Geographic location: Where end users are located geographically, for example, users in rural areas often have low-bandwidth Internet impacting usage – considerations include rural/remote vs. urban, low-bandwidth area, and access to technologies and software. Studies show location may impact users and software systems significantly [48].

Identifying the Key Challenges

Survey 1 – Challenges in addressing diverse end user requirements: We initially conducted a large survey and targeted interviews of software developers to explore some of the human aspects of their end users they found most challenging to address and why [Chapter 7 - Developers' Perspective of Diverse End User Requirements]. We wanted to gain a better understanding of current developer approaches to addressing diverse end users' human aspects and key open challenge areas in this domain and determine key focus areas for researching new techniques and tools to address these [11]. Our survey was answered by 61 developers and managers, and then we interviewed a further 12 developers in detail. We wanted to better understand how these diverse end user human aspect issues are understood and addressed from an SE perspective. We asked specifically about end users' age, accessibility needs, physical and mental challenges, language and technology proficiency, socio-economic status, gender, and cultural differences. Some of the key reasons given by respondents why they found these aspects challenging to address include the broad range of the end user human differences that exist and have to be catered for; the different languages and range of comfort with technology of different user groups; different problem-solving styles of many end user groups; complexity of user interfaces in many application domains; and differences in terminology used, digital literacy, and the need to carefully consider text and icon usage for many target end users [11]. We asked developers what would help them improve development of their software to better address some of these diverse human aspects. Examples reported include better requirements capture and human aspect modeling support; providing developers with better guidelines and practices to follow; better design frameworks and tooling to address a greater range of end user human aspects; development tools that automatically prompt and advise on missing end user human aspect issues in designs and implementations; simpler interfaces in software for many end user groups; more live testing with representative end users, including more diverse participant recruitment approaches; better defect reporting to enable end users to more easily report problems; the need for better development processes to improve target end user involvement; and a need for better education of software engineers about diverse end user human aspects.

Survey 2 – Human aspects impacting software engineering activities: As with diverse end user requirements heavily impacting software engineers, RE activities themselves are some of the most human-centric activities in SE. We chose to conduct a study aimed at exploring the influence of human aspects on RE activities. We conducted

a survey with 111 software practitioners, applying a mixed-methods approach to analyze both the qualitative and quantitative data collected [15]. Key findings include an acknowledgment of the influence of human aspects on the performance of practitioners involved in RE activities; motivation, domain knowledge, attitude, communication skills, and personality were seen as some of the most significant human aspects impacting RE activities; differences in personality were seen to influence RE activities; and motivation factors include team collaboration, personal satisfaction, customer engagement, communication skills, and individual interest. We also identified a set of human aspects that were seen to reduce the effectiveness of individuals involved in RE activities. These include communication issues, inadequate domain knowledge, customer/stakeholder nature, team behavior, and management issues. We are now conducting an in-depth study into personality and motivation issues with two software teams.

Data analysis 1 – end user app review analysis: We wanted to better understand the deficiencies end users of software report, which could lead to improving its design and implementation. We analyzed human-centric issues reported in COVID-19 apps [6] and classified the human-centered problems end users encountered into several different kinds, including age, disability, emotions, gender, language, location, privacy, socio-economic status, and others. We found of the 2,611 manually analyzed app reviews, 716 identified human aspect-related issues. Over 50% said the human value of privacy was an issue, 14% user's geographic location, 11% socio-economic status, 11% language aspects, 5% emotion, and 4% age. However, these figures did vary a little for different COVID-19 apps from different countries [6].

Data analysis 2 – GitHub human-centric issues analysis: Contrasting with the preceding end user app review analysis, we wanted to find out how developers actually discuss the impact of diverse end user needs on SE work. We conducted an analysis of 1,691 issue comments from 12 diverse projects on GitHub, ranging from small to large-scale projects, including projects designed for end users with special needs, for example, visually impaired and dyslexic users [23]. Our analysis revealed that there are a wide variety of human-centric issues being discussed in these repositories. We categorized the human-centric issues into eight categories: inclusiveness, privacy and security, compatibility, location and language, preference, satisfaction, emotional aspects, and accessibility. Some categories, such as inclusiveness, cover several other human aspects, for example, age, gender, and culture. There is also an overlap between some categories and technical-related issues, such as compatibility. We found that privacy, satisfaction, and user preference categories were very commonly discussed in these repositories, whereas there was a limited discussion of accessibility and emotional aspects. Based on

the analysis, human-centric issues were found to be different across different projects, and unexpectedly, projects designed for end users with special needs, for example, visually impaired and dyslexic users, and the ones with large-scale end users, for example, Firefox, have limited discussions of human-centric aspects, specifically in accessibility and inclusiveness categories!

Progress Addressing the Challenges

We discuss a few of our recent studies looking to study specific human aspects or small numbers of them, their impact on software engineers and/or end users, and how we can address some of the challenges of human-centric issues on SE highlighted previously.

End user age - age bias and ecommerce software: Our app review analysis and survey of software engineers indicated *aging users* have very different needs from many younger users. We conducted a study on how people from different age groups behaved when they used ecommerce applications [33]. Following the InclusiveMag methodology [34] – a meta-method to increase software inclusivity in underserved populations – we identified specific requirements for ecommerce sites for people of different ages. This involved three steps – (1) scope, describing facets of older people using ecommerce; (2) derive, developing personas from the facets; and (3) apply, testing the personas using a cognitive walk-through [31] – with two ecommerce applications. Personas were developed from evidence-based facets [2, 30, 36, 37] and interviews with people ($n = 24$) from four different generations: Generation Z and Generation Y (combined, born 1981–2002 but not including those under 18), Generation X (born 1965–1980), Baby Boomers (born 1946–1965), and the Silent Generation (born 1928–1945). The personas that were developed were a general user and an elderly user as there was a clear distinction between the Silent Generation and younger users. Interestingly, all ages found ways to mitigate risk, another common facet when using ecommerce [2]. We performed a cognitive walk-through with the personas using Amazon ([amazon.com](#)) and Alibaba ([alibaba.com](#)) and found the elderly persona struggled to use both. These results demonstrate potential age bias against older people when using ecommerce applications. This is of particular concern given the increasing use of online ecommerce and especially during the COVID-19 pandemic lockdowns when at times it was the only way to buy essential items [8]. We are working on development of a new age inclusiveness Magnifier (“AgeMag”) to help identify age bias within ecommerce applications.

End user physical and mental challenges – augmented reality browser plug-ins:

Our GitHub discussion analysis and survey of software engineers both suggested that many software engineers struggle to understand challenges of end users different from themselves, especially those with *physical and mental differences*. A popular approach to improve developer empathy for these diverse accessibility challenges is “augmented reality” (AR) browser-based plug-ins – the attempt to mimic how a user with a particular challenge or set of challenges will view and interact with a target website. We conducted a survey with 30 developers to understand current usage of these plug-ins and found that while more than 60% of participants tried to ensure accessibility in different stages of the software life cycle, only 15% used plug-ins to understand end user challenges. They agreed that more end user involvement is needed to create empathy and explained that they would be more inclined to use these plug-ins if there were more awareness, better training, and better support. To explore the current status of these plug-ins, we analyzed popular plug-ins and conducted a heuristic evaluation with a commonly used plug-in, Funkify, with banking, ecommerce, and social media web applications [7]. Our evaluation showed that tools such as Funkify elicited the desired empathetic response, were easy to use, and offered flexibility and customization across a wide range of disabilities. However, these can be improved to cater to users with multiple challenges, add in more disabilities such as autism and hearing impairment that are harder to mimic than low vision or color impairment, and support sharing of content between developers for repeat testing [46].

Developer emotions – developers responding to requirements changes: Our survey of software engineers indicated their *emotional responses* to requirements challenges are diverse. We carried out an in-depth study on the emotions experienced by 201 practitioners as they responded to requirements changes [28] using a mixed-methods approach. One of our key findings was that participants reported feeling highly pleasurable emotions (e.g., enthusiasm, increased energy, inspiration) more commonly than less pleasurable emotions (e.g., anger, anxiety, fatigue). Individual practitioners, the team, the manager, and the customer were identified as the stimulus that triggered the emotional responses to requirements changes. Finally, the emotions of practitioners were seen to vary with the stages of the requirements handling, across receiving, developing, testing, and delivering requirements changes. Critically, we identified that introducing last-minute requirements changes close to a deadline was seen to violate the developer’s emotional well-being. We developed a set of practical recommendations for practitioners and researchers, including an emotion-centric decision guide for when to introduce and accept requirements changes.

Developer gender – gender bias in SE job advertisements: Our surveys of software engineers tended to confirm that SE remains a *male-dominated workforce*. One potential reason we speculated could be the way job advertisements are designed is *biased toward male candidates* [22]. Preliminary research conducted with software engineers and hiring managers indicated that is indeed the case. Traditional word-based gender bias detection methods are somewhat inappropriate in the context of SE. Based on the recommendations made by software engineers and hiring managers, we prepared a checklist to detect gender bias within job advertisements. We developed an NLP-based tool implementing the set of recommendations. In another study we conducted a broad-ranging survey of software engineers and prepared male and female SE candidate preferences, career goals, and job application behaviors. We represented these with personas to give hiring managers a better idea of diverse candidates while designing job roles. The personas can also be used to apply a cognitive walk-through to job advertisements to detect gender bias issues [22], based on the “GenderMag” framework [4].

Key Lessons Learned from Our Studies

We summarize a range of lessons learned from conducting the preceding challenge and progress studies and highlight some actionable recommendations for practitioners and researchers.

Including challenged end users: When developing solutions for challenged end users, it is critical to ensure their engagement throughout the project, but it has always been difficult to find large numbers of participants. Traditionally the main reason was difficulties they face in traveling and sufficiently diverse users [39]. However, with COVID-19, lack of familiarity with online tools and lack of support from these tools for differently challenged users was a serious challenge. We faced difficulties when looking for sufficient participants with low vision, dyslexia, color impairments, and motor impairments for our requirements gathering and evaluation studies. As alternatives, we leveraged related work analysis, had discussions with disability unit managers, and conducted cognitive walk-throughs [18, 46] and simulated actual participants via personas [26]. While SE communities are familiar with these methods, the accessibility and HCI research communities can be quite negative about them. More work is needed to emphasize the complimentary value of these methods to solely using real end users.

Recommendation: Use multiple, complimentary approaches to ensure diverse representatives of end users in development and studies.

Study recruitment: When researchers do requirements gathering, co-design, and evaluation with real users, it is critical to keep participants deeply involved throughout the research projects. Common methods to interact with participants include surveys, interviews, focus groups, and observations. However, during the COVID-19 pandemic, these methods could not be followed via traditional face-to-face or paper-based approaches. We had to adapt our recruitment methods to include more social media advertisements and recruitment via online events such as conferences, workshops, and meetups. We had to adopt remote engagement methods such as online interviews, online observations, and online surveys. These presented challenges, including Zoom fatigue, Internet traffic and connectivity issues, privacy issues in a digital environment, IT literacy issues, as well as missed opportunities created by the lack of face-to-face cues. Surprisingly, these methods seemed to have positive impacts on some studies since they created more equitable opportunities, reduced travel time, reduced anxiety for introverts, and provided equal-power relationships between researchers and participants.

Recommendation: Ensure flexible, human-centric recruitment and engagement approaches in studies and development.

Deep participant feedback: During our preliminary interview-based study on software practitioners' responses to requirements changes [29], we faced challenges in eliciting participant's responses about their emotions. Many people were not comfortable sharing details of their emotions, or how they felt, in a workplace context. In our follow up in-depth study [28], we revised our approach – a critical strategy was to share a list of emotions with them to pick, instead of starting on the proverbial blank slate. Using the job-related affective well-being scale (JAWS) [43] proved to be effective, as participants were able to select from options to capture their emotional responses. It was also helpful for us to categorize the responses into a coherent and finite set. vs using a fully open-ended response. The asynchronous and anonymous nature of the survey, as opposed to live face-to-face interviews, seemed to work better for eliciting responses. The effect of national culture on openness to share emotions at work (preliminary study of eight New Zealander and two Australian practitioners vs. in-depth study international in nature), and improved awareness of emotions and well-being due to work-from-home experiences during the COVID-19 pandemic, may also have improved quantity and quality of responses.

Recommendation: Many projects touch on sensitive human aspects; these need care and alternative approaches to respect participant concerns and improve engagement.

Multidisciplinary teams: Our surveys and GitHub discussion analysis highlighted that human-centric SE needs to incorporate other disciplines into the development process, such as health, psychology, and sociology – areas that are increasingly utilizing technological solutions for human-centric problems. Multidisciplinary teams are becoming the norm, with representation from all stakeholders important for the final product development of human-centric software and including them in the development process. The use of “co-design” and “living lab” methodologies in health in particular puts the end users and the “experts” at the center of the development process to incorporate human aspects in the product [10, 38].

Recommendation: Multidisciplinary teams with expertise outside SE are formed and used.

Need for multidisciplinary theories and techniques: Our developer emotion, gender bias, and human-centric issues studies all involved varying amounts of qualitative data. Hence, selecting appropriate qualitative data analysis techniques was critical to deriving rich findings and insights. We applied the qualitative data analysis techniques from socio-technical grounded theory (STGT) such as open coding and constant comparison along with memoing [16]. It allowed us to reach beyond descriptive findings and formulate theoretical frameworks, taxonomies, researcher reflections, and practical recommendations to share with the software practitioner and research communities [15, 28].

Recommendation: Qualitative methods including socio-technical perspectives are needed when studying human-centered issues.

Research and Practice Directions

In the following we briefly summarize some of the key research directions that we recommend be pursued to improve ED&I issues in SE.

Diversifying software teams: It has been shown that diverse teams perform better. In our investigation of gender bias within SE job advertisements, we found that gender-related bias also exists in computer science education and SE work environment [22]. To develop a gender-inclusive SE environment and gender-inclusive CS/SE education environment, we need to ensure the recruitment and career advancement process is gender inclusive. From the findings of our preliminary work with SE job advertisements, we also found that apart from gender, there are other biases relating to age, culture,

and English language proficiency of software engineers. To ensure an overall-**inclusive** SE work environment, more research is needed to identify these specific biases and to devise ways to alleviate them.

Recommendation: Further research into other biases in software teams is needed.

Supporting diverse software engineers: To be able to solve the needs of diverse users using software solutions, the software development teams themselves need to bring together people with a variety of skill sets, backgrounds, and technical and nontechnical expertise, for example, enabling end user software development using LowCode approaches accessible for users who are reliant on screen readers through a preliminary extension of an open source software, Node-RED [24]. More research is required to further investigate and analyze the accessibility of the existing software solutions, including end user-based tools, that is, LowCode approaches for users with visual impairments.

Recommendation: Ensuring diversity is supported and respected in SE teams is essential.

Impact of human aspects in different contexts: In the ecommerce and health domains, almost everyone is an end user, and therefore ehealth and ecommerce applications must be more inclusive. Underserved and under-represented populations are most at risk of harmful effects from the exclusion of human aspects including age, gender, culture, technical literacy, etc. [25], and underserved groups also have the highest burden of disease [27].

Recommendation: Research on how human aspects impact ED&I issues in different domains.

Under-represented end user human aspects in research: Our survey of developers and analysis of app reviews and GitHub and Stack Overflow discussions [11, 13, 23] have highlighted some end user human aspects whose requirements and design approaches to address are as yet little understood and addressed in SE. These include neuro-diverse end users; end users with low literacy and technology usage proficiency; addressing human values of transparency and honesty; motivational and engagement drivers in many domains; and – despite considerable research to date in other fields – aging and young end users. There is a pressing need for software engineers to be equipped with better guidelines, techniques, and tools to address these end users' specific software needs.

Recommendation: Research on impact of many other human aspects on SE is needed, for example, cognitive differences, literacy, human values, and motivational differences.

Adaptive Software for diverse end users: Most software is designed for the majority of users with homogeneous characteristics, often neglecting those in need of special features and support such as older adults, people from different cultures and languages, people with physical and cognitive challenges. For example, a red-green color-blinded person will face difficulties in separating red and green, a person with low vision will face issues in reading small symbols and text, while a Generation Z user would skim through most content at a rate of 4.4 seconds on every 100 words of text [40]. There are also culture-based different interpretations of color, for example, Chinese newspapers using green for negative stock values and red for positive [9]. Due to this inherent variety of human needs, a single design will not cater to audiences with distinct characteristics [20]. An approach we are using to address this is adaptive software. This allows users to define their own preferences and the developer to support adaptivity of the user interface and workflows accordingly. We have explored adaptive user interfaces in detail for color-blind, low-vision, and dyslexic users [26] and plan to explore other human aspects and software features.

Recommendation: End users are very different – we need better approaches to adapting actively to their needs, not them to software.

Impact of developer empathy and culture: Our surveys demonstrated many differences between most developers in understanding their end users. While AR/VR tools assist in supporting developers in understanding such differences, better support is needed. We need to explore yet more human and socio-technical aspects, such as role of culture – as an amalgamation of organizational, team, and individual culture [17] – in SE/RE activities, practices, and roles.

Recommendation: Researching the role of empathy in developer and end user interactions and understanding how developers view the role of ethics are essential.

Acknowledgments

Support from ARC Laureate Fellowship FL190100035, ARC Discovery Project DP200100020, and ARC Industry Transformation Hub IH170100013 is gratefully acknowledged.

Bibliography

- [1] Mohammed-Amr Abd El-Migid, Damon Cai, Thomas Niven, Jeffrey Vo, Kashumi Madampe, John Grundy, and Rashina Hoda. Emotimonitor: A Trello power-up to capture and monitor emotions of agile teams. *Journal of Systems and Software*, 111206, 2021.
- [2] Steven M. Albert and John Duffy. Differences in risk aversion between young and older adults. *Neuroscience and Neuroeconomics*, 2012(1), 2012.
- [3] Ahmed Alkaabi and Carsten Maple. Cultural impact on user authentication systems. 4(4):323–343, 2013.
- [4] Margaret M. Burnett, Simone Stumpf, Jamie C. Macbeth, Stephanne Makri, Laura Beck-with, Irwin Kwan, Anicia N. Peters, and William Jernigan. GenderMag: A method for evaluating software's gender inclusiveness. *Interact. Comput.*, 28:760–787, 2016.
- [5] Maheswaree Kissoon Curumsing, Niroshinie Fernando, Mohamed Abdelrazek, Rajesh Vasa, Kon Mouzakis, and John Grundy. Emotion-oriented requirements engineering: A case study in developing a smart home system for the elderly. 147:215–229, 2019.
- [6] Mattia Fazzini, Hourieh Khalajzadeh, Omar Haggag, Zhaoqing Li, Humphrey Obie, Chetan Arora, Waqar Hussain, and John Grundy. Characterizing human aspects in reviews of COVID-19 apps. In *9th IEEE/ACM International Conference on Mobile Software Engineering and Systems*, 2022.
- [7] Funkify. Funkify - a disability simulator for the web. www.funkify.org/, 2021.
- [8] Xuwen Gao, Xinjie Shi, Hongdong Guo, and Yehong Liu. To buy or not buy food online: The impact of the COVID-19 epidemic on the adoption of e-commerce in China. *PloS One*, 15(8):e0237900, 2020.
- [9] National Geographic. Even graphics can speak with a foreign accent. www.nationalgeographic.com/pages/article/2015626-datapoints-visual-cultures, 2015. Accessed on September 27, 2022.

- [10] John Grundy, Mohamed Abdelrazek, and Maheswaree Kissoon Curumsing. Vision: Improved development of mobile ehealth applications. In *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 219–223, IEEE, 2018.
- [11] John Grundy, Ingo Mueller, Anuradha Madugalla, Hourieh Khalajzadeh, Humphrey O. Obie, Jennifer McIntosh, and Tanjila Kanij. Addressing the influence of end user human aspects on software engineering. In *International Conference on Evaluation of Novel Approaches to Software Engineering*, 241–264, Springer, 2021.
- [12] Judith Grundy and John Grundy. A survey of Australian human services agency software usage. *Journal of Technology in Human Services*, 31(1):84–94, 2013.
- [13] O. Haggag, John C. Grundy, M. Abdelrazek, and S. Haggag. A large scale analysis of mHealth app user reviews. *To appear in Empirical Software Engineering*, 2022.
- [14] Dulaji Hidellaarachchi, John Grundy, Rashina Hoda, and Kashumi Madampe. The effects of human aspects on the requirements engineering process: A systematic literature review. *IEEE Transactions on Software Engineering*, 2021.
- [15] Dulaji Hidellaarachchi, John Grundy, Rashina Hoda, and Ingo Mueller. The influence of human aspects on requirements engineering-related activities: Software practitioners perspective. *ACM Trans. Softw. Eng. Methodol.*, 1(1), 2022.
- [16] Rashina Hoda. Socio-technical grounded theory for software engineering. *IEEE Transactions on Software Engineering*, 2021.
- [17] Rashina Hoda and James Noble. Becoming agile: a grounded theory of agile transitions in practice. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 141–151, IEEE, 2017.
- [18] Kenny Huynh, Juvent Benarivo, Chew Da Xuan, Giridhar Gopal Sharma, Jeffrey Kang, Anuradha Madugalla, and John Grundy. Improving human-centric software defect evaluation, reporting, and fixing. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 408–417, IEEE, 2021.

- [19] Luke Jefferson and Richard Harvey. Accommodating color blind computer users. In *Proc. 8th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, 40–47, 2006.
- [20] Aria YukFan Jim, Hyun Shim, Jue Wang, Lionel Richie Wijaya, Rongbin Xu, Hourieh Kha-lajzadeh, John Grundy, and Tanjila Kanij. Improving the modelling of human-centric aspects of software systems: A case study of modelling end user age in wireframe designs. In *ENASE*, 68–79, 2021.
- [21] Massila Kamalrudin, John Grundy, and John Hosking. MaramaAI: tool support for capturing and managing consistency of multi-lingual requirements. In *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, 326–329, IEEE, 2012.
- [22] Tanjila Kanij, John Grundy, Jennifer McIntosh, Anita Sarma, and Gayatri Aniruddha. A new approach towards ensuring gender inclusive se job advertisements. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 1–11, 2022.
- [23] Hourieh Khalajzadeh, Mojtaba Shahin, Humphrey O. Obie, and John Grundy. How are diverse end-user human-centric issues discussed on GitHub? Preprint at *arXiv:2201.05927*, 2022.
- [24] Alice Reid, Kaijie Lin, Hourieh Khalajzadeh, John Grundy, Lachlan Anderson, Briana Barker. Node-Read: A visually accessible low-code software development extension. In *LowCode at the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2022.
- [25] Karine Latulippe, Christine Hamel, Dominique Giroux, et al. Social health inequalities and ehealth: a literature review with qualitative synthesis of theoretical and empirical studies. *Journal of Medical Internet Research*, 19(4):e6731, 2017.

CHAPTER 2 ED&I AND SE: CHALLENGES, PROGRESS, AND LESSONS

- [26] Calvin Luy, Jeremy Law, Lily Ho, Richard Matheson, Tracey Cai, Anuradha Madugalla, and John Grundy. A toolkit for building more adaptable user interfaces for vision-impaired users. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–5, IEEE, 2021.
- [27] Johan P. Mackenbach, Ivana Kulhánová, Gwenn Menvielle, Matthias Bopp, Carme Borrell, Giuseppe Costa, Patrick Deboosere, Santiago Esnaola, Ramune Kalediene, Katalin Kovacs, et al. Trends in inequalities in premature mortality: a study of 3.2 million deaths in 13 European countries. *J Epidemiol Community Health*, 69(3):207–217, 2015.
- [28] Kashumi Madampe, Rashina Hoda, and John Grundy. The emotional roller coaster of responding to requirements changes in software engineering. *IEEE Transactions on Software Engineering*, 2022.
- [29] Kashumi Madampe, Rashina Hoda, and Paramvir Singh. Towards understanding emotional response to requirements changes in agile teams. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 37–40, 2020.
- [30] David J. Madden. Aging and visual attention. *Current Directions in Psychological Science*, 16(2):70–74, 2007.
- [31] Thomas Mahatody, Mouldi Sagar, and Christophe Kolski. State of the art on the cognitive walkthrough method, its variants and evolutions. *Intl. Journal of Human-Computer Interaction*, 26(8):741–785, 2010.
- [32] Felicia Mata-Greve, Morgan Johnson, Michael D. Pullmann, Emily C. Friedman, Isabell Griffith Fillipo, Katherine A. Comtois, Patricia Arean, et al. Mental health and the perceived usability of digital mental health tools among essential workers and people unemployed due to COVID-19: Cross-sectional survey study. *JMIR Mental Health*, 8(8):e28360, 2021.

- [33] Jennifer McIntosh, Xiaojiao Du, Zexian Wu, Giahuy Truong, Quang Ly, Richard How, Sriram Viswanathan, and Tanjila Kanij. Evaluating age bias in e-commerce. In *2021 IEEE/ACM 13th Int. Conf. on Cooperative and Human Aspects of Software Engineering (CHASE)*, 31–40, IEEE, 2021.
- [34] Christopher Mendez, Lara Letaw, Margaret Burnett, Simone Stumpf, Anita Sarma, and Claudia Hilderbrand. From GenderMag to InclusiveMag: An inclusive design meta-method. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 97–106, IEEE, 2019.
- [35] LU Miao. Technology translations between China and Ghana: the case of low-end phone design. In *Critiquing Communication Innovation: New Media in a Multipolar World*, 141–166, Michigan State University Press, 2022.
- [36] Michael G. Morris and Viswanath Venkatesh. Age differences in technology adoption decisions: Implications for a changing work force. *Personnel Psychology*, 53(2):375–403, 2000.
- [37] Daniel L. Murman. The impact of age on cognition. In *Seminars in Hearing*, Volume 36, 111–121, Thieme Medical Publishers, 2015.
- [38] Victoria Jane Palmer, Wayne Weavell, Rosemary Callander, Donella Piper, Lauralie Richard, Lynne Maher, Hilary Boyd, Helen Herrman, John Furler, Jane Gunn, et al. The participatory zeitgeist: an explanatory theoretical model of change in an era of coproduction and codesign in healthcare improvement. *Medical Humanities*, 45(3):247–257, 2019.
- [39] Helen Petrie, Fraser Hamilton, Neil King, and Pete Pavan. Remote usability evaluations with disabled people. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1133–1141, 2006.
- [40] Darla Rothman. A tsunami of learners called generation Z. 2016.

- [41] Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. Making it simplext: Implementation and evaluation of a text simplification system for Spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):1–36, 2015.
- [42] Yolande Strengers and Jenny Kennedy. *The Smart Wife: Why Siri, Alexa, and Other Smart Home Devices Need a Feminist Reboot*. MIT Press, 2020.
- [43] Paul T. Van Katwyk, Suzy Fox, Paul E. Spector, and E. Kevin Kelloway. Using the job-related affective well-being scale (JAWS) to investigate affective responses to work stressors. *Journal of Occupational Health Psychology*, 5(2):219, 2000.
- [44] Nita Vangeepuram, Victoria Mayer, Kezhen Fei, Emily Hanlen-Rosado, Cesar Andrade, Shari Wright, and Carol Horowitz. Smartphone ownership and perspectives on health apps among a vulnerable population in East Harlem, New York. *Mhealth*, 4, 2018.
- [45] Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, and Margaret Burnett. From gender biases to gender-inclusive design: An empirical investigation. In *Proc. 2019 CHI Conf. on Human Factors in Computing Systems*, 1–14, 2019.
- [46] Minh Hieu Vu, Joshua (Shuki) Wyman, and John Grundy. Evaluation of an augmented reality approach to better understanding diverse end user website usage challenges. In *ENASE*, 50–61, 2022.
- [47] Tian Xu, Jennifer White, Sinan Kalkan, and Hatice Gunes. Investigating bias and fairness in facial expression recognition. In *European Conference on Computer Vision*, 506–523, Springer, 2020.
- [48] Pu Yan and Ralph Schroeder. Variations in the adoption and use of mobile social apps in everyday lives in urban and rural China. *Mobile Media & Communication*, 8(3):318–341, 2020.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 3

The Challenges of Ethnic-Racial Diversity Within the IT Sector

Michelle Borges Miranda (deceased)

Rafael Prikladnicki, Pontifical Catholic University of Rio Grande do Sul, Brazil.*

Ethnic diversity is the expression of diversity one finds within a single entity (a business, a team, a community, or a country). A person's ethnicity usually includes beliefs, nationality, and language and gives the person a distinct sense of belongingness among a group [17]. Diversity means including or acknowledging people from a wide range of backgrounds. The existence of people from various ethnic and cultural backgrounds or identities is referred to as ethnic diversity. Ethnic-racial diversity is important because people should have a commitment to evaluate their personal biases and stereotypes in order to facilitate collaboration and cooperation[1]. If someone only cares about their core beliefs and cultural identity, people will not work together to benefit the common good. There must be a common understanding and willingness to push society forward for the benefit of everyone. For this reason, diversity in IT and software development teams is a topic that has attracted the attention of researchers and organizations, who seek to understand the benefits and challenges that diversity can provide [21].

There is an increasing debate about ethnic diversity in IT and a broad body of research demonstrating the importance and effectiveness of diverse teams as being more innovative and more productive and directly impacting the business rules of companies [Chapter 21, “Bringing Diversity in Software Engineering Education from the Middle East and Africa,” Chapter 29, “Strategies for Reporting and Centering Marginalized Developer Experiences” [7]]. For example, studies on diversity in software engineering indicate

CHAPTER 3 THE CHALLENGES OF ETHNIC-RACIAL DIVERSITY WITHIN THE IT SECTOR

that the efficient use of talent from diversity groups can play an active role and benefit organizations in terms of competitiveness, performance, economic growth, and social development [8, 10, 13, 21, 24, 26]. According to Patrick and Kumar [18] and Mujtaba [15], diversity among employees tends to make the business environment conducive to serving external customers, in addition to promoting positive publicity for the company.

In terms of benefits for companies, other indicators can be mentioned such as the use of different experiences to add a different perspective, increasing the number of solutions to be developed [23]. In addition, diverse environments tend to promote the development of more robust final products and develop solutions to more complex problems [4, 25]. It is noticed that software development companies tend to increasingly observe the importance of diversity and inclusion in software development teams. As a result, they seek to implement diversity management models to enhance the positive effects of this inclusion. However, the benefits and challenges of including diversity in IT are not only related to organizations, universities, and companies; they also permeate among the people who make up the diversity groups. Through the accounts of diversity groups, such as ethnic-racial diversity, we can observe that social issues (e.g., the way people interact with each other based on their culture), for example, have a profound impact on the inclusion process, whether in the educational environment or in the industry.

The purpose of this chapter is to share how ethnic-racial diversity has been explored in software engineering literature and related areas. To achieve this objective, we carried out a systematic mapping of the literature. According to our results, we observed that the field of research on ethnic-racial diversity in IT is rich and complex. The number of papers found demonstrates that this is a research topic of recent interest in the field of computing. The papers found are divided into reports of personal experiences lived in academia or industry, reports of experiences in educational programs aimed at teaching programming, and the management model of inclusion of ethnic-racial diversity in the industry.

We also observed that few papers are dedicated to carrying out a broader discussion on the origin of the problems and challenges that impact ethnic-racial diversity groups, and there are no definitions about ethnic-racial diversity groups or even consensus on which terms define such groups.

Ethnic-Racial Diversity

Boukreris et al. [3] observed that the idea of diversity conveys the notion of variety, difference, and opposition. According to Page [17], diversity can be understood in two groups: identity and cognitive diversity. Identity diversity is characterized by other subgroups such as gender, ethnicity, religion, sexual orientation, culture, and other social markers. Cognitive diversity differentiates individuals based on their intellectual capacity.

Debating only about diversity in its general sense can result in distancing or even erasing the various groups that help compose this diversity. Therefore, identifying which diversity groups we are referring to and discussing the aspects that characterize them is essential to obtain a more accurate understanding of the data that are raised in relation to the diversity groups in question.

Within the identity diversity groups, we can infer ethnic-racial diversity. Ethnic-racial diversity according to Gomes [9] is the way in which some intellectuals refer to the black segment (formed in Brazil by black and brown people). According to the author, the term *ethnic-racial* comprises the multiplicity of dimensions and issues that involve the history, culture, and life of blacks in the African diaspora¹ (like Brazil), in addition to physical characteristics and racial classification.

For Gomes [9] the ethnic-racial expression has been adopted within the theoretical and political contexts, trying to end the impasse and dichotomy between the concepts of race and ethnicity to refer to the black segment. The author also adds that, for an in-depth understanding of ethnic-racial relations, one must consider the identity processes experienced by the subjects, that is, the way in which they observe themselves, identify themselves, and talk about themselves and their ethnic-racial belonging.

In this work, we chose to use the term *ethnic-racial diversity* to refer to the black population, whether in African countries or in their diasporas, regardless of the form of composition of this group in each country of the diaspora, for example, in Brazil it is composed of black and brown people. We also use the term *black people*, when we see it necessary to give greater emphasis to the ethnic-racial diversity group we are referring to. Thus, in the next section, we will reflect on the need to debate the nuances of ethnic-racial diversity within IT.

¹The African diaspora is the name given to a phenomenon characterized by the forced immigration of Africans to other countries. These countries are now called African diasporas.

The Challenges of Ethnic-Racial Diversity: A Necessary Conversation

In Brazilian society, there exists the “myth of racial democracy,” an idea that suggests that all white people and non-white enjoy exactly the same rights and privileges. Besides that, this does not consider the currently embattled for the social rights of the non-white population. However, there is some progress about the matters of increasing public politics to fight against racial inequality. According to Ribeiro [22] (translated by the author)

The path to change is complex, but reality is not immutable. The strengthening racial consciousness has been an important element in this construction. Ethnic-racial education is an important aspect for the realization of democracy. In the 21st century, the State must accelerate the process of inclusion, based on racial policies, as a focus of social justice.

—Matilde Ribeiro

Among the issues that are deepened by the myth of racial democracy is racism. The issues born of racism exist and are huge. They are related to all structures of society: economic, political, social, cultural, historical, and religious. However, racism does not act only in the field of structures; it creates ways of being and thinking, it is systemic, and, therefore, it determines the actions of individuals insofar as it defines and impregnates culture. Among the problems intensified by racism are the few or nonexistent opportunities for blacks in certain sectors of society and the prejudices, arising from stereotypes, built on black bodies and reinforced by structural racism²:

The stereotypes are mental plans from the cultural process of classification of pieces of information. But, this categorising is formed by the perception of social rules explicit and non-explicit, and about this point there is a correlation between the inconscient and the racism.

—Adilson José Moreira [14] (translated by the author)

²Structural racism is a set of institutional, historical, and cultural discriminations and practices within a society that often privileges some races to the detriment of others (our translation) [2].

The technology sector also has some prejudices about why certain bodies are more or less frequent within this space, for example, some gender and ethnic-racial prejudices try to explain that women are naturally less interested in the area of technology or that certain cultures and countries have more aptitude for the sector [5, 12]. These biases exclude from the analysis the necessary incentive for people to decide for a certain area and link the justification of the natural to the choices made [6, 11].

According to the Stack Overflow annual survey conducted in 2022, with more than 69,000 users working in the technology sector in the programming area, about 1.49% of people declared themselves to be black in the question about race and ethnicity. However, the response related to the white option represents about 40% of users, followed by the European option, about 37%. The other options appear below 10%. Given these results, it seems that the population within the programming area is mostly white, according to Stack Overflow. It is therefore observed that analyzing the low participation of black people in the technology sector through the lens of meritocracy is a mistake. Because to understand that racism is the main illness that prevents the access of black people to the spaces that they can occupy, mainly from the filter of culture, is to understand the dynamic, renewable, and conflictive character of racism.

Therefore, reflecting on these and other challenges faced by the black population is essential to devise intentional actions to face these challenges and promote abundant ethnic-racial inclusion in the area of technology, because from these reflections, we no longer understand the number of black people in technology as a matter of aptitude and choices, but as the result of a series of combined factors that resulted in few opportunities for a population that, only in Brazil, constitutes themselves like the majority of the population. In the next section, we present the methodology used for the development of this work.

Research Methodology

For the development of this systematic mapping, we used the guidelines provided by Petersen et al. [19]. The objective of this mapping is to identify how ethnic-racial diversity is being considered in software engineering research. As a result, the following research question (RQ) was defined:

- (RQ) What studies have been published on ethnic-racial diversity in SE, and what are their challenges?

CHAPTER 3 THE CHALLENGES OF ETHNIC-RACIAL DIVERSITY WITHIN THE IT SECTOR

According to Petersen et al. [19], to start the systematic mapping, it is necessary to define the scope of the research, which includes the definition of research questions and identification of keywords. The keywords defined were software engineering, software development, software developer, race, ethnicity, geography, socioeconomic status, ethnic-racial, and racism. We also tested the search string using keywords such as people of color and black people, but the results did not change.

We chose to search in electronic databases that met the following source selection criteria: databases that include articles from related journals and conferences in the respective context of this research, databases with an advanced search engine that allows you to filter the results by keywords that address the search questions, and databases that provide access to full articles written in English.

Based on these criteria, the following databases were selected: ACM Digital Library (DL), IEEE Xplore, and Scopus. The search *string* has been adapted for each database according to the search functionality offered by it. In parallel, a list of control articles was generated, used as a way of validating and guaranteeing reliability and relevance and evaluating the research sequences.

The search *string* was formulated according to combinations, variations between keywords, and some characteristics of these words. The end result was as follows:

(“Software Engineer” OR “Software Developer” OR “Software Development”) AND (“Ethnicity” OR “Racial” OR “Ethnic” OR “Socioeconomic Status” OR “Ethnic-Racial” OR “Racism”)

For this *string*, no time restriction was applied, since the focus was to understand when ethnic-racial diversity is a subject for research in the area of software engineering. The keyword “socioeconomic status” was selected due to the observation in previous papers that some articles, when analyzing the socioeconomic issue of people in technology, have the black population as a research group.

The inclusion and exclusion criteria were defined to optimize the filtering of the results obtained, executing the search sequence in the chosen databases, such as follows:

Inclusion criteria

- The terms defined for the search or by similarity with the subject
- Articles from academic journals, conferences, and *workshops*
- Works written in English

Exclusion criteria

- Keywords and abstracts that are not focused on software engineering or related areas
- Articles that are not focused on software engineering or related areas
- Conference proceedings, courses, standards, panels
- Study format not in pdf or not available
- Articles that do not deal directly with the black population, that is, that do not analyze this population and its nuances
- Articles that only mention ethnic-racial diversity among existing diversity groups

After the initial query, we reviewed the results to determine which results should be included in the analysis based on the inclusion and exclusion criteria. A two-step process was followed to identify the set of papers in scope of the review: The abstract of each paper was reviewed to determine whether it should be considered further. This resulted in a further reduction in the number of papers to consider further. In some cases, it was unclear from the abstract as to whether the paper met the inclusion or exclusion criteria in which case it was categorized as a Maybe and taken forward to the next step. Subsequently, the full paper was briefly read to consider whether it should be considered further. In some cases, a discussion took place between us two authors to decide whether to include the paper.

Each paper was then read in its entirety and classified according to the target audience and search field. After that, we identified common themes for each paper and summarized the main results. In the next section, we present the results obtained from the implementation of this methodology.

Findings

In this section, we present the results of the systematic literature mapping. We selected 11 papers that we accepted based on the acceptance criteria defined for this work. The string used to find these articles was applied in August 2022 to the following search databases: ACM Digital Library, IEEE Xplore, and Scopus. The number of papers found through the string and selected can be visualized in Table 3-1. We analyzed the general

themes of each paper, the target audience within the ethnic-racial population, whether the field of research was academia and/or industry, as well as the territories on which the papers were addressed, that is, if the reported experiences were about countries on the African continent or its diasporas. The accepted articles can be observed in Table 3-2.

Table 3-1. *Papers per electronic database*

Source	Initial	Selection	Accepted
ACM DL	792	25	9
IEEE Xplore	135	3	1
Scopus	42	2	1
Total	848	30	11

During the information extraction phase, we read the 11 papers to analyze and classify them according to the research question. We note that most of the papers are recent, and therefore it is suggested that it is a topic on the rise. We created a classification scheme to present the results according to extracted data. Such classifications were territory, target audience, and research field. Due to the number of papers found, we believe that the analysis of articles based on these indicators will help us obtain a greater understanding of how research on ethnic-racial diversity in the area of computing is being conducted.

We observed that, regarding the territory indicator, 100% of the articles dealt with cases and experiences in African diaspora countries, such as Brazil and the United States of America. Only P11 spoke of experiences in African countries beyond their diasporas. The other classifications can be observed in Tables 3-3 and 3-4. Some themes appeared in more than one article in different ways, such as the insertion of black people in the area of computing, whether in academia or industry, challenges related to stereotypes within the area of computing, opportunities and challenges to remain in the area, and racism. In the next section, we will analyze the data extracted from these articles and reflect on the challenges and opportunities they present.

Table 3-2. List of accepted articles

Study	DL	Title	Authors	Source	Year
P1	ACM	Towards a Model for Managing Diversity and Inclusion in Software Development Teams	Michele Miranda and Rafael Prikladnicki	SBES '20	2020
P2	ACM	Better Late Than Never: Exploring Students' Pathways to Computing in Later Stages of College	Kathleen J. Lehman, Annie M. Wofford, Michelle Sendowski, Kaitlin NS Newhouse, and Linda J. Sax	SIGCSE '20	2020
P3	ACM	I Can't Breathe: Reflections from Black Women in CSCW and HCI	Sheena Erete, Yolanda A. Rankin, and Jakita O. Thomas	CSCW3	2021
P4	ACM	Black Men in IT: Theorizing an Autoethnography of a Black Man's Journey into IT Within the United States of America	Curtis C. Cain and Eileen Trauth	SIGMIS	2017
P5	ACM	The Intersectional Experiences of Black Women in Computing	Yolanda A. Rankin and Jakita O. Thomas	SIGCSE '20	2020
P6	ACM	Characterizing the Social Ties Between Black and Tech Communities on Twitter	Marisa Vasconcelos, Priscila Rocha, Julio Nogima, and Rogerio Paula	WebSci '22	2022
P7	ACM	Information Systems in the Community: A Summer Immersion Program for Students from Historically Black Colleges and Universities (HBCUs)	Jeria Quesenberry, Randy Weinberg, and Larry Heimann	SIGMIS-CPR '13	2013

(continued)

Table 3-2. (continued)

Study DL	Title	Authors	Source	Year
P8 ACM	African American Males Constructing Computing Identity	Betsy James DiSalvo, Sarita Yardi, Mark Guzdial, Tom McKlin, Charles Meadows, Kenneth Perry, and Amy Bruckman	CHI '11	2011
P9 ACM	What Is a Computer Scientist?: Unpacking the Ontological Beliefs of Black and Hispanic Female Computing Students	Jake Lopez, Monique Ross, Atalie Garcia, and Carolina Uribe-Gosselin	SIGCSE 2022	2022
P10 IEEE	On the Relationship Between the Developer's Perceptible Race and Ethnicity and the Evaluation of Contributions in OSS	Reza Nadri, Gema Rodriguez-Perez, and Meiyappan Nagappan	IEEE Transactions on Software Engineering	2022
P11 Scopus	Black Professional Communicators Testifying to Black Technical Joy	Antonio Byrd	Technical Communication Quarterly	2022

Table 3-3. Classification by target audience

Target Audience	Paper
Black women	P3, P5, P9
Black men	P4, P8
Black population	P1, P2, P6, P7, P10, P11

Table 3-4. Classification by search field

Search Field	Paper
Academia	P2, P3, P7, P8, P9
Industry	P1, P6, P10, P11
Academy and industry	P4, P5

Analysis of the Results

According to the results presented in the previous section, we obtained three classifications of indicators: territory, target audience, and research field. However, there was an intersection of categories as we analyzed the data. Regarding education and the academic environment, the papers found (P2, P3, P7, P8, P9) observe issues related to access to higher education in computer courses, challenges related to student permanence such as dropout rates, training rate, and entry into the labor market. The paper P2 observes the entry of women and black people in the field of computing after their experiences in other fields of studies, due to not seeing themselves as computer scientists given questions of stereotypes about the area and their identities. The papers P3 highlights the violence that black academics suffer. Still on the field of education, some works, in addition to mapping the number of black people in computing, make suggestions for educational actions to increase the insertion of black people in computing or their permanence in the area. This is the case of studies P7 and P8, which suggest summer programming courses for African-American students or specifically for black men.

As for the studies that refer to the research field of industry (P1, P6, P10, P11), we observed that the issues related to insertion and permanence reported to the academic research field remain. The paper P10 looks at the nuances related to accepting contributions to OSS projects in relation to ethnic groups such as black people. The authors state that specific diversity factors related to race, personality, gender, geographic location, and other social markers can affect the acceptance or rejection of pull requests. Also according to the authors, there are differences in the acceptance rate of work pull requests between groups of different ethnicities. P11 also notes that the field of computing in recent years, through higher education and code bootcamps, has realized the promise of social mobility, which for a historically marginalized population is a breath of hope about individual achievements.

CHAPTER 3 THE CHALLENGES OF ETHNIC-RACIAL DIVERSITY WITHIN THE IT SECTOR

The papers P4 and P5 note that whether in academia or industry, the proportion of black people in technology is staggeringly low. P4 highlights that black men in the United States of America have a strong inclination toward technology during their youth, but this identification does not advance toward professionalization and performance in the technology market. This mismatch may occur because black students are historically considered academically inferior when compared with white students and are more likely to attend schools that are poorly equipped with resources that contribute to providing a quality education. The paper P5 highlights that, although there is an effort to include women in the field of computing, the number of black women who benefit from these policies is still low.

A common aspect found in the articles selected through the mapping is the report of the low insertion of black people in the computing area. The authors of P10 highlight the experience within the OSS community regarding submitting and correcting code for OSS projects. Among their results, they note that the number of black people developing software is 0.1% out of a total of 6.8% of developers on the GitHub platform. They found that 0.13% of contributions sent to OSS projects on GitHub were made by black people and that nontechnical factors related to code quality such as developer ethnicity influence the evaluation result, for example, raising the acceptance rate of contributions from white people between 6 and 10% and that contributions from developers in Asia and Africa are about 7 and 16% less likely to be accepted when compared with contributions from senders in North America. Also according to the authors, people of the same ethnicity are more empathetic in correcting the codes of their peers.

In certain studies (P2, P4), it is observed that the authors do not understand the insertion of black people either in the field of academia or industry, related only to factors of personal interest or better aptitude (reinforcing the idea of merit). Some works reinforce that there are social factors that influence the participation of black people in the field of computing, such as the issue of stereotypes. According to the study P4, among the many stereotypes aimed at black people are those that black men are aggressive and that black people in general are superstitious, lazy, carefree, aggressive, intellectually inferior, ostentatious, active in sports, artistic, and poor academic performers. In this sense, it is vital to combat this racism and stereotype for the broad participation of black people in various spaces of society, including computing. According to P8 people are more likely to identify with something if they perceive it to be the norm in their social group. However, the stereotype of the dominant group that makes up the field of computing is non-black people.

Another important analysis refers to the intersectionality of gender and racial issues. Works that address the racial issue focusing on black women deal with other elements related to female gender oppression combined with racism, which affect black women in all structures of society, whether by their gender or racial peers. This intersectionality that racism and gender oppression provoke not only pushes away the ideal of equal treatment but strengthens projects of domination, forming various social hierarchies. According to the authors of the article P3, even among black women with higher levels of education and employment, there is a wage gap, greater stress, and depression in relation to, for example, white women. According to the authors, "Being black in computing is an act of resistance to the dominant culture that denies our existence and refutes that racism exists even within the 'meritocracy' of computing."

The use of self-ethnography to report personal experiences both in academia and industry by black people stands out (P3, P4, P5). According to the authors of papers P3, P4, and P5, self-ethnography and testimonial authority are methodological approaches little used in the area of computing, but which, however, are rich sources of reports and observations from an epistemic agency to witness lived experiences, because, according to the authors, there is no separation between academic or industrial life and systemic oppression, which is reflected in the absence of better opportunities and low wages. The area of software engineering is a fertile field of recognition of studies of this nature as legitimate research and a catalyst for social change. This methodological approach sees black people as subjects of rights and in the production of the study.

Other analyses are still possible from these results that could not be included in this chapter, and, mainly, new research should be carried out in order to deepen the discussion of this topic. In the next section, we present our final remarks.

Final Considerations

Discussing ethnic-racial diversity is a challenge that begins with the definition of terms, especially when the group is composed of black people. Understanding the constitution of the black, the factors that imply in self-identification are disciplines that are beyond the scope of this work, although without them it is impossible to demonstrate the complexity and richness of the subject. We observed, for example, that the paper P10 reports on the difficulty that developers have with self-declaration when it comes to the black segment, which makes it difficult to accurately express the amount of this population within the area. Even so, although there is this complexity in the definition

of the term, according to the papers selected for this study, it was evident that the black population is smaller in quantitative terms within the area of software engineering compared with other ethnic groups, mainly that of white people.

We observed that there is an understanding that ethnic-racial diversity is important within the area of technology and that the factors that hinder the greater insertion of this group in the area of computing break the veil of meritocracy and personal aptitude, giving rise to other analyses such as stereotypes, racism, power relations in society, the myth of racial democracy, and meritocracy, which are grounded by structural racism.

Among the limitations of this work are the possible amounts of keywords that can return relevant papers on the topic. As it is a recent field of research, it is observed that the community is still learning to define and use the best terms to deal with ethnic-racial diversity within IT. Another limitation is the local theoretical framework. In Brazil there are renowned researchers who are exponents of research on ethnic-racial diversity that drive the debate in society and the creation of public policies, but who however politically choose to write their works in Portuguese. Therefore, the research that researchers in IT have taken on the task of carrying out, investigating in an increasingly profound way the origin of the problem of the insertion of black people in computing, the actions of reparative policies, and the generation of opportunities through the state and the industry that aim to mitigate this problem, combined with anti-racist actions, can play a very important role in the fight for more diversity in the area of computing. This can be seen already with a significant discussion on gender diversity, and we do believe that bringing the ethnic-racial diversity discussion to the same level is really important. And starting that was the main goal of this chapter.

Acknowledgments

Rafael Prikladnicki is partially supported by FAPERGS and CNPq in Brazil.

A Dedication to the Memory of Michelle Miranda

The first version of this chapter was mostly written by Michelle Miranda, a PhD student supervised by me, the second author. Unfortunately and sadly, Michelle passed away on November 2, 2022, a month before we received the first reviews on this chapter. Michelle was an amazing person and researcher. Her PhD was exactly on the topic of this chapter.

Most of what I've learned about diversity and inclusion in the past four years was by interacting with her, a black woman that made a difference and raised the awareness of this important topic. She has contributed significantly to this work, and for this reason and as a dedication to her memory, she remains as the first author of this chapter.

Bibliography

- [1] Madiha Ahmed. Ethnic diversity in the workplace: The good, the bad, and the ugly. *Aisthesis - Interdisciplinary Honors Journal*, 10(1):10–17, 2019.
- [2] Silvio Almeida. *Racismo estrutural*. Pólen Produção Editorial LTDA (in Portuguese), 2019.
- [3] Louafia Boukreris and Ghania Ouahmiche. Diversity: Concept and issues. *International Journal of Language and Linguistics*, 5(1):15, 2017.
- [4] John H. Bradley and Frederic J. Hebert. The effect of personality type on team performance. *Journal of Management Development*, 1997.
- [5] Sapna Cheryan, Allison Master, and Andrew N. Meltzoff. Cultural stereotypes as gate-keepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology*, 49, 2015.
- [6] Brianna Dym, Namita Pasupuleti, Cole Rockwood, and Casey Fiesler. "You don't do your hobby as a job": Stereotypes of computational labor and their implications for cs education. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 823–829, 2021.
- [7] Denae Ford and Brittany Johnson. *Strategies for Reporting on the Marginalized Developer Experience*. Equity, Diversity, and Inclusion in Software Engineering, 2023.
- [8] Jason Freeman, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. Engaging underrepresented groups in high school introductory computing

- through computational remixing with EarSketch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 85–90, 2014.
- [9] Nilma Lino Gomes et al. Alguns termos e conceitos presentes no debate sobre relações raciais no brasil: uma breve discussão. *Educação anti-racista: caminhos abertos pela Lei Federal*, 10639(03):39–62, 2005.
 - [10] Karina Kohl and Rafael Prikladnicki. Perceptions on diversity in Brazilian agile software development teams: a survey. In *2018 IEEE/ACM 1st International Workshop on Gender Equality in Software Engineering (GE)*, 37–40, IEEE, 2018.
 - [11] Colleen M. Lewis, Ruth E. Anderson, and Ken Yasuhara. “I don’t code all day”: fitting in computer science when the stereotypes don’t fit. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 23–32, 2016.
 - [12] Allison Master, Andrew N. Meltzoff, and Sapna Cheryan. Gender stereotypes about interests start early and cause gender disparities in computer science and engineering. *Proceedings of the National Academy of Sciences*, 118(48):e2100030118, 2021.
 - [13] Álvaro Menezes and Rafael Prikladnicki. Diversity in software engineering. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE 18, 4548, Association for Computing Machinery, New York, NY, USA, 2018.
 - [14] Adilson José Moreira. *Pensando como um negro: ensaio de hermenêutica juídica*. Editora Contracorrente (in Portuguese), 2019.
 - [15] Bahaudin Mujtaba. *Workforce Diversity Management: Challenges, Competencies and Strategies*. Llumina Press, 2007.
 - [16] M. C. W. Peeters, W. B. Schaufeli, and W. G. M. Oerlemans. Ethnic diversity at work: An overview of theories and research. In *The Individual in the Changing Working Life* (eds K. Nswall, J. Hellgren, and M. Sverke), 211232. Cambridge University Press, 2008.

- [17] Scott E. Page. *The Diversity Bonus: How Great Teams Pay Off in the Knowledge Economy*. Princeton University Press, 2019.
- [18] Harold Andrew Patrick and Vincent Raj Kumar. Managing workplace diversity: Issues and challenges. *Sage Open*, 2(2):2158244012444615, 2012.
- [19] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE '08, 68–77, BCS Learning & Development Ltd., Swindon, UK, 2008.
- [20] Pinterest. 2015 Inclusion & Diversity Report. <https://newsroom.pinterest.com/en/post/our-plan-for-a-more-diverse-pinterest-2015>, 2015.
- [21] Jeria Quesenberry, Randy Weinberg, and Larry Heimann. Experiences in service-learning pedagogy: lessons for recruitment and retention of under represented groups. In *Proceedings of the 50th Annual Conference on Computers and People Research*, 89–90, 2012.
- [22] Matilde Ribeiro. *Políticas de promoção da igualdade racial no Brasil (1986–2010)*. Garamond (in Portuguese), 2019.
- [23] Thomas R. Roosevelt. From affirmative action to affirming diversity. *Harvard Business Review*, 68(2):107–117, 1990.
- [24] Sayma Sultana, Jaydeb Sarker, and Amiangshu Bosu. A rubric to identify misogynistic and sexist texts from software developer communications. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–6, 2021.
- [25] Valerie Taylor and Richard Ladner. Data trends on minorities and people with disabilities in computing. *Communications of the ACM*, 54(12):34–37, 2011.

CHAPTER 3 THE CHALLENGES OF ETHNIC-RACIAL DIVERSITY WITHIN THE IT SECTOR

- [26] Kera ZB Watkins and Maurice J Watkins. Towards minimizing pair incompatibilities to help retain under-represented groups in beginning programming courses using pair programming. *Journal of Computing Sciences in Colleges*, 25(2):221–227, 2009.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 4

Breaking the Glass Floor for Women in Tech

Igor Steinmacher, Northern Arizona University, USA.*

Bianca Trinkenreich, Colorado State University, USA.

Anita Sarma, Oregon State University, USA.

Marco Gerosa, Northern Arizona University, USA.

Women¹ face many challenges in the tech industry. Implicit gender biases significantly impact hiring decisions, women disengage with technology faster than men, women are often less likely to get their code accepted, and tools, processes, and even education are not inclusive [1]. A direct outcome of these biases is that women represent less than 24% of the employees in the software development industry, with an even further imbalance in participation in Open Source Software (OSS) projects (around 10%). The software industry lacks equitable participation conditions, exacerbating the gender imbalance of those who shape technology.

Reducing the gender gap requires not only attracting but also retaining women in tech. Implicit biases, including sexist behavior, pervade the software industry and create a “glass floor” – a persistent barrier to enter the tech world. Even when women break through this barrier, they face sociocultural challenges due to the hostile social dynamics in their daily work. Instances of women needing to hide their gender in their profile, stopping their contribution to an OSS project, leaving their jobs in the software industry,

¹In this chapter, we use the term “gender” as a socially constructed concept, where gender identification, display, and performance might or might not align with a person’s sex assigned at birth. To reflect this social concept of gender, we use the term “women” and “men” as a shorthand for people who self-identify as such.

or altogether giving up on the technology area are unfortunately frequent occurrences. Even in companies and communities that care about diversity and inclusion, sociocultural problems are prevalent.

Through a systematic literature survey of 51 studies [14] and an empirical study with software development teams from a global ICT company [15], we identified seven types (Figure 4-1) of challenges that women face and eleven actions that companies and communities implement to overcome them. In the following, we present some findings reported in this literature.

Challenges



Figure 4-1. Challenges reported by women in companies [14] and OSS projects [15]

Lack of peer parity: An imbalance in gender diversity makes it difficult for women to succeed. Women report frustration at being outnumbered and invisible in men-dominated groups. As men and women have different norms for socialization, the lack of peer parity, role models, and role stereotyping further degrades their self-confidence [14]. On the other side, the literature says that women who experience peer parity engage faster with the project or community [6].

Toxic culture: Microaggressions and sexist behavior are not uncommon in tech. Besides having their voices suppressed in technical discussions, women are subjected to various diminishing comments, such as that their inclusion in the team is a result of

diversity initiatives and not a reflection of their skills or capacity [14]. Despite proving their competency in their code/project areas, women are rarely asked their opinion [16]. In a recent survey [4], about one-third of the women contributors reported experiencing written or spoken language that made them feel unwelcome in OSS interactions. Discriminatory expletives, swear words, and negative critiques often used in code reviews and mailing lists are particularly insulting to women. Even finding mentors can be a challenge, since upon discovering their mentee's gender, men mentors can treat the relationship as a dating opportunity.

Impostor syndrome: The hostile toxic culture that often pervades tech communities leads to those in the minority doubting their abilities. Impostor syndrome, wherein individuals struggle to internalize their accomplishments, results not from an individual's inadequacy, but from systemic oppression [5]. Despite being knowledgeable and professionally well-settled, women may be more reluctant to publicly display their work and speak out in meetings and may create fake accounts to hide their gender when they are new to an online community.

Prove-it-again: Women report the need to repeatedly show competence and expend extra effort to be heard when competing with men. Despite accomplishing exceptional work, women feel they need to prove themselves once again when receiving a task that is considered more important or risky and when there is someone else overseeing their work to guarantee that it will be done. Women in tech also feel they have "*no room to slip [up]*" [14]. Prove-it-again involves constantly providing more evidence to demonstrate competence.

Maternity wall: The maternity wall is a common stereotype faced by women and describes the experience of mothers whose coworkers perceive and judge them as having made one of two choices: to continue to work and neglect their family or to prioritize family over work, making them less reliable in the workplace. Thus, when women in tech have children, they tend to receive fewer responsibilities. Given the common belief (stereotype) that mothers are not capable of handling much work, women sometimes "*surprise colleagues that they are able to handle it all.*" This is clear in cases when women are asked to step down from their roles after returning from maternity leave [14].

Glass ceiling: The glass ceiling describes a corporate world phenomenon in which minorities' access to top-management positions is blocked by tradition or culture [9]. It is an invisible structural barrier that prevents minorities from career advancement [11]. In addition to the gender wage gap, women often report feeling that they have nowhere left to climb in the ladder, working harder to achieve the same positions

as men while having their ambition discouraged. Moreover, women report they are disproportionately unsupported to reach higher positions because corporate politics are played by men and men lift only their counterparts to the top [14]. The barriers to future career opportunities discourage women from continuing in the tech industry. For example, studies have found women's participation to drop from 14.2% (participants of Google Summer of Code) to about 4.3% as core developers in OSS; that is, they are better represented earlier in the joining process than in core roles.

Work-life balance: Societal role expectations, women's career ambitions, and the nature of the IT industry challenge the way women manage their professional and personal lives. When working extra and long hours, women feel more stressed and have trouble disconnecting from work, which impacts their household tasks and well-being. Women often are the primary caregiver for a family and perform a larger share of the household labor. Many people believe that working from home is easier for caregivers. However, even when working remotely, women in tech face pressure to work extra hours and face consequences for not giving in to this pressure: if they miss after-hours meetings, they are excluded from decisions made during them and are perceived by others as "*lacking in teamwork*."

Most challenges women face in the software industry have a parallel in OSS. However, some of them manifest differently. In OSS, women face bias when they explicitly identify themselves as women; in the software industry, women report a lack of recognition and face the maternity wall. In OSS, women report difficulties in finding a mentor, but onboarding is usually not as challenging in the industry.

Actions to Mitigate the Challenges

The aforementioned challenges that push women away from the tech industry are, to some extent, known. However, it is important to take action to increase women's participation and create a more diverse and welcoming environment. In the following, we will focus on actions reported in the literature, illustrated in Figure 4-2, and how they help reduce the current challenges.

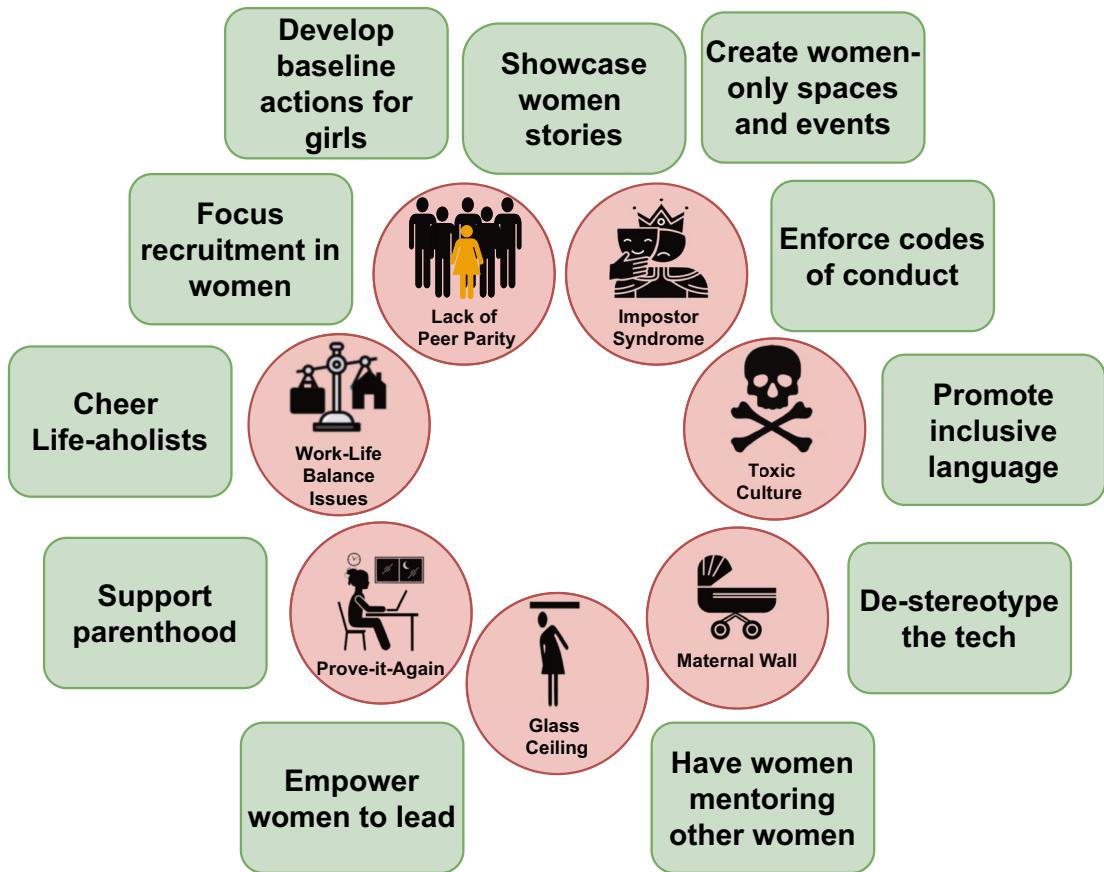


Figure 4-2. Actions to mitigate challenges faced by women [14, 15]

Women are more inclined to join companies or projects that have more women involved. This strengthens their belonging to the project. It is important to **promote awareness about the presence of women** in the project or company to help attract more women and reduce the feeling of alienation. For example, an online dashboard can be used to present the number of women participating, showing what kinds of roles they play, how long they have been involved, and information about their pathway in the project or company.

This can highlight not only that women are welcome at the project or company but that there are role models to be followed there. For example, companies can **promote women to leadership roles** as a way to empower a larger cohort. First, this sends a clear message that women belong there and that they are respected. Second, having women in decision-making positions can ensure that their voices will be heard. This would,

for example, make it easier to communicate the needs and problems faced by women, since the interlocutor also experiences them. Promoting more women to leadership is important, but it is essential to **empower** women by giving them the necessary authority to play the leadership role. Serving in a high position can be even more challenging for women, as they often lack the support or power to accomplish their strategic goals. As a result, women leaders often experience shorter tenures than their peers who are men. When recognizing women's achievements, projects and companies should provide the social attraction women seek to overcome their competence-confidence gap.

Inspiring actions can include **showcasing** by publicly celebrating women's accomplishments in blogs, project homepages, and social media and organizing events with women speakers. These actions can attract more women and retain women who are already employees or contributors. Considering that this media exposure can include women's posts and pictures, it can also help de-stereotype the software developer, which is usually associated with images of men. Two examples of this strategy are the "Women That Build Award"² in "Women Who Code"³ and "Women in Tech,"⁴ which highlights a woman who has excelled in her work in the STEAM industry. Women can nominate themselves or be nominated by other people.

Having women recruiting women may help **focus the recruitment process on women**. Moreover, making job opportunities attractive to women's needs, creating more part-time positions, reserving positions prioritized for women, and advertising job openings to women's groups are actions that help focus the opportunities on women. If no women are involved in the recruitment process, the pipeline may be easily broken, which would affect the recruitment.

A girl should enter the tech pipeline when she is at school by taking preparatory courses, becoming experienced in the use of computers, and otherwise preparing for undergraduate college degrees in STEM. Further along the pipeline – and depending on the educational system – a young woman majors in computer science, and after that she graduates from a computing discipline and, then, enters the job market to be recruited. **Promoting baseline actions**, like schoolgirls' events for girls in STEM, can inspire vocation in girls and raise awareness about how the organization supports women's growth. The baseline actions should be active on girls'/young women's social media (e.g., Snapchat/Instagram).

²<https://womensawards.globant.com/>

³www.womenwhocode.com/

⁴<https://wearexena.com/awards/>

Since girls and adult women need to see themselves in tech for actions to have an effect, we need to **de-stereotype the tech** and avoid the *feminization* of specific assignments, like those relating to nontechnical roles. More images of women should be used in technical advertisements, and women should not be only allocated to procedural tasks. Organizations should offer women to play all kinds of roles that can challenge their skills. Inclusivity also includes adjusting communication styles and de-biasing software. Another way to de-stereotype is by de-biasing software through the GenderMag technique. GenderMag uses personas and a specialized Cognitive Walk-Through (CW) to systematically evaluate software and make them more inclusive of women's cognitive styles [3].

Some actions are essential to deter women from leaving a tech position. The first is to cultivate a welcoming and supportive environment and avoid sexism. **Women-only groups and activities** bring a safe space for women to express feelings and opinions and are helpful to foster discussions and support networking. An online women-only forum can be an easy first step, and the exchanged content can be helpful in monitoring situations when women need support. This strategy can be part of more significant efforts to encourage and welcome women, which can include mentoring or inviting women to contribute to specific activities. Since many projects still have only a few women, groups can facilitate interaction between women from different projects. Trainings and booklets can help **promote inclusive language** and avoid gender pronouns and terms that assume that people are all one gender or one demographic (e.g., using the term "guys"). Bots can monitor written communication and proactively correct gender pronouns.

Mentorship can help women to find the assistance and support they need, particularly by having **women as mentors for other women**. Mentors need to be supportive, friendly, respectful, and encouraging. Besides joining ongoing support groups, women can also be assigned to formal mentors for regular one-on-one meetings. A mentoring program can include a plan for different women leaders to discuss their career trajectory and the benefits and challenges of holding their job. Women could share their techniques for managing time, balancing family and career demands, highlighting how they regularly learn new skills, making themselves heard by men, ignoring disruptions, and coping with criticism or insults.

Besides training and assisting women, the workplace itself needs to be safe. **Developing and enforcing a code of conduct** helps mitigate Tightrope effects⁵ by assisting projects in articulating good behaviors for all members. The code of conduct comprises the collective norms of a community that shape the culture of collaboration and the community's expectations and values and aim to create a friendly and inclusive community. While having a code of conduct will not prevent sexism, it indicates to any men who display sexist behaviors that such actions will not be tolerated. The code of conduct needs to be enforced among the project members. Organizations should implement mechanisms to monitor communication, implement the code, and show that violations have consequences. Provide training that covers essential topics, including advocating for women facing disrespect in meetings, navigating workplace bias, and fostering awareness about microaggressions. These are just a few examples of foundational training elements that empower participants to create a more inclusive and respectful professional environment.

Work and family are often the two most important domains in a person's life, and their interface has been the object of study for researchers worldwide. As women assume the role of working professionally in addition to their traditional role of homemaker, they are under great pressure to balance their work and personal lives. The societal role expectations, women's career ambitions, and the nature of the IT industry challenge the way women manage their professional and personal lives. The COVID-19 pandemic and the need to work from home cast new light on these issues. While it brought more flexibility to many workers, it also brought new challenges. For a great share of the population, it became hard to separate personal and professional life. Women felt this more than men, given the aforementioned societal expectations.

Supporting parenthood is crucial for women who are caregivers. Maternity leaves usually relate to each country's law. Some countries mandate a longer maternity leave, while others mandate a shorter one. Besides sponsoring childcare and children's education, organizations and projects can adjust the duration of the maternity leave beyond the relevant country's laws by offering additional paid leave. Moreover, projects can provide psychological safety with a policy to guarantee the same position for women who return from maternity leave while adding flexibility in work hours. Extra hours should be discouraged, and, instead, projects should **cheer those who are centered on their well-being**. Still, projects with global teams face difficulty finding suitable group

⁵The term *Tightrope* is usually associated with the circus, where a circus performer balances on a stretched rope. Following the analogy, the term can refer to the narrow band of socially acceptable behavior for someone, in our case, women.

meeting times, as people live in different places and time zones. Cultivating a culture of mutual respect can help avoid the “invisible punishment” for those who cannot stretch when scheduling meetings. Sabbaticals promote well-being and increase future productivity through fostering fresh ideas. Implementing sabbaticals allows for paid leaves for personal and professional development.

Not a Conclusion: The Road Ahead

In this chapter, we provide a snapshot of the challenges faced by women in tech and, more importantly, the actions that may alleviate these challenges. However, there is still a long road ahead at the research, practice, and societal levels to make the tech industry more diverse and welcoming.

The literature reports a diverse set of challenges faced by women, but there is still a big gap regarding how it connects to the reasons why women leave (or avoid) the tech industry. Theoretical understandings can help create more effective, longer-term solutions [10]. One of the few studies that have used theory to explain these phenomena found that social capital can support the long-term engagement of both men and women in OSS projects and that when team members have more diverse programming language backgrounds, women are less likely to leave the project. More broadly, we need more research about why a large portion of women who study STEM do not join the tech industry so we can create more effective, longer-term solutions based on these phenomena. Being aware of the challenges that women face, educators can address the underlying issues causing these challenges to improve students’ (all gender) awareness of biases and discuss possible mitigation actions.

Historically, the social differences influenced by gender roles (i.e., the roles that men and women are expected to occupy based on their sex) may be amplified because of the gendered division of housework and childcare tasks, especially for mothers of young children. Impostor syndrome, sexism, lack of peer parity, prove-it-again, glass ceiling, and work-life balance issues were challenges reported by women in different software development contexts, including in large companies and open source projects [14, 15]. Some challenges surpass the organization and relate to the local, regional, or company/community culture. These challenges bump into society and often contribute to this cultural legacy. One example is the “trailing spouse” – when a person follows their life partner to another city because of a work assignment. Moreover, during the COVID-19 pandemic, a longer “double-shift” in the context of lockdown and limited availability

of childcare services contributed to stress, anxiety, job insecurity, and difficulty in maintaining work-life balance among women with children [7]. It means that all of us as a society have to do our share to evolve and change this cultural legacy that hurts women more than men.

Although several actions to increase women's participation have been proposed in the literature, few works present scientific evidence about their effectiveness. For instance, few studies evaluate the effectiveness of the "code of conduct" [12, 13], even though it is one of the most-cited actions to promote women's participation. Evaluating the effectiveness of actions can be challenging, as communities need to have consistent measurements before (baseline), during, and after their implementation [8]. For example, although OpenStack created the Women of OpenStack Working Group (which included educational sessions, professional networking, mentorship, social inclusion, and enhanced resource access), the OpenStack Foundation lacked baseline information about the involvement of women [8].

Future work can also understand the intersectionalities of contributors to shift the focus away from individual-level conceptualizations of gender in tech and toward structural examinations that take into account the power dimensions of race, class, culture, sexuality, caregiving responsibilities, disabilities, and other demographics and how different systems of oppression are mutually constituted and work together to influence women's participation. Sensitivity to intersections enhances insight into the issues of inequality, thus maximizing the chance of social change [2].

Another open avenue for researchers is investigating why women do not join the tech industry. The perspective of women who are outsiders (and possible future insiders) and understanding what would entice them to join tech can bring new insights to attract women to the field.

Finally, "unting the mooring ropes" of sociocultural problems is difficult. The cultural structural sexism present in society is mirrored in the professional environment. There is still a long work ahead for the software industry, and for us, as a society, to create a more diverse and inclusive environment. We hope our results will enlighten actions toward reducing the perceived challenges and increasing awareness about the structural and cultural hurdles imposed on women that negatively influence diversity in the software industry.

Bibliography

- [1] Khaled Albusays, Pernille Bjorn, Laura Dabbish, Denae Ford, Emerson Murphy-Hill, Alexander Serebrenik, and Margaret-Anne Storey. The diversity crisis in software development. *IEEE Software*, 38(2):19–25, 2021.
- [2] Doyin Atewologun. Intersectionality theory and practice. In *Human Resource Management, Organizational Behavior, Research Methods, Social Issues*. Oxford Research Encyclopedia of Business and Management, 2018.
- [3] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephan Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.
- [4] Hilary Carter and Jessica Groopman. The 2021 Linux Foundation Report on Diversity, Equity, and Inclusion in Open Source. <https://bit.ly/3PmbKu5>, 2021. Accessed on July 20, 2022.
- [5] Pauline Rose Clance and Suzanne Ament Imes. The imposter phenomenon in high achieving women: Dynamics and therapeutic intervention. *Psychotherapy: Theory, Research & Practice*, 15(3):241, 1978.
- [6] Denae Ford, Alisse Harkins, and Chris Parnin. Someone like me: How does peer parity influence participation of women on stack overflow? In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 239–243, IEEE, 2017.
- [7] World Economic Forum. Global gender gap report 2021. Technical report, World Economic Forum, 2021. Accessed on October 18, 2021.
- [8] Daniel Izquierdo, Nicole Huesman, Alexander Serebrenik, and Gregorio Robles. Open-stack gender diversity report. *IEEE Software*, 36(1):28–33, 2018.
- [9] Janet Cooper Jackson. Women middle managers perception of the glass ceiling. *Women in Management Review*, 2001.
- [10] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. Going farther together: The impact of social capital on sustained participation in open source. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 688–699, IEEE, 2019.

- [11] Sakshi Sharma and Parul Sehrawat. Glass ceiling for women: Does it exist in the modern India. *Journal of Organization & Human Behaviour*, 3(2–3):9–15, 2014.
- [12] Vandana Singh and William Brandon. Open source software community inclusion initiatives to support women participation. In *IFIP International Conference on Open Source Systems*, 68–79, Springer, 2019.
- [13] Parastou Tourani, Bram Adams, and Alexander Serebrenik. Code of conduct in open source projects. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 24–33, IEEE, 2017.
- [14] Bianca Trinkenreich, Ricardo Britto, Marco A. Gerosa, and Igor Steinmacher. An empirical investigation on the challenges faced by women in the software industry: A case study. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 24–35, IEEE, 2022.
- [15] Bianca Trinkenreich, Igor Wiese, Anita Sarma, Marco Gerosa, and Igor Steinmacher. Women's participation in open source software: a survey of the literature. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(4):1–37, 2022.
- [16] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Perceptions of diversity on git hub: A user survey. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, 50–56, IEEE, 2015.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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.

PART II

Inclusive Software: How Inclusion Impacts Products

CHAPTER 5

How Users Perceive the Representation of Non-binary Gender in Software Systems: An Interview Study

Negin Hashmati, Chalmers University of Technology, Sweden.*

Birgit Penzenstadler, Chalmers University of Technology, Sweden.

We research the users' perception of the representation of gender in software through an interview study. We capture the perception of how non-binary people are being represented in different software systems. We aim to understand how users perceive the value of their gender being collected in the software they use, where it is not necessary to be captured, and how to make our systems more gender-inclusive. Through thematic coding, the results of the study present various themes related to gender representation in software. It is important that software engineers take the issues of gender representation seriously. This could imply not to register a user's gender in the first place. Where relevant, it should be done in a way that makes sure that all gender identities feel included in the digital world.

Introduction

There are multiple aspects of inclusivity that should be considered in software design, such as the user's gender, age, ethnicity, and special abilities. This study focuses on the gender aspect, namely, how gender is represented in software – how software systems ask about a user's gender and how that gender information is used by the software. The use of gender in software varies depending on the system, where it is not relevant in some systems and crucial in others. It is therefore in our interest to not only understand how gender is represented in software but also where gender is a relevant factor in a system [1].

Current literature looks at gender in software from various different aspects, such as conceptual modeling for gender-inclusive requirements [2], how a non-binary person was discriminated through different websites when trying to register their gender [3], how the quality of requirements is affected depending on the stakeholder's gender [4], the difference between men and women regarding privacy concerns in e-health applications [5], and how men and women felt about different gendered aesthetics on a website [6]. There is a gap in the literature with regard to how users feel that their gender is being represented in software, as well as what the software engineers can do to make the systems they create be more gender-inclusive.

In this study, the definition of *gender* is the gender a person identifies as, that is, a person's gender identity. People may experience a gender identity on a mental and emotional level that cannot be captured by documenting a physical expression.¹ The contribution is to understand how gender representation in software systems is perceived by users to help practitioners understand how to make software more inclusive already at the stage of requirements elicitation.

Related Work

Spiel [3] writes about their experience as a non-binary person and how they, over the course of 1.5 years, faced discrimination with various digital interfaces, detailing their thoughts and feelings on how they were represented across various systems. They propose a number of solutions for how to include non-binary people in software (e.g.,

¹See also <https://transequality.org/issues/resources/understanding-nonbinary-people-how-to-be-respectful-and-supportive>

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

more fields in the databases) and that computer science education take these issues seriously and make sure that systems are created more inclusively.

Nunes et al. [2] have created a model for conceptualizing and helping developers understand how their software can be made more gender-inclusive adapted to each organization and system's goals. Their aim was to guide requirements engineers to creating more gender-inclusive requirements using a framework to help practitioners with recognizing gender bias in their systems.

Ehrnberger et al. [7] recreated a drill and hand blender to talk about and challenge gender norms surrounding everyday items. Clarkson et. al. [8] discuss the various aspects of inclusive or universal design and how practitioners can work with them. Within the area of user experience (UX) design, there are many voices raising about making software and designs more inclusive [9, 10, 11], giving guidelines and suggestions for how software can become more gender-inclusive.

Rowan and Dehlinger [5] look into the difference between men and women with regard to privacy concerns for e-health mobile applications, and women reported higher concerns than men. The authors suggest that health applications should provide different services related to privacy as the concerns and behavior of the users differ depending on their gender.

Metaxa-Kakavouli et al. [6] created two different web interfaces that displayed the same content for a computer science course, where one page had aesthetics that were perceived as gender-neutral and the other had aesthetics that were perceived as masculine. Results showed that women reacted negatively to the more masculine website.

Criado Perez [12] details how data bias and the gender data gap have affected and continue to affect women in various areas of life. The data bias can be seen in everything, from software to city planning and the design of restrooms. The book is not strictly about software and software engineering, and the study at hand therefore compliments it by looking at gender representation solely in software.

Further studies cover how gender differences impact building social goal models [13], how algorithm bias can affect the use of user feedback in app stores [14], and how the software engineers' human aspects can affect requirements engineering [15].

Study Design

Purpose of the study: The aim of this interview study [16] is to explore and understand how gender representation in software is being perceived. The study aims first at understanding how users perceive the value of their gender being collected in the software they use, how it is currently represented and what effects that has, and what experiences users have with gender being represented in software systems. Informed consent from the participants was obtained by first explaining the study and the planned use of the data before the participants made a decision about whether they wanted to take part or not and then signing the consent form. In the interviews, their answers were anonymized in the interview transcript, and the interviewer collected their email address only to send the finished transcript to the interviewee for them to read over and give their permission to use. The research protocol was reviewed at university level and not required to go through further review at the national ethics authority.²

Research questions: The two research questions are (RQ1) What is the perception of representing gender in software systems? and (RQ2) How are the options of representing non-binary gender in software systems perceived? RQ1 aims to look at what people (the interviewees) think of how gender is represented in different types of software systems. RQ2 aims to look at the perceptions of how non-binary people are being represented in software systems. It is partially answered with the help of interviewees who do not identify as non-binary, but put themselves in the shoes of someone who identifies as it. This design choice was made as (1) we were not able to recruit what we would have deemed a sufficient number of non-binary interviewees and (2) the ability of people who identify as female or male to put themselves into the perspective of a non-binary person might indicate the feasibility of a software engineer doing the same for a gender they do not identify with.

Piloting of interview questions: The interview questions (Table 5-1) were piloted with the help of interviewee 1, who completed the interview and then gave feedback.

²“Under the statutory rules, research ethics review by the Authority is a mandatory precondition for commencement of all research and clinical trials planned in Sweden that involves physical intervention, on living and deceased persons alike, is carried out with a method that aims to affect the research participant physically or mentally, or involves an obvious risk of harm to them in body or mind (...). The Swedish Ethical Review Authority neither can nor may issue any advance ruling on the question of whether an ethical review is required. It falls on the entity responsible for the research to decide whether an initial application needs to be submitted” (<https://etikprovningsmyndigheten.se/en/what-the-act-says/>).

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

Interviewee 1 is a professor and equality representative at the university, and we therefore felt that their opinions and insights were valuable. After the feedback was positive, the rest of the interviews were then conducted as planned, and interviewee 1's answers could be used.

Ideally, a more extensive piloting of the interview would have been carried out, but because of the short time frame for this study and the already small pool of interviewees, this was not seen as feasible. The time frame was due to the project being a bachelor's thesis, with a limited time to find and carry out the research. The small interview pool resulted from a convenience sample. We reached out to students on a Discord server with around 400 members, as well as other personal acquaintances. As we were asking about non-binary gender, the interviewees either identified as non-binary (minority) or had relatable expertise or experience so they felt comfortable answering questions around the topic, for example, equality representative, researcher of gender studies, etc. Some participants struggled with coming up with examples for *government software* in Q4, and the question was therefore clarified to *government or tax software* instead.

Data collection: The subjects for the interviews were chosen through convenience sampling where they were either personal acquaintances or people who had some kind of interest in the subject of the study and wanted to take part in it. The majority of the participants were European, with a couple of interviewees from the United States and Brazil. The interviews were conducted in a semi-structured manner either in person or through video calls. The reason for using semi-structured interviews was to give the participants the option of speaking freely and for the researcher to be able to ask follow-up questions to interesting points they made. The participants consented to recording, and their answers were then transcribed and sent back to them for feedback before using thematic coding. All their data was anonymized while creating the transcripts. The interviews lasted up to 30 minutes, with most of them averaging 20 minutes.

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

The interview questions were derived from the research questions, as detailed in Table 5-1. Q3 was only asked if the participant did not identify as non-binary, because for non-binary participants, the answers to Q1 and Q2 were deemed sufficient to answer RQ2. We are aware that Q3 poses a threat to validity since we aim to understand perceptions of non-binary people using software systems. In Q4, four different examples of software systems were given: dating software, tax or government software, and medical software. These were chosen as examples as they cover a range of different types of software systems where gender potentially matters and therefore give an overview of the issues with gender representation across different fields. The participants were first asked a set of demographic questions, including how they identified their gender. Table 5-2 details all the participants' demographic information.

Table 5-1. Interview questions mapping to research questions

Interview Question	RQ
Q1. When using software, how do you think people might feel discriminated against because of their gender?	RQ1
Q2. In what way could they feel discriminated against? In what type of software?	RQ1
Q3. Imagine you were identifying as non-binary. Can you imagine that you might feel discriminated against by some software systems? Which ones in that case and how? Why do you think you would be discriminated in that way?	RQ2
Q4. Think of using dating software, tax or government software, and medical software, do you think people would be discriminated against when using them?	All
Q5. Do you think it is relevant to register the gender of people in software systems at all? In which cases is it relevant?	RQ1
Q6. Do you think there are software systems where it is important to disclose a person's gender, but right now it is not being asked for? If so, which ones?	RQ1
Q7. Are there software systems where you feel it is not important to disclose a person's gender but where you currently have to? If so, what are examples?	RQ1
Q8. Do you think it is important that more genders than the binary male/female are considered in software? Why?	RQ2
Q9. Do you have any ideas on how to make software systems more gender-inclusive?	All
Q10. Anything else you would like to add?	All

Table 5-2. Interviewees' demographic information

Interviewee	Gender	Pronouns	Profession
1	Male	He/him	Senior lecturer
2	Female	She/her	Professor
3	Female	She/her	Student
4	Female	She/her	Student
5	Male	He/him	Student
6	Female	She/her	Professor
7	Female	She/her	Associate professor
8	Female	She/her	Student
9	Male	He/him	Student
10	Male	He/him	Student
11	Female	She/her	PhD Student
12	Non-binary	They/them	Student
13	Androgynous	They/she	Director of marketing
14	Female	She/her	Lecturer
15	Non-binary	They/them	Postdoc

Data analysis: The interview data was analyzed through thematic coding following the guidelines by Saldaa [17]. The coding was carried out through a mixed approach, using both inductive coding (creating the themes and codes as we read through the transcripts) and deductive coding (developing a set of themes and codes before starting the coding). The deductive themes were derived from the interview questions presented in Table 5-1. The coding was carried out by the first author and verified by the second author.

Results

Answering research questions:

RQ1: What is the perception of representing gender in software systems? Summarizing the results from the thematic coding, the state of the practice in representing gender in software systems is that more can be done with regard to representing gender. Many of the interviewees said that the current gender options are inadequate and that software asks for gender when it is not necessary.

RQ2: How are the options of representing non-binary gender in software systems perceived? The non-binary people who were interviewed for this study said that they often felt that their gender was not accurately represented and that the option of “other” gender was not useful and discriminatory. They mentioned that certain types of software, namely, medical software, should have other ways of presenting their gender instead of the binary male and female. The responses from all subjects were considered when answering this research question. This could therefore bias the results (see p. 16), but the answers were combined because there was a large overlap in the answers from the interviewees who identified as non-binary and the ones who were imagining that they were identifying as non-binary. The overlap was seen in the answers for Q3 from the interviewees who did not identify as non-binary and in the answers for questions Q1, Q2, and Q4 from the people who did identify as non-binary.

Participants: The subjects of the study were selected through connections and personal acquaintances. Out of the fifteen participants, seven worked in academia and seven were university students across different fields of study. One participant worked as a director of marketing. Four of the participants identified as male, eight identified as female, two identified as non-binary people, and one identified as androgynous. This information can also be found in Table 5-2.

Results from thematic coding: The thematic coding was carried out by doing multiple iterations, reading over the transcripts, and assigning themes and codes to pieces of text. An overview of the deductive themes is presented in Figure 5-1, and an overview of the inductive themes is presented in Figure 5-2.

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

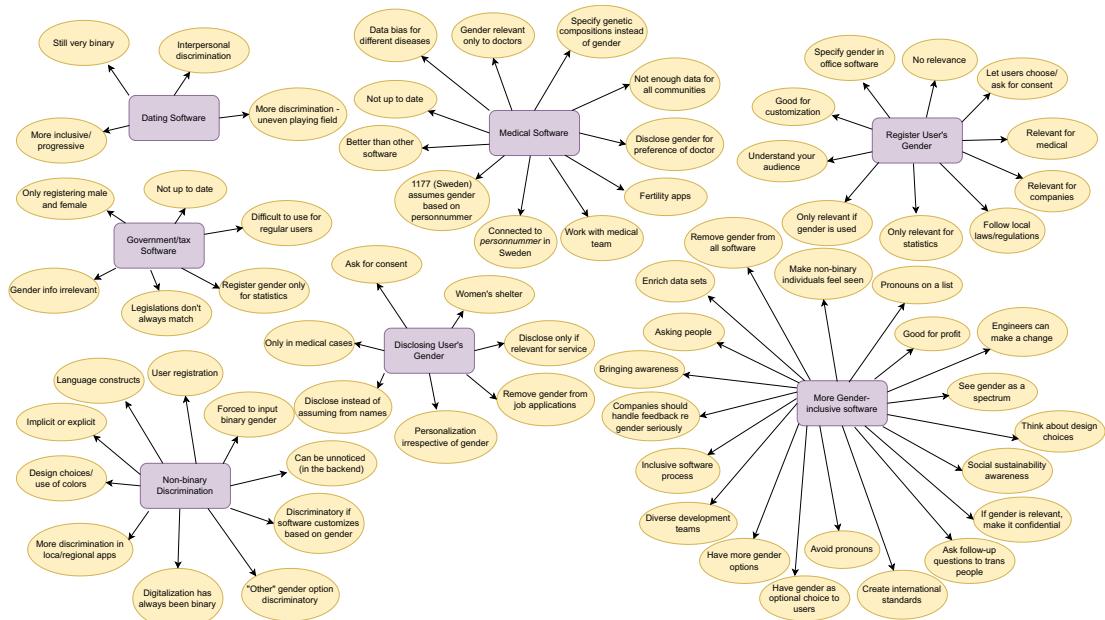


Figure 5-1. Overview of deductive themes and codes

Deductive themes: *Non-binary discrimination* often happens in language constructs in software, according to many interviewees. They explained that using a gendered language when not necessary adds nothing but further excludes non-binary individuals. One interviewee mentioned that they might be forced to input a binary gender in places that do not offer other options. Interviewees also mentioned that this often happens in the user registration state and that digitalization has always been very binary. Similarly, interviewees noted that the option of “other” as gender is seen as discriminatory. One interviewee explained that it is because users will feel not seen or acknowledged and will feel that they are being put in an exception box if they choose the “other” option and thus will more likely not go for that option in surveys, forcing them to choose a gender option that may not be representative of the gender they identify as. Other interviewees mentioned the design choices and uses of colors as a way of cementing the binary gender norms.

For *dating software*, interviewees said that they are generally more inclusive and progressive compared with other software. One interviewee pointed out that dating software is still very binary. For *medical software*, some interviewees said that they are generally not “up to date,” while others said that they are better than other types of software. One non-binary interviewee explained that the information regarding a

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

person's gender is only relevant to the doctors and that they would work with a medical team if they were developing this kind of software. The *Personnummer in Sweden* was mentioned in relation to medical software, where one interviewee mentioned that the medical system 1177 [18] assumes a person's gender based on their *personnummer* when they sign in. Several interviewees mentioned that there is not enough data collected for all communities of people, which leads to data bias, for example, for different diseases. One interviewee suggested that instead of medical software asking for a person's gender, they should ask about their genetic composition instead.

For *government/tax software*, interviewees mentioned that they often only register male or female and no other genders and are therefore not as up to date as other types of software. One interviewee mentioned that registering a person's gender in governmental software should only be done for statistical purposes and that the gender information is not relevant in this type of software otherwise. Another interviewee mentioned that legislation regarding gender does not always match between countries, which can be an issue for people who live and work in one country but have a citizenship in another country.

Registering the user's gender as a deductive theme was asked in Q5 in the interviews. Here, almost all interviewees claimed that it is generally not relevant to register users' gender information. Some explained that it could be useful to understand your audience and for the companies. Other interviews said that registering a user's gender could be relevant in medical cases, for example, if a patient is unconscious when arriving to a medical institution. Registering a user's gender should follow the laws and regulations in a country, one interviewee explained. Another interviewee explained that registering of gender could be good if the software is being customized to the user.

Disclosing the user's gender should be a choice, or that the software asks for consent before disclosing it. Two interviewees said that the gender should be completely removed from job applications as that only feeds into prejudices. One interviewee said that the software should offer personalization irrespective of a user's gender and focus on other information they have provided. Some interviewees said that disclosing of a user's gender should only be done in medical cases. Disclosing a user's gender should not be done unless it is relevant for the service the software is providing, according to one interviewee. Another said that it would be better for the software to ask for and disclose gender rather than trying to assume people's gender based on their names.

For *more gender-inclusive software*, the primary solutions according to the interviewees were to have more gender options and make non-binary individuals feel

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

seen. Interviewees suggested having diverse development teams and an inclusive software development process and, also, bringing awareness to the gender perspective by asking people about how they would prefer to have their gender represented in software and that gender should be seen as a spectrum and that pronouns should be avoided altogether. Another interviewee suggested that to make it more inclusive, a selection of pronouns should be presented on a list to users and that people who have chosen that they identify as trans are asked follow-up questions. With regard to the data bias, interviewees said that it is important to enrich the data sets to have a better representation of all the different types of people that use the software. One interviewee said that if gender is relevant to have, it should be made confidential:

Systems can be designed in such a way that this information is not directly accessible to anyone who just happens to be maintaining the systems. It can start with encrypting the data, what is sensitive, and storing the data without having a name attached to it, just ID number and references.

—Interviewee 13

Inductive themes: *Discrimination due to UI* was one theme that appeared in the interviews. Four codes were found: *use of colors, graphics, pronouns in messages, and marketing through colors*. The interviewees explained that the choices of colors were made based on gender, where darker tones were generally seen as targeting men. Other aspects of the user interface, such as graphics and messages, could also be seen as discriminatory, namely, where pronouns were used in pop-up messages and the default pronouns were he/him.

Some interviewees mentioned *discrimination due to anonymization*, where they explained that people using different forums and communities would have an easier time spreading harmful comments. One interviewee mentioned that if a non-binary person were to disclose their gender in an anonymized community, that might be more noticeable and they are at a greater threat of receiving hurtful comments.

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

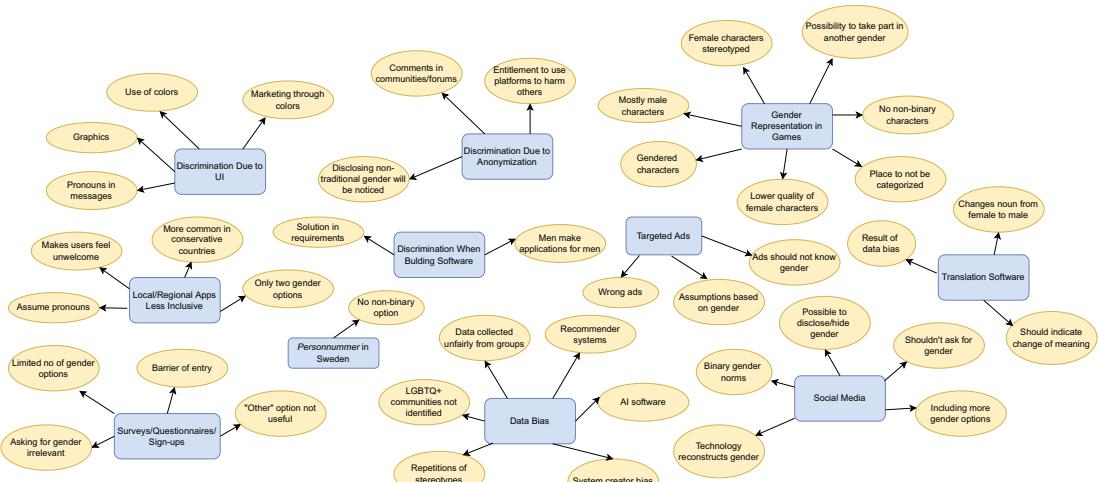


Figure 5-2. Overview of inductive themes and codes

Gender representation in games was a theme that was mentioned multiple times. Interviewees explained that there is a large amount of male characters in games, together with many characters being gendered in general and female characters often being stereotyped, and that there are no non-binary characters in games. One interviewee mentioned that games having gendered characters could be a possibility for people to take part in a gender different from the one they identify as and thus could lead to a better understanding of different genders.

Some interviewees mentioned *discrimination when building software*. One interviewee mentioned that the solution to overcoming gender discrimination in software lies in the requirements part and that it is at that stage where inclusivity needs to be taken into account. This is in line with what another interviewee said: that the engineers really can make a change regarding these issues. Other interviewees mentioned that men often make applications for men.

Less inclusive local/regional applications were mentioned a few times. The interviewees explained that it is more common in conservative countries that applications only have two gender options, one for male and one for female. They further explained that users feel more unwelcome when using those applications when their gender is not accurately represented.

Another theme that came up in the interviews was *data bias*. Interviewees said that many systems came with creator biases, where data was not collected fairly from different user groups. One interviewee specifically mentioned that LGBTQ+

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

communities were often not identified in the process of collecting data. The same interviewee mentioned recommender systems as places where the data bias becomes clear. Another interviewee mentioned AI software as inherently biased.

Targeted ads were another theme. It was mentioned in relation to assumptions being made based on a person's chosen gender, as well as people getting ads that did not correspond. One interviewee mentioned that when the non-binary are forced to choose a binary gender when signing up, the ads they get based on their gender are wrong. Some interviewees mentioned that the ads should not know people's gender at all. Another interviewee explained that this type of discrimination could be either implicit or explicit and that it could be unnoticed where the binary genders are still being used in the back end:

For example, when Facebook had this multitude of options that you could provide, but in the back end for advertisers, it's kind of slotted people into binary genders again.

—Interviewee 15

Surveys/questionnaires/sign-ups refer to the different places where users have to sign up. Interviewees mentioned that these forms often have a limited amount of gender options. As a consequence, people may not exist in systems because they are hindered by the barrier of entry, as one of the interviewees mentioned.

Some of the interviewees mentioned *social media* as a place where technology reconstructs gender. They explained that the binary gender norms are present in the different social media platforms. One interviewee felt that the platforms should not ask for gender at all, while another said that if they do need to ask for gender, they should include more gender options for the users.

One interviewee mentioned *translation software* as an example of software that produces wrongful information as a result of data bias. They explained that the software changes the noun of a word from female to male when translating it. For example, the word "researcher" as a female noun in one language would be translated to "researcher" as a male noun in another language. The interviewee said that these types of software systems should have some indication to the user that the meaning of the word has changed.

The theme *Personnummer in Sweden* refers to the system in Sweden where everyone registered in the Swedish Population Register receives a personal identity number [19]. The number consists of a person's birth date in the form YYYYMMDD, followed by four

digits where the third one is decided by the person's sex. The digit is odd if a person's legal sex is male, and if a person's legal sex is female, the digit is even. As of now, there are no options for a non-binary *personnummer*, but a motion has been sent to the Swedish government about creating a third legal sex as well as gender-neutral personal identity numbers [20]. As the information structure is insufficient in the first place, representation cannot be accurate either. The interviewees explained that because the system is inherently binary, it leads to assumptions:

And they have in medical areas the assumption that if the personal number says that you are this gender, we have to add it in your journal that you also can read, but it's not really important actually.

—Interviewee 12

Discussion

Software development process: *Discrimination when building the software* is a theme that is very relevant to software engineers. The interviewees explained that men often make applications for men, and one specifically mentioned that discrimination can happen already at the stage of requirements writing:

[...] I think in the domain itself as well, there is so many different aspects of how we capture the requirements of the system that are not aware of the systematic discriminations that these systems can reinforce.

—Interviewee 1

One possible solution to this would be to use the GenderMag method [21] in the software development process. By using the process and provided set of materials, that is, the developed personas, companies and software engineers could make sure that the software they are developing is gender-inclusive. GenderMag is a good solution because it easily provides the developers with personas that they should consider for their software and thus can help them find where their software displays gender bias. Aside from using GenderMag, developers could use Hidellaarachchi et al.'s [15] paper to understand how human aspects of the engineers can affect the requirements they write. They did conclude that the gender of the developers was of lower importance than, for example, knowledge of the domain,

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

but we think that making sure that both the process and the development teams are inclusive leads to the software being more gender-inclusive. More on this aspect can be found in Chapter 10, “Beyond Diversity: Computing for Inclusive Software,” and Chapter 11, “Gender Diversity on Software Development Teams: A Qualitative Study.”

Design of systems: The findings from Costanza-Chock [22] can be incorporated into software systems design. Specifically, we can apply the principles of the Design Justice Network that are put forth in the book.

Let’s take the first two principles as examples. Principle 1: *We Use Design to Sustain, Heal, and Empower Our Communities, As Well As to Seek Liberation from Exploitative and Oppressive Systems* [22, p. 190]. The Design Justice Network encourages designers to move beyond critiquing oppressive systems and actively empowering community. Software engineers have an agency here that is visible in the fitting or not fitting design choices they make, and that agency comes with responsibility.

Principle 2: *We Center the Voices of Those Who Are Directly Impacted by the Outcomes of the Design Process* [22, p. 191]. The idea of “nothing about us without us” is a partial answer to the question of who gets to do design work as an appeal for more inclusivity in software engineering as a discipline. While female and male interviewees in our study provided insightful answers when prompted to put themselves into the shoes of a non-binary person, the participants of our study are not representative of software practitioners in general. To design well for non-binary users means to have one on the development team or as a client stakeholder.

Further principles detail potential impacts in social movements and technological innovation as well as best practices for community-engaged research.

User interface and design: The theme *discrimination due to UI* showed that the choice of colors and other design elements often can make users of certain genders feel unwelcome, which is in line with the findings of Metaxa-Kakavouli et al. [6]. More focus needs to be put on recognizing these patterns of design-related discrimination, specifically when creating the software. If users don’t feel welcomed when opening an application or a website, they most likely will not continue to use the software, which means less profit for the company, but also that the user misses out.

Medical software and data bias: With regard to the *medical software* theme, the most interesting findings are the ones related to data bias. Several interviewees mentioned that data is not collected fairly from different groups of people and this then makes the medical data biased in relation to gender. It can even be life-threatening, as stated by one interviewee:

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

If you think about heart attacks, a lot of the symptoms related to females are different to the ones related to males. And quite often women die from heart attacks because their symptoms are not recognized.

—Interviewee 14

Data bias is something that is prevalent in most software today, and there are a lot of things that can be done to solve it. Our first suggestion is something that was mentioned by multiple interviewees, which is to enrich the data sets. By making sure that the data collected comes from a large and diverse sample of people, it would make the systems detecting, for example, medical problems more inclusive and could in the long run save lives.

Data bias as a theme was mentioned multiple times, also in relation to other types of software besides medical, in line with Criado Perez [12]. She proposes several solutions for minimizing the gender data gap, the most prominent being that women should be included in the collection of data. For that, women need to be included in all aspects of life, and when more women are in positions of power, they remember that women and their needs exist.

Rowan and Dehlinger [5] results about privacy concerns are echoed by one interviewee, who talked about being wary of giving out their information to companies:

[...] I think that it's important to break down barriers and to be inclusive. But I don't necessarily think that part of being inclusive is to harvest and collect all the data we could possibly muster up because then I feel like that delves into a completely different territory.

—Interviewee 8

Non-binary discrimination, registering and disclosing users' gender: *Non-binary discrimination* as a theme was related to RQ2. A result that was quite surprising was how many interviewees mentioned that language constructs were contributing to the discrimination of non-binary people. One suggestion to make the language more inclusive is to follow the Gender Guidelines by Scheuerman et al. [23], and we encourage all practitioners to read Spiel's paper [3], where they detail their experiences with different types of software as a non-binary person and propose several solutions.

The large majority of interviewees mentioned that it is not relevant to register a user's gender at all. While it was not surprising that people thought it unnecessary, we were surprised by the fact that almost all of the interviewees said that they found it to

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

not be necessary. In addition, there are examples where gender bias is a known fact, for example, job application systems [24, 25], but also in tax systems [26], so these may serve as an indicator to not capture gender when not strictly necessary. In light of the General Data Protection Regulation (GDPR), the guideline of not collecting unnecessary data might have even more weight.

A motion has been sent to the Swedish government about including a third legal gender as well as gender-neutral *personnummer*. This is similar to what other countries, such as the United States [27], Germany [28], and Austria [29], have implemented. These countries show that it is possible to include non-binary genders in legal documents, and more countries should follow in their footsteps.

Limitations and threats to validity: The following threats to validity are of concern for the study at hand:

Internal validity: The interviewees were chosen based on if they had an interest in the subject matter regarding their experiences of gender in software. While most of them were in academia, they were at different stages of their studies or were professors in different areas. It was also reasoned that if subjects who had no interest in the topic of the study would be interviewed, they would not give any meaningful answers. This is a trade-off that has to be made between having a slight bias in the results and receiving not meaningful data. Another aspect of internal validity is that Q3 gives results that may not be applicable to answer RQ2. This is because when asking people to imagine a scenario, it does not give a truthful view of how non-binary people experience representation in software systems. This question remains in the study because answers thematically overlapped significantly for all genders.

Credibility: The results of the interviews could threaten the credibility of the study. The interviewee could be biased in giving answers that do not actually reflect their opinions on this subject because they wanted to be helpful to the research. The mitigation strategy for this threat is similar to the one for internal validity, namely, that the subjects were from a variety of backgrounds and that there is a trade-off between having a slight bias and difficulty of saying if the findings are true and receiving not meaningful data.

Dependability: The findings within the study were fairly consistent as many interviewees mentioned similar themes and codes. It is however difficult to say whether the findings could be repeated in a similar study as it depends on the subjects. This threat is not possible to mitigate at this stage, because there is only one study made on

this topic thus far. More studies of similar nature would have to be conducted to be able to say if the findings are congruent and, if so, what made it possible for the results to be similar.

Conclusion

The results of our study point out that registering a user's gender in software systems is often not relevant or even detrimental to the experience of non-binary users. Other important points were recognizing what design choices can be discriminatory for different types of users and that the data sets used for training algorithms need to be more diverse to reduce the data bias that is prevalent in so many software systems. If the software we create is inclusive, it makes our communities and society as a whole more inclusive and welcoming. Perhaps it is best summarized by the following quote from one interviewee:

[...] The software engineers, they really have the power in their hands to change a lot of assumptions and stigmas. [...] People use software all the time, and you can really change the world in a different and positive way where people can feel included. I think that's important.

—Interviewee 12

For how to ask about gender, we suggest to read Chapter 28, “How to Ask About Gender Identity of Software Engineers and “Guess” It from the Archival Data.”

Acknowledgment

Thank you to all the interviewees for your interesting perspectives. We wish to conclude this chapter by acknowledging the privilege that we have, including that we, as CIS women, have been lucky enough to never have had our gender identity questioned. That we were able to conduct this study is further proof of our privilege, coming from good socioeconomic backgrounds, as well as being born and raised in countries that offer free higher education. Despite the privileges we have, we are also no strangers to discrimination and marginalization. We are women in the male-dominated field of software engineering as well as immigrants. We have both experienced racial discrimination and one of us sexual harassment based on gender in different countries we have lived in. This is a motivation for us to support others who experience discrimination and marginalization.

Bibliography

- [1] M. K. Scheuerman, K. Spiel, O. L. Haimson, F. Hamidi, and S. M. Branham, “HCI Gender Guidelines, Context and Motivation,” www.morgan-klaus.com/gender-guidelines.html#Context, 2020. Last accessed on March 8, 2022.
- [2] I. Nunes, A. Moreira, and J. Araujo, “Conceptual modeling of gender-inclusive requirements,” in *International Conference on Conceptual Modeling*, Springer, 2021, pp. 395–409.
- [3] K. Spiel, “Why are they all obsessed with gender? (Non)binary navigations through technological infrastructures.” New York, NY, USA: Association for Computing Machinery, 2021. [Online] Available at <https://doi.org/10.1145/3461778.3462033>
- [4] E. Díaz, J. I. Panach, S. Rueda, M. Ruiz, and O. Pastor, “Are requirements elicitation sessions influenced by participants’ gender? An empirical experiment,” *Science of Computer Programming*, vol. 204, p. 102595, 2021.
- [5] M. Rowan and J. Dehlinger, “Observed gender differences in privacy concerns and behaviors of mobile device end users,” *Procedia Computer Science*, vol. 37, pp. 340–347, 2014.
- [6] D. Metaxa-Kakavouli, K. Wang, J. A. Landay, and J. Hancock, “Gender-inclusive design: Sense of belonging and bias in web interfaces,” in *Proceedings of the 2018 CHI Conference on human factors in computing systems*, 2018, pp. 1–6.
- [7] K. Ehrnberger, M. Räsänen, and S. Ilstedt, “Visualising gender norms in design: Meet the mega hurricane mixer and the drill dolphia,” *International Journal of Design*, vol. 6, no. 3, 2012.
- [8] P. J. Clarkson, R. Coleman, S. Keates, and C. Lebbon, *Inclusive Design: Design for the Whole Population*. Springer Science & Business Media, 2013.
- [9] @rmaanushi_joshi, “Gender-inclusive design is the only way,” <https://uxdesign.cc/gender-inclusive-design-is-the-only-way-968494d5afc2>, June 2021. Accessed on January 12, 2023.

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

- [10] S. Fonseca, “Designing forms for gender diversity and inclusion,” <https://uxdesign.cc/designing-forms-for-gender-diversity-and-inclusion-d8194cf1f51>, April 2017. Accessed on January 12, 2023.
- [11] T. Hunter, “How to make your software more trans-inclusive,” <https://builtin.com/software-engineering-perspectives/trans-inclusivity-tips-developers>, August 2020. Accessed on January 12, 2023.
- [12] C. Criado Perez, *Invisible Women: Exposing Data Bias in a World Designed for Men*. London, England: Chatto & Windus, 2019.
- [13] C. Gralha, M. Goulao, and J. Araujo, “Analysing gender differences in building social goal models: a quasi-experiment,” in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, IEEE, 2019, pp. 165–176.
- [14] E. Guzman and A. P. Rojas, “Gender and user feedback: An exploratory study,” in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, IEEE, 2019, pp. 381–385.
- [15] D. Hidellaarachchi, J. Grundy, R. Hoda, and I. Mueller, “The influence of human aspects on requirements engineering: Software practitioners perspective,. Preprint at *arXiv:2109.07868*, 2021.
- [16] F. Shull, J. Singer, and D. I. Sjøberg, *Guide to Advanced Empirical Software Engineering*. Springer, 2007.
- [17] J. Saldaa, *The Coding Manual for Qualitative Researchers*. London: Sage, 2013.
- [18] “About 1177 vrdguiden healthcare guide 1177,” 1177 Vrdguiden, 2021, www.1177.se/en/Vastra-Gotaland/other-languages/other-languages/About-1177.se/about-1177-vardguiden/. Last accessed on May 11, 2022.
- [19] “Personal identity numbers and coordination numbers,” Skatteverket, 2018, <https://skatteverket.se/servicelankar/otherlanguages/inenglish/individualsandemployees/livinginsweden/personalidentitynumberandcoordinationnumber>. Last accessed on May 9, 2022.

CHAPTER 5 HOW USERS PERCEIVE THE REPRESENTATION OF NON-BINARY GENDER IN SOFTWARE SYSTEMS: AN INTERVIEW STUDY

- [20] “Ett tredje juridiskt knö och knsneutrala personnummer,” Sveriges Riksdag, 2019, www.riksdagen.se/sv/dokument-lagar/dokument/motion/ett-tredje-juridiskt-kon-och-konsneutrala_H7023316. Last accessed on May 9, 2022.
- [21] “GenderMag,” [Gendermag.org, https://gendermag.org/index.php](https://gendermag.org/index.php). Last accessed on May 18, 2022.
- [22] S. Costanza-Chock, *Design Justice: Community-Led Practices to Build the Worlds We Need*. London, England: MIT Press, 2020.
- [23] Follow the Gender Guidelines by Scheuerman et al. [1] (genderguidelines.html#Nonbinary).
- [24] R. J. Steinberg, “Gendered instructions: Cultural lag and gender bias in the hay system of job evaluation,” *Work and Occupations*, vol. 19, no. 4, pp. 387–423, 1992.
- [25] S. Tang, X. Zhang, J. Cryan, M. J. Metzger, H. Zheng, and B. Y. Zhao, “Gender bias in the job market: A longitudinal analysis,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. CSCW, pp. 1–19, 2017.
- [26] M. J. G. Stotsky, *Gender Bias in Tax Systems*. International Monetary Fund, 1996.
- [27] “Selecting your gender marker,” [State.gov, https://travel.state.gov/content/travel/en/passports/need-passport/selecting-your-gender-marker.html](https://travel.state.gov/content/travel/en/passports/need-passport/selecting-your-gender-marker.html). Last accessed on May 19, 2022.
- [28] S. Bundesverfassungsgericht, “Bundesverfassungsgericht – decisions – civil status law must allow a third gender option,” Bundesverfassungsgericht, www.bundesverfassungsgericht.de/e/rs20171010_1bvr201916en.html, October 2017. Last accessed on May 19, 2022.
- [29] “Intersex persons have the right to adequate designation in the civil register – der ö sterreichische verfassungsgerichtshof,” Vfgh.gv.at, www.vfgh.gv.at/medien/Civil_register_-_Intersex_persons.en.php. Last accessed on May 19, 2022.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 6

Elicitation Revisited for More Inclusive Requirements Engineering

James Tizard, The University of Auckland, New Zealand.*

Tim Rietz, Respeak Germany.

Kelly Blincoe, The University of Auckland, New Zealand.

To create inclusive software, development teams need to consider how they identify inclusive requirements for a software product. Requirements elicitation is the first stage in the process of developing the requirements of a software system. Elicitation is about describing the functionality, reliability, efficiency, and usability of the system to be developed, so that it suits the end users' needs [12].

Recent research has found that both traditional elicitation techniques (e.g., user interviews) and newer online crowd-based approaches may have challenges in gathering the views of a diverse set of users. In particular, there are significant challenges in eliciting requirements from users with cognitive disabilities, as well as ensuring that the full demographic spectrum (e.g., by gender, age, ethnicity) of a user base is adequately represented.

This chapter discusses the motivations for more inclusive requirements elicitation and the challenges that need to be overcome and finally makes recommendations for both requirements engineering practitioners and researchers.

Motivation for Inclusive Elicitation

Understanding user needs and desires for a software product is a critical part of modern software development. In the modern software landscape, development teams must keep their users happy to remain competitive as, in many cases, the competition is just one click away. Elicitation of user needs is central both in the initial design of the software and in its ongoing maintenance and evolution. A 2021 survey of software developers found the vast majority of developers (97%) agreed that user feedback gives them a better understanding of user needs and makes them aware of usability issues [25]. Thus, user needs as described in feedback are often being used to drive product development decisions.

However, the user base of software products can be extremely diverse, in terms of demographics (e.g., age, gender, geography, cultural background), as well as specialized needs related to physical and cognitive disabilities [4, 12, 22, 23]. If the diversity of the users being engaged through elicitation processes is not representative of the actual user base, this introduces the possibility of developing biased software that does not meet the needs of all users. A clear example of this comes from the broader field of engineering in the design of car safety devices. Women today are still significantly more likely to be seriously injured or killed in car accidents because car safety devices were designed and tested considering the size of the average man's body [7]. There are also many examples of software systems failing to consider the needs of all users. When YouTube first launched its mobile app, approximately 10% of videos were being uploaded upside-down because the software did not accommodate left-handed users.¹ More recently, various AI systems have been shown to be biased against some users. For example, Amazon's recruiting tool was found to be biased against women,² and Twitter's image cropping tool had built-in racial biases.³

Focusing on the needs of under-served people can make products better for everyone. Again, looking at an example in the broader field of engineering, curb cuts, which were originally designed to make city streets more accessible for wheelchair users, have improved the accessibility for many others, including people riding bikes or skateboards, pushing strollers, delivering packages, and pulling suitcases [3]. For a

¹www.cio.com/article/234087/consciously-overcoming-unconscious-bias.html

²www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G

³www.cbsnews.com/news/twitter-kills-its-automatic-cropping-feature-after-complaints/

software-specific example, consider closed captioning of videos, which was originally designed to make videos accessible for people with hearing impairments [13]. Today, with the advent of social media, we see captions benefiting nearly everyone since they provide viewing flexibility; people can scroll through their social media feed and watch videos without the volume or consume videos in different languages.⁴

Therefore, it is imperative that software requirements elicitation considers the needs of all of its users, to ensure the software is inclusive and fair for everyone. However, recent research has shown that both traditional requirements elicitation methods and recent crowd-based requirements elicitation approaches have representation challenges. These challenges are discussed in the next Chapter. (See Chapter 7, “Developers’ Perspective of Diverse End User Requirements,” for additional diversity-related challenges faced throughout the software development lifecycle.)

Challenges in Traditional Elicitation

Traditionally, software requirements have been elicited through methods such as interviews, focus groups, observations, and questionnaires. However, these approaches may miss segments of society. They need special focus to include diverse perspectives, especially since many of these traditional methods can only engage with a limited number of users, due to time and resource constraints. For example, you can’t interview every user of your software product and must instead engage with an extremely small sample, relative to the total user base. Other techniques, such as design thinking as described in Chapter 10, “Beyond Diversity: Computing for Inclusive Software,” can provide rich insights into the needs of users, but also face similar scalability problems.

Traditional techniques can also have a lot of inherent bias, in both the selection of elicitation participants and selective perception during elicitation. The requirements engineers’ own perceptions can cloud how they understand requirements. This can lead to miscommunications and a lack of shared understanding, which may produce misunderstood or simply missed requirements.

Inclusivity in traditional requirements elicitation requires intensive communication between all participating stakeholders, especially when engaging users with cognitive disabilities [12]. In a recent study, researchers recommended an approach based on user-centered design (UCD) [16], to engage with users with cognitive disabilities [12].

⁴www.3playmedia.com/blog/benefits-captioned-social-media-videos/

They found requirements development with those with cognitive disabilities was feasible, with the participants proving to be reliable interview partners, who were quite capable of expressing their needs for a software product. Through this process the development team gained a deeper insight into the requirements of their end users, which led to new interaction and information presentation concepts.

Collecting requirements from a diverse set of users may require a diverse set of traditional elicitation techniques, as not everyone would be comfortable, or able, to participate using the same methods. Therefore, an inclusive approach to traditional elicitation can be time consuming and expensive. For it to be done well, a development team needs a lot of motivation and drive to focus on inclusion. This can be a challenge in many software projects, where time-to-market, or other business factors, may also be an important consideration. These challenges were emphasized in a recent study that found software companies often do not prioritize accessibility needs in practice [17]. They cited various reasons for this, including that there are no methods or tools available to help the teams with this process and a general lack of training on how to consider accessibility needs.

Newer crowd-based elicitation approaches can give developers access to large volumes of diverse user perspectives, through mining online channels such as app stores, social media, and support forums. However, recent research has highlighted representation challenges here also, which are discussed in the next section.

Challenges in Crowd-Based Elicitation

Online crowd-based elicitation is a modern approach that has promise for eliciting requirements from a diverse set of users. There are large volumes of user feedback on online channels, such as app stores, social media (e.g., Twitter), and user support forums. Recent research has identified significant amounts of requirements relevant information in each of these channels, including bug reports and feature requests [22]. Through mining user opinions online, requirements engineers are no longer limited by time and other resource limitations that constrain the number of users who can be involved in more traditional elicitation techniques, such as interviews or focus groups.

Recent research has found a diverse set of users give feedback on these channels, with respect to traditional demographic categories (e.g., age, gender), geographic location, and accessibility needs [10, 11, 22, 23]. However, this research also suggests the representation across these groups in online feedback may not be in proportion to

the actual user bases of software products. Without considered attention, requirements generated from online feedback will disproportionately represent the loudest voices online and miss groups that are underrepresented.

This problem was emphasized in Tizard et al.'s 2020 survey of software users, where women reported to give significantly less online feedback than men, across all the studied channels (app stores, forums, social media) [23]. This was in line with Guzman et al.'s earlier gender study of feedback on the Apple App Store [11]. With age, the 2020 survey found that software users between 35 and 44 years reported to give the most feedback on all channels, with younger and older respondents reporting to give significantly less feedback. Research has also found that feedback behavior varies significantly between different countries and may be impacted by cultural factors [6, 10, 22].

Due to the large volume of online feedback, it is often necessary to prioritize user requests for development attention. One popular approach to prioritization is to find requests that are made frequently [5, 9, 14, 15]. However, this may exacerbate the issue of considering the views of underrepresented groups. An additional challenge is that online feedback often doesn't contain much demographic information about feedback givers, meaning directly identifying requests from underrepresented groups is difficult or perhaps impossible [23].

Those with physical or cognitive disabilities may also be missed in requirements generated from online feedback. A recent study of user reviews on the Google Play Store identified requests related to accessibility needs, including vision, hearing, and cognitive impairment [4]. However, all the accessibility requests combined made up just 1.2% of the sampled app reviews. Therefore, these accessibility requests would certainly be missed by prioritization techniques based on frequency.

While mining user opinions online is a promising approach to source valuable requirements information, there remain challenges in ensuring the generated requirements are representative of the underlying user base of a software product. In the final section, we make recommendations for practitioners on how to elicit the most representative user views for software products, with the goal of producing products that meet the needs of the broadest possible set of users. We also make recommendations for researchers, identifying several promising paths forward to better understand representation issues in requirements elicitation and develop new approaches to address these challenges.

Recommendations for Inclusive Requirements Elicitation

Recommendations for Practitioners

Where users are being directly engaged through more traditional elicitation techniques (e.g., interviews), requirements engineers must take initiative to understand the diversity within their user base and engage with them. Collecting requirements from a diverse set of users may require a diverse set of approaches, as not everyone will be comfortable, or able, to participate using the same methods [12]. In the case of users with cognitive disabilities, Heumader et al. recommend an approach based on user-centered design, finding that their process produced meaningful insights into the user needs.

Another possibility is for software teams to utilize the method described in Chapter 27, “How to Measure Diversity Actionably in Technology,” to measure diversity gaps in their requirements elicitation process using a GenderMag survey. By employing this survey, teams could gain insights into the cognitive styles of those participating in the requirements elicitation process, allowing them to identify who is missing from the process from a cognitive style perspective. As described in the same chapter also, cognitive styles can give insight into how users interact with software systems. Therefore, this survey can help teams identify whose voices are missing.

Crowd-based elicitation, where user opinions are mined from online feedback channels, can overcome many of the time and resource constraints associated with traditional elicitation approaches. Online user feedback has been found to contain much requirements relevant information, from a diverse set of users [19, 23]. Analysis tools are available to help automatically extract relevant information from the large volumes of feedback, which have shown promising performance in research settings [8, 20, 24].

As discussed, there are still challenges in ensuring user views mined online are representative. In their 2020 user study, Tizard et al. recommend that to elicit the most representative requirements information, development teams should consider feedback from multiple feedback channels. Their study found that different demographics are more likely to engage with different feedback channels. For example, younger software users reported to be more likely to engage with app stores, whereas older users may prefer support forums. They also found that a majority of feedback givers reported only engaging with one online feedback channel; therefore, focusing on a single channel will certainly miss some users.

Furthermore, the lack of demographic information, such as age and gender, continues to be a practical problem for mining the views of underrepresented groups in online feedback. Being aware that women and certain age groups may be underrepresented gives requirements engineers the option to directly engage with those groups to supplement online feedback mining. Traditional elicitation techniques such as interviews or questionnaires will be effective tools to target underrepresented demographics.

Mining user opinions from different geographic locations is more achievable in the current online landscape, as country-level location data is often available. For example, the Apple App Store divides itself by country, and location data is often available for feedback givers on social media (e.g., Twitter). Requirements engineers can therefore sample user opinions to closely match a geographically diverse user base. In doing so, the diverse views of users from different backgrounds and cultures can be uncovered and help broaden the appeal of a software product.

Finally, there are smart analysis tools available to help extract accessibility requests from app store reviews. As previously mentioned, recent research found accessibility requests in app reviews related to vision, hearing, and cognitive impairment, among others [4]. These reviews were identified with keyword searches, followed by manual analysis. Subsequent research then applied the identified reviews to build smart analysis tools to automatically extract accessibility requests with promising accuracy, which have been made available [1].

Recommendations for Researchers

For traditional elicitation techniques, there are several promising avenues of research to improve inclusivity. Heumader et al.'s work [12] points to a path forward in eliciting requirements from those with cognitive disabilities. They suggest the investigation of approaches that combine two existing design methods – inclusive participatory action research (IPAR) [18] and user-centered design (UCD) [16] – showing promising early results. Similarly, Chapter 10, “Beyond Diversity: Computing for Inclusive Software,” describes success using design thinking techniques to elicit requirements from diverse users. However, such techniques are time intensive and difficult to scale to a large number of users. Another direction for research is to address the challenges of scale facing traditional elicitation techniques, such as interviews and focus groups. For example, automated conversational agents, such as LadderBot [21], hold promise

for overcoming the time and location constraints of person-to-person elicitation. A conversational agent could enable end users to articulate needs and requirements, by mimicking a human (expert) interviewer. By automating the interview process, a significantly larger sample of a user base could be engaged. Combined with the ability to target potentially underrepresented groups, automated user interviews hold significant potential to support inclusive requirements elicitation. Future work can evaluate the effectiveness of new conversational agents (e.g., LadderBot) against traditional person-person interviews and digital questionnaires. Additionally, these chatbot approaches would be well suited to evaluation in lab-based experiments.

Crowd-sourcing software requirements has been a significant focus for requirements engineering researchers in recent years. Traditionally, the crowd has been conceptualized from a high level, taking an aggregated view of their needs. However, as discussed, a growing number of studies suggest feedback habits and attitudes vary significantly between user groups (*e.g., with gender, age, country*). In the interest of more representative requirements engineering, researchers should continue to follow current trends and investigate a more fine-grained view of the crowd. With this goal in mind, we see three key areas for research: (1) Continue to investigate the representativeness of online feedback, and so identify areas where there are representation issues. (2) Investigate the causes of representation issues, such as feedback channel design and the impact of culture. (3) Investigate new approaches to encourage more representative feedback. These research directions are discussed in the following paragraphs.

Researchers should continue to investigate the representativeness of online feedback. Perhaps the primary challenge in understanding who gives online feedback is that feedback channels give very little information about their users. On some feedback channels, such as the Google Play Store, even the full name of the person providing the feedback is often unavailable. Recent research has made progress through indirect analysis techniques, such as user surveys, inferring gender through usernames, and comparing the content of feedback across regions in the Apple App Store [6, 10, 11]. Looking forward, these research approaches can continue to be leveraged; in particular, directly engaging software users (*e.g., user surveys*) continues to hold promise for gaining meaningful insights.

One avenue open for new research is the study of additional feedback channels, beyond the existing studies. The gender and regional analysis studies from Guzman and Fisher [6, 10, 11] focused on the Apple App Store, while Tizard et al.'s user survey

studies [22, 23] focused on app stores, social media, and product forums. Extending these studies to additional feedback channels (e.g., issue trackers) would likely provide additional insights into online feedback behavior.

Future work should also endeavor to understand additional demographic and minority groups within the crowd and could also be extended to include intersectionality between groups [23]. For example, little is known about the ethnicity of feedback givers or differences across the economic spectrum. With gender, current work has been limited to only the differences between participants who identified as men and women. This can be extended to understanding the feedback behavior of non-binary software users. There is also significant room to continue to investigate differences in feedback behavior between countries [6, 10, 22].

The second main research direction we see is to investigate the causes of representation issues in online feedback. Previous work found underrepresented groups were more likely to cite several key reasons not to give online feedback. For example, both women and those under 25 years old more frequently (than their counterparts) reported that they found app stores confusing or hard to use, felt a resolution to their problem would take too long, and were not aware feedback could influence software improvements. Research has found that most software has gender inclusivity issues [2], so it's possible that similar inclusivity issues exist in the software that collects online feedback.

A recent study also found software users in China and Germany reported significantly diverging reasons not to give feedback and suggested underlying cultural factors, such as collectivism and power distance [22]. Similar to other underrepresented groups, Chinese respondents were more likely (than Germans) to find online channels confusing or hard to use and were less likely to be aware they could influence software improvements through their feedback. Future research should investigate why certain groups are disproportionately impacted by these factors. There is also significant room to investigate differences in the motivations to give feedback between counties and the possible impact of culture.

Finally, researchers should investigate new approaches to encourage more representative online feedback. One promising direction for investigation is to directly address the factors underrepresented groups identify for not giving feedback, as discussed previously. Methods proposed by software users in previous work hold promise for addressing these challenges and should be investigated [23]. For example, giving a quick response to online feedback could be used to emphasize the connection

to software improvement and help address the perception that a resolution will take too long. Clearly showing a track record of addressing feedback could also promote awareness of the process and help motivate user input. Future work should also investigate feedback interfaces that underrepresented groups find encouraging and easy to use. Lab trials could be carried out to evaluate if the approaches identified previously encourage feedback in a practical context.

Summary

Understanding and addressing user needs through diligent requirements elicitation is critical to success in the modern software landscape. In this chapter, we described the challenges in gathering views from a diverse set of users. Traditional elicitation methods (e.g., interviews) can exclude a significant proportion of the user base due to practical constraints, such as limited time. They can also suffer from bias and misunderstandings. For crowd-based elicitation, certain demographic groups can be significantly underrepresented in the online feedback it leverages. This issue is exacerbated by the lack of demographic information available about the online feedback givers, meaning it's difficult (or impossible) to target feedback from specific groups. We made several recommendations to help practitioners overcome these challenges, including using various elicitation techniques to accommodate diverse users, employing user-centered design practices, and various strategies to increase the diversity of those participating in the requirements elicitation process. Finally, we outlined several promising directions for requirements engineering researchers to advance the literature on inclusive requirements elicitation.

Bibliography

- [1] Eman Abdullah AlOmar, Wajdi Aljedaani, Murtaza Tamjeed, Mohamed Wiem Mkaouer, and Yasmine N. El-Glaly. Finding the needle in a haystack: On the automatic identification of accessibility user reviews. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15, 2021.
- [2] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephan Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.

- [3] Elizabeth F. Churchill. Putting accessibility first. *Interactions*, 25(5):24–25, 2018.
- [4] Marcelo Medeiros Eler, Leandro Orlandin, and Alberto Dumont Alves Oliveira. Do Android app users care about accessibility? An analysis of user reviews on the Google Play Store. In *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, 1–11, 2019.
- [5] Layan Etaiwi, Sylvie Hamel, Yann-Gaël Guéhéneuc, William Flageol, and Rodrigo Morales. Order in chaos: prioritizing mobile app reviews using consensus algorithms. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMP-SAC)*, 912–920, IEEE, 2020.
- [6] Ricarda Anna-Lena Fischer, Rita Walczuch, and Emitza Guzman. Does culture matter? Impact of individualism and uncertainty avoidance on app reviews. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 67–76, IEEE, 2021.
- [7] Jason Forman, Gerald S. Poplin, C. Greg Shaw, Timothy L. McMurry, Kristin Schmidt, Joseph Ash, and Cecilia Sunnevang. Automobile injury trends in the contemporary fleet: Belted occupants in frontal collisions. *Traffic Injury Prevention*, 20(6):607–612, 2019.
- [8] E. Guzman, R. Alkadhi, and N. Seyff. A needle in a haystack: What do twitter users say about software? In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 96–105, 2016.
- [9] Emitza Guzman, Mohamed Ibrahim, and Martin Glinz. Prioritizing user feedback from Twitter: A survey report. In *2017 IEEE/ACM 4th International Workshop on Crowd Sourcing in Software Engineering (CSI-SE)*, 21–24, IEEE, 2017.
- [10] Emitza Guzman, Lúis Oliveira, Yves Steiner, Laura C. Wagner, and Martin Glinz. User feedback in the app store: a cross-cultural study. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 13–22, IEEE, 2018.

- [11] Emitza Guzman and Andres Paredes Rojas. Gender and user feedback: An exploratory study. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 381–385, IEEE, 2019.
- [12] Peter Heumader, Cordula Edler, Klaus Miesenberger, and Sylvia Wolkerstorfer. Requirements engineering for people with cognitive disabilities – exploring new ways for peer-researchers and developers to cooperate. In *International Conference on Computers Helping People with Special Needs*, 439–445, Springer, 2018.
- [13] Richang Hong, Meng Wang, Mengdi Xu, Shuicheng Yan, and Tat-Seng Chua. Dynamic captioning: video accessibility enhancement for hearing impairment. In *Proceedings of the 18th ACM International Conference on Multimedia*, 421–430, 2010.
- [14] Swetha Keertipati, Bastin Tony Roy Savarimuthu, and Sherlock A. Licorish. Approaches for prioritizing feature improvements extracted from app reviews. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 1–6, 2016.
- [15] Fitsum Meshesha Kifetew, Anna Perini, Angelo Susi, Aberto Siena, Denisse Muñante, and Itzel Morales-Ramirez. Automating user-feedback driven requirements prioritization. *Information and Software Technology*, 138:106635, 2021.
- [16] Travis Lowdermilk. *User-Centered Design: A Developer's Guide to Building User-Friendly Applications*. O'Reilly Media, Inc., 2013.
- [17] Darliane Miranda and João Araujo. Studying industry practices of accessibility requirements in agile development. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 1309–1317, 2022.
- [18] Janice Ollerton. IPAR, an inclusive disability research methodology with accessible analytical tools. *International Practice Development Journal*, 2(2), 2012.

- [19] D. Pagano and W. Maalej. User feedback in the app store: An empirical study. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, 125–134, 2013.
- [20] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. ARdoc: App reviews development oriented classifier. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016*, 1023–1027, ACM, New York, NY, USA, 2016.
- [21] Tim Rietz and Alexander Maedche. LadderBot: A requirements self-elicitation system. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 357–362, IEEE, 2019.
- [22] James Tizard, Tim Rietz, Xuanhui Liu, and Kelly Blincoe. Voice of the users: A study of software feedback differences between Germany and China. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 328–335, IEEE, 2021.
- [23] James Tizard, Tim Rietz, Xuanhui Liu, and Kelly Blincoe. Voice of the users: an extended study of software feedback engagement. *Requirements Engineering*, 27(3):293–315, 2022.
- [24] James Tizard, Hechen Wang, Lydia Yohannes, and Kelly Blincoe. Can a conversation paint a picture? Mining requirements in software forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 17–27, IEEE, 2019.
- [25] Simon van Oordt and Emitza Guzman. On the role of user feedback in software evolution: a practitioners perspective. In *2021 IEEE 29th International Requirements Engineering Conference (RE)*, 221–232, 2021.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 7

Developers' Perspective of Diverse End User Requirements

John Grundy, Monash University, Australia.

Tanjila Kanij, Swinburne University of Technology, Australia.*

Jennifer McIntosh, Monash University, Australia.

Hourieh Khalijah, Monash University, Australia.

Ingo Mueller, Monash University, Australia.

Software is designed and developed primarily to serve human needs. However, many software systems continue to fail to take into account diverse end users' characteristics, causing frustration, errors, and even potentially life-threatening situations. These end user human-centric aspects include, but are not limited to, age, ethnicity, gender, personality, cognitive style, language, culture, physical and mental challenges, emotional reactions, socio-economic status, etc. Software applications need to cover many if not all of these end user human-centric aspects in order to provide a suitable interface, workflow, and solution for diverse end users.

There may be a number of reasons software engineers do not sufficiently take their end user human-centric aspects into account. This includes poor understanding of user needs, inappropriate designs, and time pressures [3, 9, 12, 28]. Some larger organizations have dedicated UX/UI and/or customer experience teams that separate developers from end users [5]. Many companies are very small, and developers need to do all such work themselves, but lack sufficient training in UX, participatory design, or other human-centric design methods [16, 19, 22]. Software developers are generally

well-educated, relatively young, mostly male, most well-conversant in English, of high socio-economic status, and very comfortable with technology. Because of this, they may find it difficult to empathize, understand, and subsequently incorporate diverse human-centric aspects during the software engineering (SE) process [9, 11, 15, 24].

As part of our larger research effort to improve support for diverse end user human-centric aspects during software development, we wanted to better understand how developers currently go about addressing these challenging human-centric aspects of their end users in contemporary software development projects. We wanted to find out which are the key end user human-centric aspects that software developers currently find challenging to address and how they currently go about trying to address diverse end user human-centric aspects. We wanted to find out what sorts of end user human-centric aspects they tend to encounter, which ones they view as more important and which more challenging to address, what techniques (if any) they currently use to address (some of) them, and where they perceive further research in this area could be done to provide them practical support. To this end we carried out a detailed online survey of developers and development team managers, receiving 60 usable responses. We interviewed 12 developers and managers from a range of different practice domains, role specializations, and experience levels to explore further details about issues.

Human Aspects of Users

In the following are some of the end user human-centric aspects that software teams need to consider:

Gender: Several prominent mainstream articles and books have highlighted gender bias in various technologies, including apps and smart living technologies [20, 25]. Recent work has investigated how software and other systems are gender biased in various ways [1].

Age: Many smart living systems focus on supporting aging people. Many educational software systems are targeted to children [10, 17]. Different ages may have different expectations, challenges, and reactions to the same software that need to be addressed [14, 26].

Ethnicity and culture: Software that fails to take into account or is biased in terms of ethnicity of people is highly problematic, especially for many emerging smart city applications, for example, policing and surveillance [8].

Physical/mental challenges: Many people live with mental health challenges, cognitive impairment, and a wide variety of physical challenges, for example, impaired mobility, sight, hearing, and speech [24, 29]. Many software solutions have been developed to assist with these challenges or to take account of them to increase accessibility to software [2, 23].

Language: Different users speak different languages, have different educational attainment levels, have specific colloquialisms and jargon, and have different language competencies. Considering these aspects is particularly important during dialogue design, including multilingual software and software that adapts to different user dialogue preferences [21].

Human values: Values, for example, inclusiveness, equality, privacy, openness, etc., reflect how, why, and to what degree humans value people, objects, and ideas [27]. Many apps conflict with one or more human values, causing expectation mismatches and reducing app usage, take-up, and acceptance [18].

Emotions: Different people react differently to technology solutions emotionally. This includes positive reactions, for example, to a smart home solution providing a feeling of safety, and negative reactions to the same software, for example, feeling lack of control or being monitored intrusively [4].

Engagement and entertainment: Some people are highly driven by enjoyment, entertainment, and “fun” aspects of using software – computer games and gamification. Developers need to be aware of how to best design such solutions to achieve high levels of engagement and enjoyment [7, 13].

Study Design

We formulated the following research questions to guide our study:

RQ1: What are the range and nature of end user human-centric aspects that have to be addressed by software developers?

RQ2: How are different human-centric aspects addressed at different phases of software development?

RQ3: What current support is available to developers and what improvement is needed?

Survey and Interviews

We designed an online survey targeted at a broad range of software developers to provide us with a big picture view of current practices, challenges, and approaches to address key end user human-centric aspects. The survey was composed of three sections: demographic questions, participant views on end user human-centric aspects, and particular techniques – guidelines, practices, and tools to address diverse end user human aspects. To complement this online survey, we developed an interview protocol allowing us to drill down to more detailed information in one-on-one interviews. The research is approved by the Monash University Human Research Ethics Subcommittee.

Recruitment and Data Collection

We ran our developer survey from July 2020 to March 2021. We recruited participants from personal networks, by advertising on LinkedIn and Twitter, and through snowballing. We were particularly interested in surveying those developing software applications where (some of) their end users have particular “challenges,” for example, physical and mental, those developing with consideration for some of the end user aspects, such as age (very young or aging), language proficiency, low socio-economic status, low access to technology, and/or technology skills. We recruited for the interviews from our own professional software developer networks but also asked survey respondents to volunteer to be interviewed. We then selected a representative range of interviewees (domain of work, role, experience, etc.). Originally we planned to conduct face-to-face interviews, but due to COVID-19 restrictions, all were done via Zoom, allowing us to interview participants from other countries and time zones.

Data Analysis

Analysis of the quantitative data is mainly descriptive and explores common and uncommon aspects and key associations. Qualitative analysis included content analysis and thematic analysis [6]. We identified key themes via open coding and grouped common themes and responses. We used closed coding for further analysis and found key themes.

Results

Participants

We had over 130 online survey responses, but only 60 were usable; the rest were removed from the final analysis due to incompleteness and/or poor quality of responses. Poor quality was decided where irrelevant or “throw-away” responses were encountered. Forty-four of these participants were male and twelve female; four did not state their gender. Ages of the participants were 21–30 (23), 31–40 (18), 41–50 (8), 51–60 (6), 61+ (1), and under 20 (1); three did not state their age. Years of experience ranged from 1 to 5 years (22), 6 to 10 years (12), 11 to 15 years (9), 16 to 20 years (6), 21 to 25 years (3), 26 to 30 years (2), 31 to 35 years (1), and 36 to 40 years (1); three did not state their years of experience. Most developers came from a computer science or SE training background – 22 (CompSci), 9 (SoftEng), 9 (IT/InfoSys), and 5 (computer engineering) – and others (one each) were from robotics, physics, forensic computing, AI/ML/Vision, and neuroscience; the rest did not state their background. The current roles of the respondents were programmer (35), software architect (13), user interface designer (10), tester (9), project manager, requirements engineer, operations (6 each), and others (15). The domains they worked in included finance (24), education (23), health (19), transport and logistics (15), government services (13), and social media and insurance (7 each). Nineteen respondents reported other domains.

We also interviewed 12 respondents – 11 were male and 1 female (9 from Australia, 2 from New Zealand, and 1 from the Middle East). Ages ranged from 21 to 30 (1), 31 to 40 (4), 41 to 50 (5), and 51 to 60 (2). Years of experience were 1–5 years (1), 6–10 years (3), 11–15 years (1), 16–20 years (4), and 30+ years (3). The interviewees covered a broad range of roles including project managers (4), requirements engineers (1), software architects (2), user interface designers (2), programmers (2), testers (1), and others (5); many people performed more than one role.

Answers to Our Research Questions

RQ1: What Are the Range and Nature of End User Human-Centric Aspects That Have to Be Addressed by Software Developers?

We asked survey participants to tell us what end user human-centric aspects they have had to address in their software projects, summarized in Figure 7-1. Aging users, users with accessibility needs, those with physical challenges, those with language proficiency issues and uncomfortable with technology, and those with diverse cultural backgrounds were areas more highly reported. Only one participant reported never addressing any of the issues. Interviewees described specific human-centric aspects they had to address and how they managed these challenges. Most common issues included aging users, users who were technologically challenged, those who were from diverse cultural backgrounds and/or spoke languages other than English, specialized groups with unique work contexts, and even personality types. Figures 7-2 and 7-3 demonstrate the human aspects our respondents consider as critical in their work and according to different SE phases, respectively.

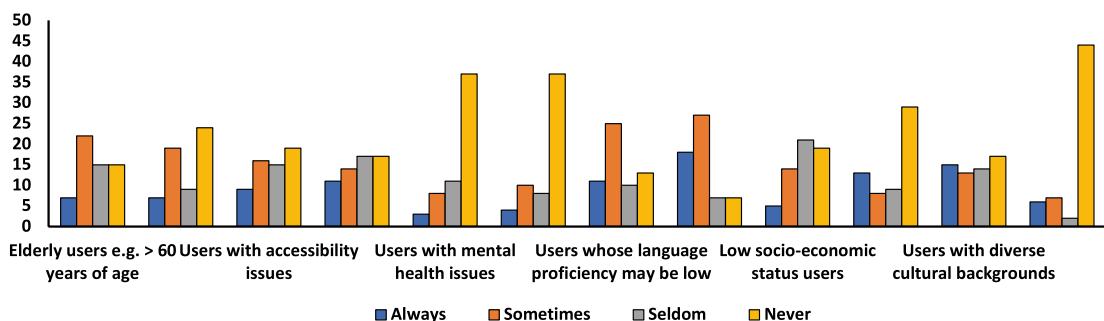


Figure 7-1. End user human-centric aspects survey respondents need to address

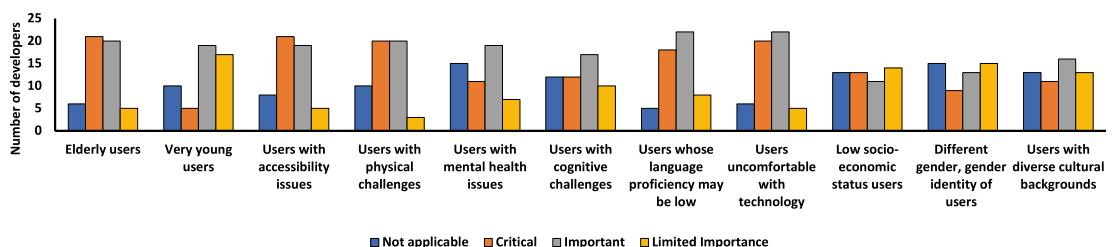


Figure 7-2. End user human-centric aspects judged to be critical (or not) in their work (survey)

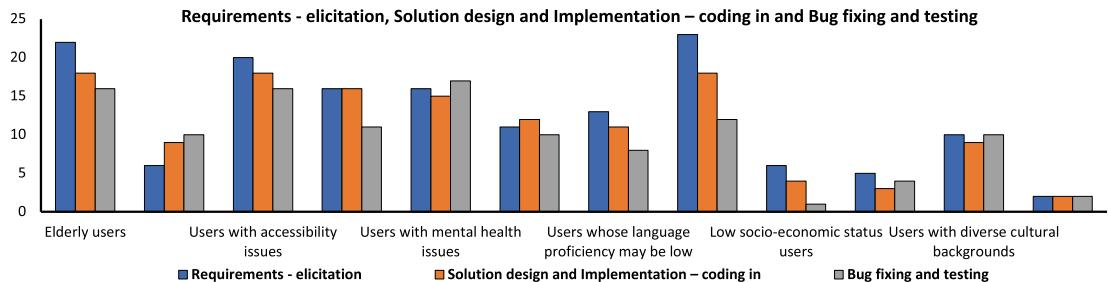


Figure 7-3. End user human-centric aspects judged to be critical in SE phases (survey)

Technical proficiency: For example, developers had to adapt software for users with low technological capability: “*There’s a lot of [users] that struggle with digital technology, even to the point where we’re actually building a web application that was previously just a mobile application just so that it’s accessible to everyone.*”

Age: Some addressed the issue of developing for users of different age groups: “*Many ticket officers/operators are middle-aged or even more senior. They’re usually busy, less willing to explore the functionality of our software, and have to multitask.*” and “*In our health systems, we have a large group of users [who] are ‘elderly.’ They include both clinical service providers (e.g., doctors, nurses and service staff, etc.) and elderly patients.*”

Culture: Cultural differences were observed between the developers and the users: “*You put the robot into the wild; you discover things you didn’t foresee. We were very conservative, to try not to offend anyone. But you still discover things, like I did not expect this question: Do you believe in God? People were very insistent on getting answers on this topic in this area. I think it is very important to have people who can think in this context in an early phase.*”

Some developers suggested that some domains tend to come with more end user human-centric aspects, for example, health, financial, community, social media, and safety-critical systems.

Clinical software: In a clinical setting, to make sure the software was used according to the clinicians’ needs, they used different panels for different parts of input for the clinician to click the panel that was being discussed, and if something else was suddenly being discussed, they could just switch to a different panel without skipping forward and backward to a few screens to get to the right spot. However, they “*did not really realize this up until they tried it in the clinics and the first version was trialed.*” The developers realized that “*it didn’t match up to the way people having conversations with the [doctor].*” Another issue they did not realize until it was tested was that they should not present the

details related to the user's cancer prediction in a same way to all the patients. *"Because if a person has a risk calculator and their risk comes out to be in a high-risk category, if you just program the tool to present that risk to the person like anyone else, that can really be stressful for the patient and it can induce anxiety, which is all the things we wanted to avoid in the goals of this project."*

Games: It is essential to address user emotions, engagement, age, and language.

Others: In financial, community, and social media application domains, there are very wide range of end users with different expectations and needs. Some developers flagged the issue of end users with multiple, interacting human-centric aspects that are very challenging to address: *"It ties in with physical challenges (e.g., screen readability and its impact on deteriorating eyesight) and being comfortable using technology, which many older people are not."* Some developers also noted they had little control over how many end user human-centric aspects were addressed or even if they were addressed. Several organizations had dedicated UX teams and larger ones "customer experience" teams – a development manager noted how their team didn't have direct access to end users and communication of end user needs or difficulties came through mixed channels.

There are a range of end user human-centric aspects that developers acknowledge need to be better considered by developers, but there are also particular domain-specific challenges for some application domains. We want to investigate these domain-specific challenges in more depth in future work.

RQ2: How Are Different Human-Centric Aspects Addressed at Different Phases of Software Development?

We asked developers about the relative difficulty of addressing these human-centric aspects during one or more phases on a scale from 0 (no challenge) to 100 (most challenging). The survey and interviews reflected similar findings; however, interview participants emphasized that human-centric aspects need to be better considered at all stages of development.

Requirements engineering: Some of the key challenges stated included (1) gathering requirements for these end users (reported for elderly, children, mentally and physically challenged end users), (2) addressing the issues this end user group

has (accessibility, physically challenged, cultural differences), (3) lack of sufficient knowledge on how to address these issues (reported for several of these human-centric aspects), (4) ethical issues in gathering these requirements (children), (5) finding and communicating with suitable end users with these challenges (reported for many of these human-centric aspects), (6) a very wide range of issues for end users with this human-centric aspect (accessibility needs and aging users), (7) satisfying all end users with these human-centric aspects (accessibility, culture, language), (8) the issue being very complex (cultural differences), and (9) meeting these requirements with suitable designs (reported for several human-centric aspects). While some developers believed the requirements phase should be the most important, many acknowledged that it was not unusual either for users to expect too much and developers had to manage their expectations or for users to change their mind, making it important to have checks in place throughout the development cycle. One developer put it succinctly saying that even once a prototype was developed, it was still important to evaluate human-centric aspects as "*sometimes when we are starting to develop, they are only concepts. We don't even know what kind of implications the technology may have.*"

Design and development: Key difficulty reasons reported for design- and implementation-related tasks included (1) finding balance between designs that met different needs (elderly, children, accessibility), (2) designing solutions (accessibility), (3) including end users in the design process (children), (4) finding suitable design tools (children and accessibility aspects), (5) applying existing standards (accessibility), (6) getting to know characteristics of end users and their preferences (accessibility and gender), and (7) foreseeing the possible range of end user human issues (culture, accessibility, and language aspects).

Testing and maintenance: Key reasons given for difficulty for test- and maintenance-related tasks included (1) finding representative testers (reported for aging users, those with cognitive challenges, different genders, and cultural diversity), (2) the need for extensive testing (accessibility), (3) difficulty of testing and fixing if the developers do not have the challenges themselves (many), (4) difficulty determining who the end users are (for language and socio-economic status), (5) difficulty determining specific actual usage issues (for children, accessibility, culture, language, socio-economic aspects), and (6) potential ethical issues (for recruiting testers with mental health challenges). Testing was seen as a problem if testers were not using the software in the context it was designed for, for example, under stress: "*When they're just doing stuff, they have a very different behavior than when they're stressed. You have to be*

much more clear. It's really important for that testing under that real-life sort of situation, testing under the worst case, because it's really important that the behavior under all those work conditions is taken into account."

There was some variation in how hard and long it takes developers to address different human-centric aspects in their software development. However, overall, many are challenging and developers often lack expertise, time, and support to address them all. A lot of software is currently released without testing for how well it supports diverse end user human-centric aspects.

RQ3: What Support Is Available to the Developers and What Improvement Is Needed?

We asked survey respondents to tell us what key techniques and tools, if any, they currently use or have used to address some of these end user human-centric aspects in their project work. Figure 7-4 summarizes key techniques used. Surprisingly few reported using "human-centered" RE and design approaches. Agile and iterative software development methods and usability evaluation techniques were claimed to be beneficial by several. A few reported feedback mechanisms, "best practices" such as applying standards, and including end users in the process were all critical. A few use standards/guidelines for specific human-centric aspects, especially usability and those with physical or cognitive challenges. Figure 7-5 illustrates tools used by participants. A few use visual design tools to model user human-centric aspects, Jira to track human-centric aspect-related defect fixing, Trello cards to capture human-centric requirements, and accessibility APIs. Interviewed developers reported using tools including Jira, Zendesk, DevOps, Nagios, and Selenium to increase communication between developers, stakeholders, and users. Similarly to the survey results, many talked about successfully using Agile and iterative software development methods, participatory scrum, brainstorming techniques, and methods for increasing rapid feedback to increase capturing and addressing human-centric aspects. One team even trained users to be scrum masters. Smaller teams and solo practitioners employed informal methods for getting feedback and resolving problems, for example, email and spreadsheets.

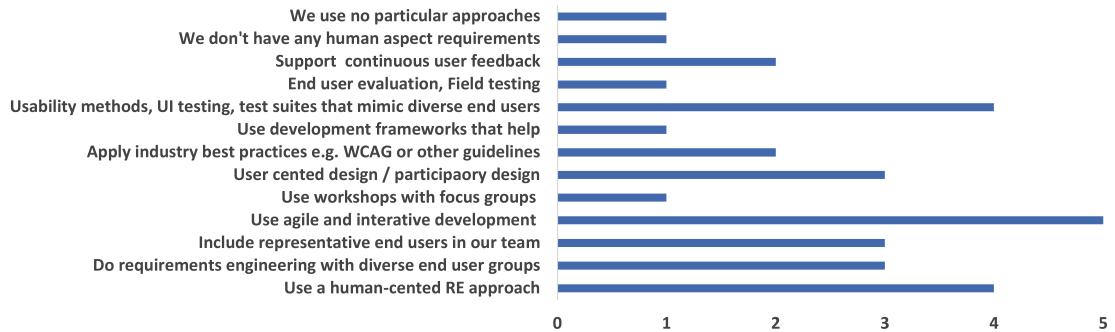


Figure 7-4. Techniques used to address end user human-centric aspects

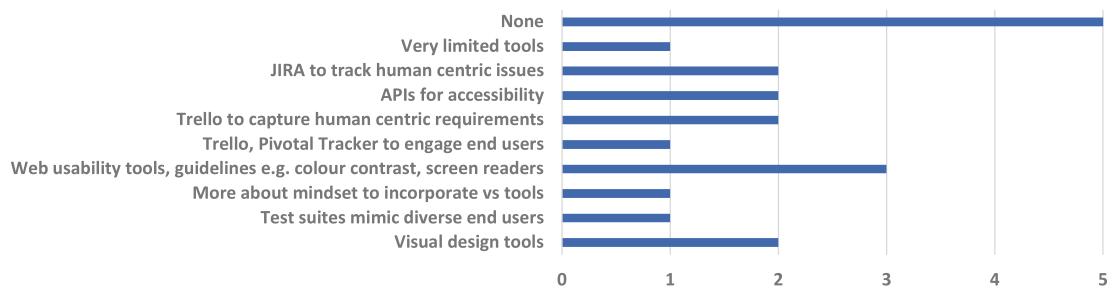


Figure 7-5. Tools to address human-centric aspects

We asked developers what improved tools and techniques they thought would help them. Key examples given included better development processes to improve target end user collaboration; better guidelines and practices to follow to address diverse end user human-centric aspects in software; better requirements capture and human-centric aspect modeling support that would enable them to identify and better track these end user needs throughout development; AI-based tools to automatically advise on missing end user human-centric aspects, for example, to prompt them to consider certain end user human-centric aspects in different situations; and more “live” or *in situ* testing with representative end users, to get richer feedback on issues that arise in software from lack of consideration of end user human-centric aspects. A number of other suggestions were given including a need for better education of software engineers about diverse end user human-centric aspects and their impact on software usage; simpler GUIs for many end user populations; better defect reporting to enable diverse end users to more easily identify, describe, and report problems they have with their software; and so on.

Participants also suggested developers try out being “users”: *“I think if you take two days out of a development cycle and send half of your developers to be the user for a couple of days, you’ll pay that. You’ll save that in tons later on that project.”*

While some development methods and tools are used to help address human-centric aspects, many opportunities exist to improve the use of existing methods and tools and create new ones.

Limitations

Ideally, we would have had a larger number of survey respondents and interviewees. The demographics of the respondents did however give us a reasonable spread of experience, domain, and gender. We purposively chose interviewees from volunteers to give us a broad range of demographics. Our survey questions may have been misinterpreted by some respondents, and some may have not taken due care with the survey. We did our best to use terminology and brief explanations in the survey that developers would correctly interpret based on a pilot.

Summary

We reported results of online survey and interviews of software engineers exploring challenges they face in addressing a range of human-centric aspects of their end users. Most software engineers share few human-centric aspect characteristics with some of their end user groups. Key challenges identified included lack of education, knowledge, experience, guidelines, and tools about ways to best address some end user human-centric aspects; difficulty in recruiting representative end users and working with them throughout development; sheer difficulty in addressing a wide range of sometimes conflicting human-centric aspects; inability to satisfy all potential end users with differing human-centric aspects; and lack of time, budget, and management support in addressing many of the aspects. We want to carry out observational studies with a small number of software teams to observe developers working on software to see how they discuss and address these issues. We also want to survey and selectively interview a range of stakeholders and end users of software applications to better understand their challenges using the software. We want these learnings to help us trial with developers and end users new SE processes, techniques, and tools to address (some of) the challenging, outstanding issues in human-centric aspects in software for end users.

Acknowledgments

All the authors are supported by ARC Laureate Fellowship FL190100035.

Bibliography

- [1] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.
- [2] Marta G. Carcedo, Soon Hau Chua, Simon Perrault, Paweł Wozniak, Raj Joshi, Mohammad Obaid, Morten Fjeld, and Shengdong Zhao. HaptiColor: Interpolating color information as haptic feedback to assist the colorblind. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 3572–3583, 2016.
- [3] Enrico Coiera, Jos Aarts, and Casimir Kulikowski. The dangerous decade. *Journal of the American Medical Informatics Association*, 19:2–5, 2011.
- [4] Maheswaree Kissoon Curumsing, Niroshinie Fernando, Mohamed Abdelrazek, Rajesh Vasa, Kon Mouzakis, and John Grundy. Emotion-oriented requirements engineering: A case study in developing a smart home system for the elderly. *Journal of Systems and Software*, 147:215–229, 2019.
- [5] Tiago Silva Da Silva, Milene Selbach Silveira, Claudia de O. Melo, and Luiz Claudio Parzianello. Understanding the UX designers' role within agile teams. In *International Conference of Design, User Experience, and Usability*, 599–609, Springer, 2013.
- [6] Martin Denscombe. *The Good Research Guide: For Small-Scale Social Research Projects*. Open University Press, Milton Keynes, 2003.

- [7] Anna Fensel, Dana Kathrin Tomic, and Andreas Koller. Contributing to appliances? Energy efficiency with internet of things, smart data and user engagement. *Future Generation Computer Systems*, 76:329–338, 2017.
- [8] Clare Garvie and Jonathan Frankle. Facial-recognition software might have a racial bias problem. *The Atlantic*, 7, 2016.
- [9] John Grundy. Human-centric software engineering for next generation cloud-and edge-based smart living applications. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 1–10, IEEE, 2020.
- [10] John Grundy, Kon Mouzakis, Rajesh Vasa, Andrew Cain, Maheswaree Curumsing, Mohamed Abdelrazek, and Niroshine Fernando. Supporting diverse challenges of ageing with digital enhanced living solutions. In *Global Telehealth Conference 2017*, 75–90, IOS Press, 2018.
- [11] Kathleen Hartzel. How self-efficacy and gender issues affect software adoption and use. *Communications of the ACM*, 46(9):167–171, 2003.
- [12] Alenka Kavcic. Software accessibility: Recommendations and guidelines. In *EUROCON 2005 – The International Conference on “Computer as a Tool,” Volume 2*, 1024–1027, IEEE, 2005.
- [13] Janaki Kumar. Gamification at work: Designing engaging business software. In *International Conference of Design, User Experience, and Usability*, 528–537, Springer, 2013.
- [14] Jennifer McIntosh, Xiaojiao Du, Zexian Wu, Giahuy Truong, Quang Ly, Richard How, Sriram Viswanathan, and Tanjila Kanij. Evaluating age bias in e-commerce. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 31–40, 2021.
- [15] Tim Miller, Sonja Pedell, Antonio A. Lopez-Lorca, Antonette Mendoza, Leon Sterling, and Alen Keirnan. Emotion-led modelling for people-oriented requirements engineering: The case study of emergency systems. *Journal of Systems and Software*, 105:54–71, 2015.

- [16] Jessica Nguyen and Marc Dupuis. Closing the feedback loop between UX design, software development, security engineering, and operations. In *Proceedings of the 20th Annual SIG Conference on Information Technology Education*, 93–98, 2019.
- [17] Marije Nouwen, Maarten Van Mechelen, and Bieke Zaman. A value sensitive design approach to parental software for young children. In *Proceedings of the 14th International Conference on Interaction Design and Children*, 363–366, 2015.
- [18] Humphrey O. Obie, Waqar Hussain, Xin Xia, John Grundy, Li Li, Burak Turhan, Jon Whittle, and Mojtaba Shahin. A first look at human values-violation in app reviews. *2021 International Conference on Software Engineering*, 2021.
- [19] Tina Øvad, Nis Bornoe, Lars Bo Larsen, and Jan Stage. Teaching software developers to perform UX tasks. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*, 397–406, 2015.
- [20] Caroline Criado Perez. *Invisible Women: Exposing Data Bias in a World Designed for Men*. Random House, 2019.
- [21] Johann Roturier. *Localizing Apps: A Practical Guide for Translators and Translation Students*. Routledge, 2015.
- [22] Kristen Shinohara, Saba Kawas, Andrew J. Ko, and Richard E. Ladner. Who teaches accessibility? A survey of US computing faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 197–202, 2018.
- [23] Javier Sánchez Sierra and JS Togores. Designing mobile apps for visually impaired and blind users. In *The Fifth international Conference on Advances in Computer-Human Interactions*, 47–52, Citeseer, 2012.
- [24] Steven E. Stock, Daniel K. Davies, Michael L. Wehmeyer, and Susan B. Palmer. Evaluation of cognitively accessible software to increase independent access to cellphone technology for people with intellectual disability. *Journal of Intellectual Disability Research*, 52(12):1155–1164, 2008.

- [25] Yolande Strengers and Jenny Kennedy. *The Smart Wife: Why Siri, Alexa, and Other Smart Home Devices Need a Feminist Reboot*. MIT Press, 2020.
- [26] Drew Williams, Mohammad Arif Ul Alam, Sheikh Iqbal Ahamed, and William Chu. Considerations in designing human-computer interfaces for elderly people. In *2013 13th International Conference on Quality Software*, 372–377, IEEE, 2013.
- [27] Emily Winter, Steve Forshaw, and Maria Angela Ferrario. Measuring human values in software engineering. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–4, 2018.
- [28] Simone Wirtz, Eva-Maria Jakobs, and Martina Ziefle. Age-specific usability issues of software interfaces. In *Proceedings of the IEA*, volume 17, 2009.
- [29] Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang. Seenomaly: Vision-based linting of GUI animation effects against design-don't guidelines. In *42nd International Conference on Software Engineering (ICSE20)*, ACM, New York, NY, 2020.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 8

UI Development Experiences of Programmers with Visual Impairments in Product Teams

Maulishree Pandey, University of Michigan, USA.*

Sharvari Bondre, University of Michigan, USA.

Vaishnav Kameswaran, University of Michigan, USA.

Hrishikesh Rao, University of Michigan, USA.

Sile O'Modhrain, University of Michigan, USA.

Steve Oney, University of Michigan, USA.

The tools and techniques that software engineers use to collaborate are critical in deciding who can contribute to software projects and the roles they can play within those teams. The consistent growth of UI developer job roles [7, 8, 9] has made many programmers seek UI engineering jobs. It is important to understand the accessibility of the profession and identify ways to make it more inclusive. We conducted two qualitative studies [10, 11] to better understand the strategies that mixed-ability teams – specifically teams where some team members identify as having a visual impairment and some do not – use to collaborate on user interface (UI) development. In this chapter,

we summarize and synthesize the findings from our prior studies to highlight the challenges programmers with visual impairments encounter in collaborative UI programming. The chapter concludes with recommendations for building more inclusive software engineering teams by fostering communication and help-seeking interactions, which we hope product teams would find valuable. We also derive implications for UI frameworks that aim to support accessible application development. These implications can inform the engineering choices of product teams as well as inform the efforts of researchers and developers building these frameworks.

Background

Much of the existing empirical research has investigated the experiences of programmers with visual impairments as individual contributors. These prior studies offer insights into the challenges of creating websites from scratch using HTML (which is primarily responsible for defining a site's content), CSS (which is intended to specify how browsers should visually render the content), and JavaScript (which is primarily responsible for specifying the interactive aspects of the site) [3, 4, 6]. Programmers with visual impairments have shared that they feel less confident about modifying CSS rules on their own [2, 5]. Layout editors within IDEs and browser inspector tools interface poorly with desktop screen readers such as NVDA and JAWS and tend to not provide pixel positions, relative locations, and dimension information. Therefore, they often seek sighted assistance to spot-check the CSS edits and verify the layout of their design [4, 6, 11]. To foster more independent UI creation, Andy Borka developed the Developer Toolkit [1], an NVDA add-on that informs developers of pixel location and dimensions of UI elements. Another approach is to represent the HTML in nonvisual form, for instance, using tactile printouts [4] or tactile beads that can be organized on sensing boards, to teach students about UI development [13]. Existing research suggests that programmers with visual impairments find UI development for mobile platforms easier than developing for desktops because they can use the screen readers' gestures (e.g., single tap to explore the UI, double tap to select, swipe to move focus, etc.) to verify the size and position of UI elements [11, 12]. Programmers with visual impairments have leveraged the combination of touchscreens and cross-platform frameworks to develop UIs with relative independence for multiple platforms [10]. However, the aforementioned research does not reveal much about their collaboration with sighted developers and designers in the context of UI development. Prior research has mainly

reported on challenges in collaboration over technical diagrams, presentations, and data visualizations [4, 5]. Given the importance of teamwork and collaboration in UI programming, this chapter describes ways software development teams can improve their work practices and tools to foster the participation of programmers with visual impairments.

Methodology

Our research was motivated by the program-l mailing list (program-l@freelists.org) – an active online discussion group for programmers with visual impairments to ask and share programming-related resources. We joined the mailing list in 2018 and regularly came across questions such as the following one, asking for accessible resources and tools to carry out UI development:

I want to learn to develop mobile apps for iOS and Android. I have both Windows 10 and MacBook Pro. I am totally blind so I'm dependent on JAWS and VoiceOver. The development stack is completely my choice. I am considering Xamarin and C#, Ionic and JavaScript, or React Native and JavaScript. My question is, what development stack are blind programmers having success in developing mobile apps?

Posts such as these revealed that programmers with visual impairments had to identify accessible frameworks and development tools before they could dive into UI programming. Our first study broadly focused on understanding the accessibility challenges across various programming activities such as pair programming and code reviews, including participants' experiences learning and performing UI development. We conducted semi-structured interviews with 23 adult programmers with visual impairments (19 men, 4 women) between July 2019 and March 2020 [11]. Participants were between 21 and 73 years old. They hailed from the United States, India, China, and Europe and included software engineers, data analysts, freelancers, and researchers. The interviews elicited rich accounts about participants' collaboration with other developers and designers, including details on challenges and workarounds they identified to work in contexts primarily designed for sighted developers.

In the second study, we focused on how the use of UI frameworks and libraries shaped the workflows of programmers with visual impairments [10]. We first scraped all emails from the program-l mailing list between 2018 and 2021. Next, we identified

CHAPTER 8 UI DEVELOPMENT EXPERIENCES OF PROGRAMMERS WITH VISUAL IMPAIRMENTS IN PRODUCT TEAMS

the emails that seemed related to UI development by going over the posts' subject lines. Finally, we randomly sampled 96 emails and their replies from the filtered set of emails. This was followed by semi-structured interviews with 18 programmers with visual impairments (17 men, 1 woman) between 19 and 46 years old. We recruited participants from the mailing list and r/Blind subreddit. The eligibility criteria included that programmers should have experience using UI frameworks such as React Native, Flutter, Angular, etc.

We refer to participants from study 1 and study 2 as P*-I (e.g., P1-I) and P*-II (e.g., P1-II), respectively. Quotes from the program-1 mailing list are indexed as T* (e.g., T1). We obtained prior approval from the University of Michigan's Institutional Review Board for both studies. We presented \$15 and \$30 USD gift cards (or their equivalent in local currency) to participants in studies 1 and 2, respectively.

Analysis and Limitations

For both studies, we used open coding followed by inductive and deductive coding to organize the data into high-level themes pertaining to UI development and collaboration. The research team met weekly to discuss the emerging themes and wrote memos to identify the missing details in the data to refine the questions for subsequent interviews.

Despite our best efforts to have a balanced gender representation, our participants' sample was skewed toward men. This was due to the software engineering field and the online communities we recruited from being largely male-dominated. Another limitation of our studies is that our participants and the mailing list members reported different vision-related disabilities. Since any disability falls on a spectrum, we refrained from analyzing the data based on the onset and the nature of the visual impairment. Instead, we distinguish between screen readers and assistive technologies such as screen magnifiers. Our findings report on developers' experiences who primarily rely on screen readers and scope our recommendations to people interested in designing for the audio modality.

Findings

We first discuss the collaborative experiences of programmers with visual impairments with sighted designers and developers, followed by their efforts in sharing their contributions broadly in the workplace.

Collaborating with Sighted Designers

UI development in software development teams often includes design discussions where developers and designers arrive at a mutual understanding of the UI's form, functionality, and interactions. These discussions are centered around visual artifacts such as wireframes and design documents, which specify the design details for developers to utilize in UI construction. We found that the design specifications could range from detailed to high-level. A detailed set of specifications stated the colors, size, and placement of individual GUI elements, enabling our participants to plug the specifications directly into the UI code. However, it was essential to provide the documents in accessible file formats such as word documents or PDFs correctly tagged for screen readers.

A high-level design document lacked strict rules and guides, requiring the developer to approximate size and placement from the visuals provided. Our participants shared that they needed to seek sighted assistance more often with loosely defined design documents. They would reach out to sighted friends, colleagues, and family members to spot-check the interface they were developing. Specifically, they would ask sighted people to verify that all UI elements lay within screen margins, did not overlap, and were visible on the screen. Additionally, if tasked with making decisions regarding the visuals (color selection, font selection, etc.), they would ask sighted people to determine if the UI looked aesthetically pleasing.

Through our studies, we found that collaborating with designers entailed a lot of communication and discussion to understand the UI's design. Participants mentioned that designers often struggled to describe the interface and omitted essential details that could help the participants visualize the interface. They felt responsible for framing and asking the right questions as well as narrowing down their questions to elicit the macro and the granular details from designers progressively:

When I put the question very precise one [...], they answer and they are eager to answer. But if I ask, for example, can you give me an idea of the layout [...], they used to say maybe much more than I need or maybe they miss some parts. It's to me, just to try to at the beginning, to ask very, very precise questions [...]. It's not always easy to know which are the elements that they want to put on the page, so I try and to refine step by step.

—P13-I (quoted from [11])

CHAPTER 8 UI DEVELOPMENT EXPERIENCES OF PROGRAMMERS WITH VISUAL IMPAIRMENTS IN PRODUCT TEAMS

The discussions were more extensive when the UI in question had to be built from scratch using HTML and CSS. The conversations had to crystallize details regarding size, appearance, interaction, and the relative placement of widgets. If the team utilized existing components from a UI framework (e.g., Bootstrap, React Native, etc.), the communication became simpler – designers could cite the component to be used and state the expected modifications, and our participants could import them into the source code.

Several posts on the mailing list and accounts from our participants suggested that collaboration with designers helped programmers with visual impairments delineate between design and development roles and made them more confident about their programming skills. They shared how they came to understand that the ability to see had little to do with the ability to do UI development; designers were responsible for making the interface user-friendly and visually appealing, and as developers they were responsible for implementing the designers' ideas:

My boss brought our company's graphic designer into my department to help. He has taken my super-simple UI and turned it into something my company could show off. So there definitely is a certain art to it and vision is not the issue.

—T15-II (quoted from [10])

Collaborating with Sighted Developers

When doing UI development, sighted developers often use GUI builders provided within IDEs such as Android Studio and Visual Studio. These are based on the WYSIWYG (What You See Is What You Get) paradigm. Users can drag and drop the widgets, modify their size, and adjust the relative placement to quickly create the UI with mouse-based interactions. However, mouse interactions are inaccessible to programmers with visual impairments, and GUI builders seldom support UI creation through keyboard shortcuts. Our participants reported typing the XML code by hand to create the UI. Accessible GUI builders were far and few in between, existing for select programming languages and IDEs. The different approaches to UI development led to some tensions during collaboration, which we describe in the following.

Our studies revealed that typing the UI code often meant dealing with verbose source code for programmers with visual impairments. It presented challenges in code readability and navigation with screen readers. The problem was exacerbated when modifying the UI, requiring them to recalculate the values for dimensions and positions and update the source file throughout. Their sighted colleagues could adjust these values by looking at the UI in the GUI builders. The XML code that represents the UI is often nested, which made it difficult to identify the location of visual parameters that needed to be updated:

I just found myself overwhelmed by the number of options and layouts with very little idea how to make sure they do what I want. I lose track once I am about two levels deep into the user interface element structure.

—T2-II (quoted from [10])

The use of GUI builders also led to generic variable names for UI controls, which further affected navigation. Since sighted programmers did not deal with raw code, they did not always realize how the poor variable names could cause confusion for their colleagues:

Putting 2 buttons on a WPF designer surface, then tabbing around, forces the screen reader to say “grid,” “button,” “button,” “window.” What button is what one? The (WPF) designer needs to assign default names to controls dropped on the designer surface and expose them to screen readers.

—T56-II (quoted from [10])

One of our participants shared that he had given strict instructions to his team to modify the variable names to descriptively map to UI controls' functionality before sharing the source code with him.

Incorrect focus or tab order was another common issue due to the differences in approaches to UI development among our participants and sighted programmers. We found that sighted people would randomly drag and drop the UI components when creating the UI. While this did not alter the visual representation of the UI, it significantly affected the interaction for people with visual impairments by altering the focus order on screen readers (see Figure 8-1), thereby impacting the debugging and testing workflows for programmers with visual impairments.

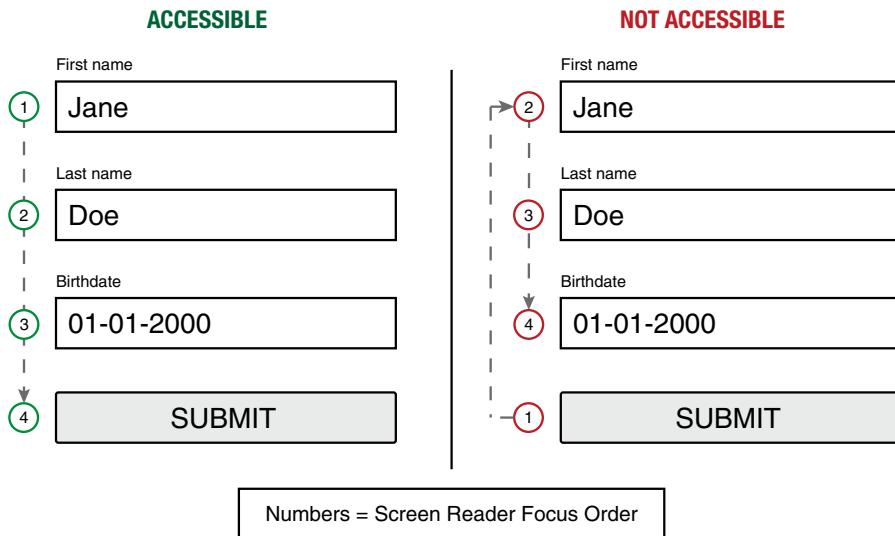


Figure 8-1. Effect of dragging and dropping form elements in the wrong order on screen reader focus order

Participants also shared that they often had to advocate for adoption of accessible UI frameworks and code editors. These decisions were often taken by developers on the team collectively. However, if the choice suffered from poor accessibility, it would impact the productivity of our participants and ultimately affect the collaboration workflows among developers. One participant shared that he convinced his team to program the Android application using Xamarin in Visual Studio instead of writing the code natively in Android Studio. The participant found the programming environment offered by Xamarin and Visual Studio more accessible and, therefore, a more productive option for him.

Occasionally even the more accessible choice could be rendered inaccessible in part due to software updates. For instance, P17-I and his team used Visual Studio. However, upgrading to the 2019 version from the 2017 one impacted certain settings with the screen reader. P17-I had to work out of both versions to maintain an accessible workflow for himself as well as keep the project compatible with the rest of the team.

When programmers with visual impairments joined existing projects where the team had already made the decisions about the technical stack, they had to deal with legacy UI code. Participants shared how sighted developers were unlikely to have implemented accessibility for screen readers while developing the UI. The lack of accessibility kept them from interacting and experiencing the UI independently, preventing them from building a full context of the project. In addition, it could interrupt debugging and testing workflows. It was also difficult to add the relevant accessibility modifiers, such

as ARIA labels¹ or APIs, to legacy code to make it accessible. The decision to do so could also require approval from senior management, who would base it on the time investment in making the code accessible vs. the impact on the developer's productivity without these changes. Plus, asking the team to improve legacy UI's accessibility as a new team member foregrounded the developer's disability and could lead them to form misperceptions about their ability as a programmer.

Sharing Contributions in the Workplace

We found that accessibility challenges would also hinder programmers with visual impairments from sharing their code contributions more broadly. The stakes differed significantly when they had to present internally to the team vs. when they had to demonstrate their work externally to stakeholders and clients. In the case of the former, inaccessible tools and work practices would prevent our participants from participating in meetings such as discussions of the system architecture. One participant (P16-I) mentioned that after his annual review, he and his manager mutually figured out a way for him to contribute to such team discussions, including creation of diagrams and visuals. However, not every participant had a similar positive experience. They had to often forgo participation in these activities as primary contributors and let other team members take the lead. This ultimately could impact their career trajectories within the organization.

When presenting to clients and stakeholders, participants expressed concerns about the UIs glitching or breaking down due to the accessibility issues in the technical stack. These could suggest poor quality of work by them and their team. One participant mentioned that he sometimes recorded the presentation ahead of time to avoid demoing the UI live to people. Another participant shared that he preferred handling the narration and had a sighted colleague operate the UI using mouse interactions. This avoided issues that might arise due to poorly operating the UI with screen readers:

If it's a demo for stakeholders or an audience outside of the team [...], I ask someone else from my team to drive the visuals and I do the technical narration [...]. It's hard when I'm the one dealing with the visuals.

—P3-II

¹ Accessible Rich Internet Applications (ARIA) is a set of roles and attributes that define ways to make web applications and content more accessible for users with disabilities.

Discussion

Our research studies were motivated by the limited reporting of collaboration within mixed-ability teams and the fast rise in UI development jobs, whose lack of accessibility can tilt the playing field against programmers with visual impairments. Our findings highlight the challenges that programmers with visual impairments face when collaborating with sighted developers and designers as they do UI programming. In this section, we discuss how our findings can inform the practices within mixed-ability software engineering teams as well as the design of UI frameworks.

It is not enough for the “tools” that most people associate with software and UI development – IDEs, code editors, browsers, etc. – to be accessible. Accessibility breakdowns can also occur as the result of collaborative practices or communication tools. For example, pair programming can limit the roles of programmers with visual impairments [11] unless accommodations are made to enable them to serve as the observer. In the case of UI programming, relying solely on inaccessible collaborative artifacts (like visual drawings and wireframes) or inaccessible tools for communication can restrict the roles in which programmers with visual impairments can serve. Providing nonvisual alternatives to these (e.g., text-based descriptions, accessible PDFs, tactile printouts, etc.) will enable more inclusive team discussions and collaboration.

Further, many tools can be inaccessible in subtle ways. For example, a code editor might be working fine, but updates may modify certain settings. Similarly, editor plugins, add-ons, and configuration tools might create accessibility problems [11]. With a heavy reliance on third-party APIs and frameworks, the scope of tools that are essential for development has also expanded. For example, programmers might rely on cloud services with advanced configuration tools. For some programmers, these tools are just as important for their work as their IDEs, and accessibility problems can represent significant barriers to progress.

Sighted team members often do not realize the impact of inaccessible UI frameworks, programming tools, and documentation on their colleagues with visual impairments. For instance, designers would share loosely defined or inaccessible design documents that required programmers with visual impairments to ask precise questions about the UI. Developers would select inaccessible UI frameworks and code editors for development, in which case programmers with visual impairments had to either convince their team to switch to more accessible alternatives or work with the choices made by their colleagues. These findings illustrate the additional communication and articulation that programmers with visual impairments must perform in mixed-ability

teams. However, the workplace and the team have to offer an inclusive environment for programmers with visual impairments to communicate and advocate for accessible software and practices. It is also important to note that our participants were often the only team members advocating for accessible tools and educating team members [10, 11]. This work on advocacy and education can come with a social cost within their workspace and represent additional hidden work and can have implications for their future career prospects.

One way to inform sighted team members about accessibility is by improving the documentation of UI frameworks and programming tools. If the official documentation emphasizes compatibility with screen readers, much like how they emphasize compatibility with various operating systems, it can be the first step toward enabling informed software choices among teams. It would also prompt people behind programming tools to think more deeply about accessibility, and they describe it in the documentation. For example, UI frameworks would then be more mindful about calling their components as “out-of-the-box accessible” and use more accurate descriptions.

Large technical companies tend to have their own internal code authoring guidelines, which software engineers are expected to follow. Prioritizing the code writing preferences of programmers with visual impairments can be made part of these code styling guidelines. These include (1) using camel case for variable names to enable appropriate announcement on screen readers, (2) avoiding generic identifiers, (3) modifying the generic names assigned to UI elements by GUI builders, (4) including descriptive comments to facilitate quick search and navigation in source code, and (5) following the screen reader focus order when creating the UI. This would not only ensure accessible UI code from sighted engineers but also reduce the articulation that programmers with visual impairments have to do in the workplace.

Conclusion

In this chapter, we summarized and discussed findings specific to UI programming from our prior studies. Our reporting of the challenges and recommendations can be valuable to the software engineering teams, researchers, and creators of UI frameworks and tools. It can help them make the profession of UI development and the related collaborative activities more accessible to programmers with visual impairments.

Bibliography

- [1] Andy Borka. *Developer Toolkit*. 2019.
- [2] Claire Kearney-Volpe, Chancey Fleet, Keita Ohshiro, Veronica Alfaro Arias, and Amy Hurst. Making the elusive more tangible: remote tools & techniques for teaching web development to screen reader users. In *Proceedings of the 18th International Web for All Conference*, 1–14, 2021.
- [3] Claire Kearney-Volpe and Amy Hurst. Accessible web development: Opportunities to improve the education and practice of web development with a screen reader. *ACM Transactions on Accessible Computing (TACCESS)*, 14(2):1–32, 2021.
- [4] Jingyi Li, Son Kim, Joshua A. Miele, Maneesh Agrawala, and Sean Follmer. Editing spatial layouts through tactile templates for people with visual impairments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–11, 2019.
- [5] Sean Mealin and Emerson Murphy-Hill. An exploratory study of blind software developers. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, 71–74, 2012.
- [6] Kirk Norman, Yevgeniy Arber, and Ravi Kuber. How accessible is the process of web interface design? In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, 1–2, 2013.
- [7] Stack Overflow. Stack overflow developer survey results. 2019.
- [8] Stack Overflow. Stack overflow developer survey. 2020.
- [9] Stack Overflow. Stack overflow developer survey. 2021.
- [10] Maulishree Pandey, Sharvari Bondre, Sile OModhrain, and Steve Oney. Accessibility of ui frameworks and libraries for programmers with visual impairments. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–10, IEEE, 2022.

CHAPTER 8 UI DEVELOPMENT EXPERIENCES OF PROGRAMMERS WITH VISUAL IMPAIRMENTS IN PRODUCT TEAMS

- [11] Maulishree Pandey, Vaishnav Kameswaran, Hrishikesh V. Rao, Sile O'Modhrain, and Steve Oney. Understanding accessibility and collaboration in programming for people with visual impairments. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1-30, 2021.
- [12] Venkatesh Potluri, Liang He, Christine Chen, Jon E. Froehlich, and Jennifer Mankoff. A multi-modal approach for blind and visually impaired developers to edit webpage designs. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 612–614, 2019.
- [13] Ashrith Shetty, Ebrima Jarjue, and Huaishu Peng. Tangible web layout design for blind and visually impaired people: An initial investigation. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 37–39, 2020.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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

The Role of Ethics in Engineering Fair AI (and Beyond)

Brittany Johnson, George Mason University, USA.*

Justin Smith, Lafayette College, USA.

The decisions we make are typically influenced by our principles and values, or our *ethics*. Ethics guide us toward what we believe to be the best course of action. This decision-making process is particularly important in the context of software development, as our society relies on software for critical systems and increasingly relies on artificial intelligence (A)-driven software in these systems. Ethical decisions made by software developers of these systems can therefore have amplified impacts. Thus, it is essential to understand and support ethical practices in software development.

Rarely do software developers actively think about or openly discuss whether their actions are ethical. However, whether or not developers consider ethics when making decisions, their decisions often have quite tangible impacts on society that regularly capture media attention. By examining the software development process through the philosophical lens of ethics, we can better understand the types of decisions that software developers make and how to better support them in refining and applying their ethical principles.

We argue that diversity, equity, and inclusion are three core principles that ought to be considered under the umbrella of ethical software development. If we encourage ethical practices, we are inherently making foundational strides toward these principles. In this book chapter, we will bring these aspects to the forefront of our discussion of ethics in software development. More specifically, in this chapter we will

- Examine a *case study* that demonstrates the importance of ethics in an AI-driven software development environment by highlighting the potential harms when unethical software persists.
- Explore *ethics from a philosophical perspective* and discuss how existing frameworks can be applied in the context of software development.
- Discuss *existing efforts in understanding and supporting ethical decision-making in software development*, specifically how software developers think about the role of ethics in their decision-making and signals they use to indicate when and how they are thinking about ethics.
- Outline existing tools and techniques that support ethical decision-making and other ways we can work toward explicitly considering the ethics behind decision-making throughout the process of building and maintaining software.

Making a Case for Ethics

There are many examples that help emphasize the importance of making explicit ethical considerations when building technology [9, 13]. Anecdotes range from the “Dieselgate” scandal where Volkswagen vehicles were programmed to evade emission regulations [12] to racial bias in algorithms and software used in criminal justice [1, 5, 8, 21]. To illustrate the value of explicit ethical considerations, in this section we will examine one of these many case studies. Namely, we will examine the work done by Obermeyer and colleagues to identify racial bias, and ultimately improve equity, in healthcare tech [10, 14].

Despite the numerous examples of tech gone wrong, more and more we are seeing technology injected into everyday societal interactions and decision-making processes. There is no silver bullet to solving the problem of biased technology. However, there are actionable ways we can work toward reducing and eventually eliminating inequitable outcomes.

“Dissecting Racial Bias”

In 2019, Obermeyer and colleagues identified bias against Black patients in a diagnostic and treatment algorithm used widely across the United States [14]. Collaborating with an academic hospital, they collected and analyzed the algorithmic risk scores for 6,079 patients who self-identified as Black and 43,539 patients who self-identified as White. Their data covered 11,929 and 88,080 patient-years, respectively, where 1 patient-year refers to the data that was collected from one patient in one calendar year.

The data and various analyses conducted by Obermeyer and colleagues pointed to a higher need for healthcare among Black patients. In other words, they found that Black patients exhibited more “illness burden,” or a greater incidence rate of chronic illness. However, their analyses also found that the algorithm in question erroneously assigned similar risk scores to ill Black patients and healthier White patients.

The root cause of this bias was the fact that the algorithm used *healthcare costs* as a proxy for health risk. Black patients with higher risks for chronic illness had similar healthcare costs when compared with healthier White patients. One explanation for this was a correlation between race and income. Black patients in the study were more likely to have lower incomes and thus less likely to have access to and engage with the medical system (even when insured). This lower engagement has been studied and linked to factors such as reduced trust and differences in access to healthcare, among others.

So what action can we take to help balance the scales?

Seeking a Solution

The results of the studies conducted by Obermeyer and colleagues point to the impact decision-making can have on technological outcomes. By choosing cost as the predictor, given the fact that Black patients are generating fewer medical expenses, accurate predictions yielded inequitable outcomes.

While cost prediction as a proxy for health risk score accuracy seems reasonable, according to Obermeyer and colleagues, we could and should be doing more. One suggestion was that perhaps it makes more sense to focus on “future avoidable costs,” which would be those associated with emergency care and hospitalization. Or maybe we move away from predicting the costs of healthcare and instead use a more direct measure of health, such as the number of active chronic health conditions.

To better understand the potential for label choice to reduce bias in this case, Obermeyer and colleagues conducted a series of experiments that started with the development of three new models. These models are used to predict the following outcomes:

- **Total cost in year t :** Overall, how much will patients spend on healthcare in a year?
- **Avoidable cost in year t :** These are just the costs associated with emergency room visits and hospitalizations.
- **Health in year t :** This is determined based on the number of chronic condition flare-ups in year t .

They trained each model on a random two-thirds subset of their data and tested their models on the remaining one-third. They also excluded race from the training data. The three models all performed reasonably similarly at predicting the three outcome variables (total cost, avoidable cost, and health in year). However, depending on the model chosen, the composition of the highest-risk group varied drastically. For the models trained to primarily predict the healthcare costs of patients, Black patients comprised 14.1% of the highest risk category. On the other hand, for models trained to primarily predict chronic conditions, Black patients comprised 26.7% of the highest risk category.

This variation in outcomes based on label choice could be seen as an insurmountable challenge and inherent feature of learning algorithm use. We see it as an opportunity, as Obermeyer and colleagues did, to be more informed and take intentional action to actively consider the ethical implications of the choice we make while developing software.

Following the publishing of these findings, the authors joined forces with the company that developed the original cost predictor algorithm to make these improvements in practice [10]. While not always an easy or straightforward process, decision-making is the control developers have over the outcomes of the technology they produce regardless of the domain.

The Philosophy of Ethics: Defining “Ethical”

Notions of ethics have been discussed by philosophers since long, long, before the Obermeyer study and the rise of modern software development and the increased use of AI. In this section we will briefly summarize a few of these pre-existing philosophical frameworks and explore how they can be applied in the context of software development. Here, the key question we'll focus on is one of *normative ethics*: what constitutes “ethical” behavior? Three prevailing philosophical theories that approach this question are *deontology*, *consequentialism*, and *virtue ethics*.

Deontology: The theory of deontology, or duty ethics, states that behavior can be deemed ethical or unethical by strictly applying some universal set of moral rules or laws, for instance, “do no harm” or “do not steal.” Under this theory, we can disregard situational factors such as an individual’s intent as well as the consequences of their actions, so long as their behavior adheres to a particular set of moral rules. Of course, philosophers disagree over which moral rules should be used and whether they should be derived from a divine power, nature, or some other source. In the context of software development, many organizations, like the ACM, have begun to capture and describe what could be considered a set of domain-specific moral principles. For instance, the ACM Code of Ethics includes principles like “Respect privacy” or “Be honest and trustworthy.”¹ (For a more thorough discussion of codes of conduct, which have become increasingly pervasive in open source software development, refer to [Chapter 17, “Codes of Conduct in Open Source”].)

Consequentialism: The theory of consequentialism argues that the *consequences* of our actions are more important than any set of rules or laws. Positive actions are those that result in some benefit to the actor or to society. This theory is tied to notions of utilitarianism, or the argument that actions are good if they maximize benefit for the majority of people. A common critique of utilitarianism is that actions that consistently benefit the majority may also consistently harm minorities. In the field of software engineering, we might view grey-hat hackers (security engineers who identify vulnerabilities in systems, sometimes without prior authorization) as consequentialists. Grey-hat hackers may not subscribe to rules like “do not hack.” Instead they could argue that the consequences of their actions (revealing potentially harmful vulnerabilities before malicious actors are able to exploit) provide greater benefit to society.

¹www.acm.org/code-of-ethics

Virtue ethics: The theory of virtue ethics can be traced back to the ancient Greek philosophers Plato and Aristotle. In contrast with deontology and consequentialism, virtue ethics claims that ethical behavior stems from who we are as people, rather than a set of rules or the consequences of our actions. This perspective argues that it is more important to have and to develop good character over one's lifetime. Under this framework we might think of individuals as "good" or "bad" software developers.

Understanding Ethics in Practice

In this section, we outline previous studies that empirically examined the state of how practitioners view ethics. Here we focus both on previous works that examined AI-driven development practices and more general software development practices. In contrast to studies that aim to evaluate interventions, which will be discussed in later sections, research in this section studied the question of ethics more broadly.

Much of the limited work in this space has been conducted by Vakkuri and colleagues [17, 18, 19, 20]. In a series of studies, this research group has conducted case studies [17, 20], semi-structured interviews [18], and a survey of practitioners [19]. Their findings characterize how software developers implement (or disregard) ethics in some types of AI-driven systems. For instance, in startup-like environments, Vakkuri and colleagues report that developers take responsibility for issues related to software development, such as finding bugs, and they generally care about ethics on a personal level. However, little is done to tackle ethical concerns that arise during product development [18]. A separate study reveals a similar disconnect in the development of autonomous cyber-physical systems – developers unanimously indicate that they consider ethics useful to their organization, but also unanimously report that their practices do not account for ethics [20].

In another attempt to better understand how developers are thinking about and applying ethics in practice, specifically in the context of AI technologies, Vakkuri and colleagues surveyed over 200 developers across over 200 software companies to gain insights on the state of practice in AI ethics [19]. Their sample included companies that build AI-enabled technologies as well as those that do not. The survey asked developers to evaluate the importance of various ethical concepts, such as transparency and responsibility, and recount experiences dealing with software unpredictability. Their findings suggest that while ethics is not completely absent from the software landscape, the state of practice is a mixed bag and mostly immature or undefined.

Most recently, Lu and colleagues conducted an empirical study with 21 practitioners at an Australian research agency to better understand how AI ethics is being considered and applied in practice [11]. They found that while AI practitioners are sometimes explicitly taking ethics into consideration, they lack guidance on how to operationalize ethical principles. Based on their findings, as well as other existing efforts, they offer a template, or list of patterns, that AI practitioners can use to better integrate ethical practices into their work.

While there has been a recent increase in interest around ethics in software development due to the rapid advancement and integration of AI technologies, research on ethics in software development has been happening for decades. Much of this work involved case studies aimed at better understanding the role of ethical decision-making in software development and how to adequately support it. For example, Stapleton studied the effects of not making ethical considerations in large-scale software systems and found that ethical issues may be more complex than they seem but that lacking ethical considerations can have an impact on project outcomes [16]. Others have also conducted case studies to better understand ethics in practice in various domains and contexts [2, 7, 15] and studied the effects and implications of code of ethics in practice [6, 13].

Supporting Ethical Decision-Making

Researchers and practitioners have proposed numerous interventions to better support the ethical software development, most of which aim to support the development of AI and machine learning (ML) software systems. Some of these contributions take the form of actionable frameworks and guidelines, while others are tools that can be used for various tasks throughout the development process. In this section, we outline some of the many existing contributions to ethical software development practices. All of the works discussed in the following, along with others, can be found in our short paper on ethical software development practices [9].

Ethical Frameworks

Over the years, there have been numerous contributions to ethics in the form of frameworks, principles, or guidelines. For many existing efforts, an important goal to achieve is ethics by design. Prior efforts have proposed the concept of “ethical by design”

and provided some best practices in general and specific domains, such as natural language process (NLP) systems. The frameworks that emerged to accomplish the goal of ethical by design vary. Some propose new entities that can be integrated into existing processes to explicitly review and support ethical considerations from algorithmic design all the way to system design.

Other frameworks that aim to support ethical design take a more stakeholder-centric approach that centers on effectively integrating stakeholders for identifying and addressing potential ethical concerns. This includes an ethical by design manifesto, which outlines principles for supporting various software stakeholders when attempting to integrate ethical concerns in the design process. Considerations in the space of ethical by design frameworks include offering alternatives to support shared, decision-based usage and designing through empathy for users. All of these efforts aim to increase accountability and responsibility for potential impact on users and other stakeholders when designing software systems.

While there is a concerted effort to integrate ethical considerations before development, some existing frameworks aim to support ethical considerations throughout the entire software development pipeline. These frameworks provide guidelines to be followed before, during, and after system development.

Many of these efforts centered on supporting both ethical decision-making and providing practitioners with insights into the consequences of their decisions. A specific example of this is the “ethics-aware software engineering” framework, the steps for which are depicted in Figure 9-1. This framework centers on a more exhaustive view of ethics beyond just artificial intelligence technologies and the engineers developing the software. It starts with the *articulation* of ethics requirements and organizing them into an ethics *specification*, followed by *implementation* of both software and processes centered on that specification and lastly *verification* and *validation* of the software against the ethics specification. As implied by Figure 9-1, this is intended to be an iterative process.

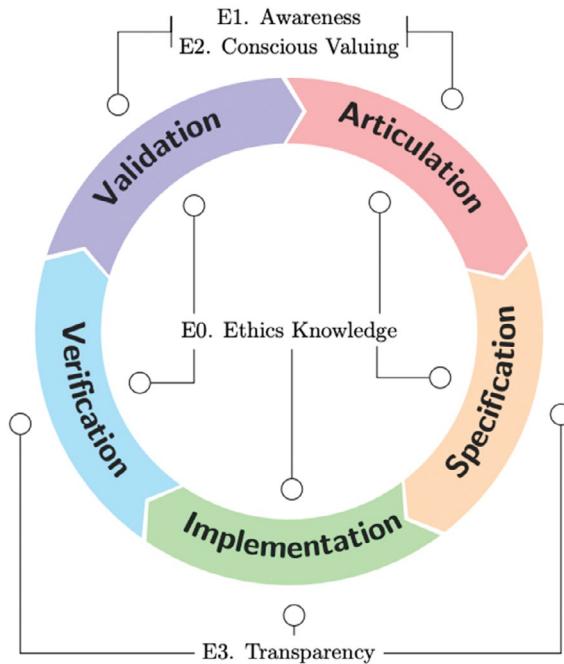


Figure 9-1. Visualization of the methods from ethics-aware software engineering [3]

The researchers who proposed this framework describe four “enablers” that can be used to facilitate the adoption of ethics-aware software engineering:

- **Ethics knowledge** is required for each phase of the framework and refers to the process of specifying what is and is not considered ethical. This speaks directly to the articulation and specification portions of the framework.
- **Awareness** connects practitioners to ethical issues and their potential impacts (which as we've mentioned is an important step in realizing ethical decision-making).
- **Conscious valuing** goes beyond awareness to placing value behind ethical issues that pertain to the software system. It's generally after adding value that requirements start to form and specifications can be documented.

- **Transparency** is important in software development to ensure that the behavior of the artifact and the development processes align with specifications and ethical requirements. This is achieved by making both the processes and the artifact's behavior visible for validation.

Another specific example of an ethical framework that targets AI-based systems is Australia's AI Ethics Framework [4]. Their framework is separated into two components: *core principles for AI* and *toolkits for ethical AI*. Core principles range from the common mantra "do no harm" to promoting regulatory and legal compliance of AI technologies. The framework's focus on implementing ethical AI outlines tool support that ranges from assessing impacts and risks to supporting collaboration and consultation.

Efforts to develop frameworks and guidelines that can be applied in practice are a meaningful step toward realizing ethical software practices. However, studies have shown that the existence of guidelines may not be enough to effectively support ethical decision-making. In both research and practice, there have been efforts to bridge the gap between ethics principles and in-practice. Much of the effort in this space has been centered on connecting principles to action. For some frameworks this means grounding guidelines in strategies from relevant domains to support actionability. For others, it means mapping elements of the framework to relevant information, such as concerns and actions, that directly point to rationale and course of action. The goal of these action-focused frameworks ranges from individual developer support all the way to industry-level support.

Ethics Tools

Separate from the ethical frameworks that continue to emerge, there is also a steady increase in the tools that are being developed to support ethical software development practices. Many of these efforts have focused on increasing software fairness, and most aim to support development of AI-driven systems.

One of the first published efforts at providing fairness tooling was FairML,² a toolbox aimed at mitigating bias in black-box machine learning models. FairML provides support for evaluating the effects of inputs on a model's decision-making to determine the effects on fairness.

In 2018, IBM introduced AI Fairness 360 (AIF360), a Python toolkit for measuring and mitigating bias in machine learning models.³ The toolkit provides an exhaustive

²<https://github.com/adebayoj/fairml>

³<https://aif360.mybluemix.net/>

and extensible set of open source models and algorithms, along with fairness metrics for models and datasets. Similar to AIF360 is Aequitas, a Python toolkit for systematic auditing of model fairness.⁴ As with most fairness tools, Aequitas was designed to be used by data scientists; however, it was also designed for use by policy makers. It also provides tooling for analyzing bias in datasets and determining optimal metrics for a given situation.

In the same year, we saw the introduction of Themis, the first tool designed to test any kind of software for discrimination.⁵ Similar to other available fairness tools, Themis works based on common definitions of fairness. In contrast, Themis allows for measuring and detecting bias in software separate from any model that may (or may not) be integrated into the software. Themis also generates tests that engineers can take advantage of in their own test suites.

The contributions to this space, specifically with respect to ethics in AI and ML software systems, continued in the years to come. This includes tools like fairkit-learn,⁶ FairVis,⁷ Fairlearn,⁸ FAT Forensics,⁹ and the LinkedIn Fairness Toolkit (LiFT)¹⁰ that support detecting and measuring bias during model training and selection. LiFT, introduced in 2020, is a Scala/Spark library that supports measuring and mitigating bias in large-scale machine learning workflows. Unlike other fairness tools, fairkit-learn, FairVis, and Fairlearn use visualizations to support the ability to explore and discover biases in machine learning models. Fairkit-learn and Fairlearn provide additional unique features, such as interactive comparison and fairness and performance tradeoffs. FAT Forensics is another unique tool that also supports the inspection of accountability and transparency aspects of machine learning software.

As we have outlined, there is no shortage of interventions available for attempting to support ethical decision-making during software development. Check out our paper on ethical practices to learn more about these efforts and others [9]. While none of these solutions provide a “silver bullet” for the problem of ethical decision-making in software development, collectively they provide hope for a more equitable technological future.

⁴ <https://github.com/dssg/aequitas>

⁵ <https://github.com/LASER-UMASS/Themis>

⁶ <https://github.com/INSPIRED-GMU/fairkit-learn>

⁷ <https://github.com/poloclub/FairVis>

⁸ <https://fairlearn.org/>

⁹ <https://github.com/fat-forensics/fat-forensics>

¹⁰ <https://github.com/linkedin/LiFT>

Summary

In this chapter, we made an argument for ethical decision-making as the umbrella over diversity, equity, and inclusion efforts. For those who skipped to the credits, let's recap ☺:

- Ethical concerns, such as fairness and safety, are addressable when explicitly considered.
- There is more than one way to define ethics, but all definitions are centered on the actions taken by an individual (or organization) and why.
- While ethics has a long history in the context of computing, most of the recent efforts have focused on ethics when developing machine learning- and artificial intelligence-based software systems.
- Research has provided some insights into ethical software development practices (though much more is needed to understand and support ethics in practice).
- There are numerous frameworks and tools available to support ethical software development practices, many of which target components of AI-based software systems.

By encouraging and supporting ethical decision-making with concepts like “ethical by design” and frameworks like ethics-aware software engineering, we can work toward realizing ethical software development in practice.

Bibliography

- [1] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, May 23, 2016. propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.
- [2] Josephina Antoniou and Andreas Andreou. Case study: The internet of things and ethics. *The Orbit Journal*, 2(2), 2019.
- [3] Fatma Basak Aydemir and Fabiano Dalpiaz. A roadmap for ethics-aware software engineering. In *Software Fairness (FairWare)*, 15–21, IEEE, 2018.

- [4] Dave Dawson, Emma Schleiger, Joanna Horton, John McLaughlin, Cathy Robinson, George Quezada, Jane Scowcroft, and Stefan Hajkowicz. Artificial intelligence: Australia's ethics framework – a discussion paper. 2019.
- [5] Anthony W. Flores, Kristin Bechtel, and Christopher T. Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to machine bias: There's software used across the country to predict future criminals. And it's biased against blacks. *Fed. Probation*, 80:38, 2016.
- [6] Jan Gogoll, Niina Zuber, Severin Kacianka, Timo Greger, Alexander Pretschner, and Julian Nida-Rü melin. Ethics in the software development process: from codes of conduct to ethical deliberation. *Philosophy & Technology*, 34(4):1085–1108, 2021.
- [7] Nicolas E. Gold and Jens Krinke. Ethical mining: A case study on msr mining challenges. In *Proceedings of the 17th International Conference on Mining Software Repositories*, 265–276, 2020.
- [8] Kashmir Hill. Wrongfully accused by an algorithm. *The New York Times*, August 3, 2020. [nytimes.com/2020/06/24/technology/facial-recognition-arrest.html](https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html).
- [9] Brittany Johnson and Justin Smith. Towards ethical data-driven software: filling the gaps in ethics research & practice. In *2021 IEEE/ACM 2nd International Workshop on Ethics in Software Engineering Research and Practice (SEthics)*, 18–25, IEEE, 2021.
- [10] Heidi Ledford. Millions of black people affected by racial bias in health-care algorithms. *Nature*, 574(7780):608–610, 2019.
- [11] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, David Douglas, and Conrad Sanderson. Software engineering for responsible ai: An empirical study and operationalised patterns. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, 241–242, 2022.
- [12] Nazanin Mansouri. A case study of volkswagen unethical practice in diesel emission test. *International Journal of Science and Engineering Applications*, 5(4):211–216, 2016.

- [13] Andrew McNamara, Justin Smith, and Emerson Murphy-Hill. Does ACM's code of ethics change ethical decision making in software development? In *Foundations of Software Engineering*, 729–733, 2018.
- [14] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.
- [15] Ehsan Sargolzaei and Mohammad Nikbakht. The ethical and social issues of information technology: A case study. *International Journal of Advanced Computer Science and Applications*, 8(10), 2017.
- [16] Larry Stapleton. Ethical decision making in technology development: a case study of participation in a large-scale information systems development project. *Ai & Society*, 22(3):405–429, 2008.
- [17] Ville Vakkuri, Kai-Kristian Kemell, and Pekka Abrahamsson. Implementing ethics in ai: Initial results of an industrial multiple case study. In *International Conference on Product-Focused Software Process Improvement*, 331–338, Springer, 2019.
- [18] Ville Vakkuri, Kai-Kristian Kemell, Marianna Jantunen, and Pekka Abrahamsson. This is just a prototype: How ethics are ignored in software startup-like environments. In *International Conference on Agile Software Development*, 195–210, Springer, 2020.
- [19] Ville Vakkuri, Kai-Kristian Kemell, Joni Kultanen, and Pekka Abrahamsson. The current state of industrial practice in artificial intelligence ethics. *IEEE Software*, 2020.
- [20] Ville Vakkuri, Kai-Kristian Kemell, Joni Kultanen, Mikko Siponen, and Pekka Abrahamsson. Ethically aligned design of autonomous systems: Industry viewpoint and an empirical study. *arXiv:1906.07946*, 2019.
- [21] Aleš Zavřník. Algorithmic justice: Algorithms and big data in criminal justice settings. *European Journal of Criminology*, 18(5):623–642, 2021.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 10

Beyond Diversity: Computing for Inclusive Software

Kezia Devathasan, University of Victoria, Canada.*

Nowshin Nawar Arony, University of Victoria, Canada.

Daniela Damian, University of Victoria, Canada.

This chapter presents, from our research on inclusive software within the context of a diversity and inclusion-based STEM program at the University of Victoria, INSPIRE: STEM for Social Impact (hereafter Inspire).¹ In a society with an ever-increasing reliance on technology, we often neglect the fact that software development processes and practices unintentionally marginalize certain groups of end users. While the Inspire program and its first iteration in 2022 are described in detail in Chapter 26, “Software Engineering Through Community-Engaged Learning and an Inclusive Network,” here we describe our insights from an analysis of the development processes and practices used by the teams. We found that empathy-based requirements gathering techniques and certain influences on the software development teams’ motivation levels impact the teams’ ability to build inclusive software. This chapter begins with an explanation of the Inspire program and a discussion on what the term “inclusive software” actually means in our context before highlighting useful practices for designing inclusive software.

¹ inspireuvic.org

INSPIRE: STEM for Social Impact

The Inspire program at the University of Victoria was designed to empower underrepresented individuals in STEM with the ambition of increasing the retention of minority groups in STEM fields. The program consists of assigning enrolled students (both undergraduate and graduate) to a four-month-long internship on a community-led project based on students' expressed interest. Each project required students to solve a local community problem using their various STEM skills. As each project was closely tied to a real community partner, success in these undertakings yielded observable impacts for real clients, thus empowering our students. For example, we had a team of students create a website for the Victoria Brain Injury Society (VBIS) that was navigable by patients with acquired brain injuries. This allowed many patients to regain a sense of independence and autonomy over their own recovery program. We also had a team of students create a platform to help shelters coordinate available beds to assist women fleeing domestic violence or facing homelessness. As we had allowed students to choose which project they would work on, they were passionate about the cause of their projects, which has been shown to boost motivation levels [8].

Each team of four to five individuals were diverse in terms of academic background, and other salient characteristics, although it is important to note that students were not assigned to teams based on diversity. Among the 24 students in the program, 10 identified as female, 12 identified as male, and 2 preferred not to disclose their gender. Nineteen students came from the undergraduate level and five from the graduate level. The students were also diverse in terms of (1) academic background – they came from computer science, software engineering, electrical engineering, mechanical engineering, biomedical engineering, physics, chemistry, and business – and (2) ethnicity: our students were South Asian, East Asian, Black, Arab, Hispanic, Indigenous, and White. The teams went through several phases over the course of the four-month-long project. These phases followed the suggestions of enterprise design thinking (DT), an empathy-based approach to gathering and eliciting requirements from end users for creative projects such that end users have a voice in how a product is designed for them [2].

The students were asked to submit a weekly individual reflection in addition to a weekly team reflection. By analyzing these reflections, we were able to learn about the challenges faced by software developers in making inclusive software. Previous research already suggests that software developers find considering the needs of their diverse end users challenging [10]. Thus, we look deeper into this problem and discuss techniques that are effective in encouraging inclusive software development.

Research Methods

The student reflections as described previously were our main source of qualitative data, in addition to several interview and focus group sessions conducted with the Inspire students. We analyzed approximately 400 reflections and 20 interview and focus group sessions using Braun and Clarke's [1] six-step thematic analysis process, which is a widely used method to identify, analyze, and report patterns or themes that represent important information from the data in relation to the research question. The steps of this method are (1) familiarizing with the data, (2) generating initial codes, (3) searching for themes, (4) reviewing potential themes, (5) defining and naming themes, and (6) producing the report [1]. We analyzed the data for evidence of challenges designing inclusive software and also looked for how students responded to the practices outlined in the following in terms of their effectiveness in helping create inclusive software.

Inclusive Software

The term "*inclusive software*" seems intuitive upon first glance. We aim to create software that meets the needs of all potential end users of the software product. However, upon further thought, an overwhelming problem becomes apparent. The current state of knowledge defines inclusive software as software that is completely usable by *any* end user for any task [11]. To understand the magnitude of this problem, we can take a look at a widely used application such as Google Chrome. Can such an application realistically meet the needs of every possible end user? Which accessibility or usability issues should be prioritized during the development process? How should these issues be prioritized?

A problem with such a large scope is understandably difficult to tackle. As a result, modern software developments frequently fail to meet the needs of potentially diverse end users [5]. Unfortunately, this renders many software products difficult or unusable by certain user groups [5]. The idea of inclusive software itself brings an entirely new lens into the software development cycle, which traditionally consists of technical steps such as requirements defining, solution building, and testing [6]. Though effective, these traditional methods typically neglect human-centric issues. Previous work by Khalajzadeh et al. categorizes these human-centric issues into eight groups: Inclusiveness, Privacy and Security, Compatibility, Location and Language, Preference, Satisfaction, Emotional Aspects, and Accessibility. This work thus defines human-centric

problems as “*the problems that diverse end users face when using a software system, due to the lack of proper consideration of their specific characteristics, limitations and abilities.*” For now, we consider inclusive software as software that considers all of these human-centric issues, and we explore practices that help developers do so.

Repairing the Digital Divide

Before delving into the practices we can employ to create more inclusive software, it is important to recognize why the concept is so important in the first place. A movement toward creating more inclusive software can help ensure that a broader range of end users can use a software product. While this may include those with disabilities, it also should consider those of different backgrounds, cultures, genders, ages, languages, etc. Furthermore, when a software product can be used by more end users, the associated organization benefits as the chances of successful adoption increase.

Software products that contain biases discriminate against certain groups of end users, both subtly (perhaps in an underlying algorithm) and explicitly (a feature of the user interface). These biases have the potential to affect the health of an individual, community, and even entire organizations [12]. A popular example in the software engineering community that highlights the detriment of non-inclusive software is gender. Previous research suggests that many software applications, everything from programming environments to educational platforms, favor males [12]. Biases in such software thus have the potential to deter other genders from this technology. Gender, however, is only one dimension of discrimination. Truly inclusive software must mitigate such discrimination across multiple dimensions.

Software that fails to be inclusive or unethical software has an incredible impact on our world. With software dictating many facets of our lives [13] that we often don’t notice, an unintentional bias may be woven into our everyday lives. However, the biases seen in increasingly popular software developments such as machine learning algorithms likely come from those creating the software products [13]. For example, in 2015, Flickr’s image recognition feature was flagged for yielding racist results, as it was unable to accurately identify those of African descent [13]. This raises questions on whether or not the training set for this algorithm included enough samples of diverse races in the first place. With these considerations in mind, it becomes apparent that much of a team’s biases affect the inclusiveness of the software that they produce. This leaves us with the question: *despite inevitable human biases, what software development practices help teams create inclusive software?*

Design Thinking for Inclusive Software

All five projects in the Inspire program had a very unique subset of end users, each containing different dimensions of diversity that the teams had to consider. In the Inspire program, we relied on the expertise of each community partner to help us decide on how inclusive the software produced by each student team was. For example, the team that created a website for patients with acquired brain injury had to consider many needs of various patients, as brain injuries are each unique in terms of the cognitive deficits they impose. The community partner at the Victoria Brain Injury Society was pleasantly surprised at how accommodating the features of the website were. Another Inspire project resulted in a system to track visitation patterns to a local nature sanctuary in order to protect endangered species. The community partner for this project was again delighted with how our students were able to solve this problem in a manner that worked for all potential site visitors and respected all privacy regulations and concerns. Our third project required Inspire students to come up with creative requirements elicitation methods to engage youth in discussions surrounding climate change and climate change anxiety. This was especially challenging as youth are particularly vulnerable to negative emotions surrounding climate change and can also be more challenging to engage for sustained periods of time. The fourth project in Inspire was especially concerned with gathering requirements in a delicate and respectful way, as this team was engaging with women+ individuals fleeing domestic violence and/or experiencing homelessness. Students on this project had to be prepared to deal with different levels of severity of the client's personal situation and tread carefully to avoid offensive or triggering language. The fifth Inspire project dealt with creating a platform to show climate risk zones in our local region (i.e., highlighting areas prone to flooding, landslides, and other natural threats). In this case, the potential end users for this project were anyone in our local community, and the team was tasked with being inclusive of varying levels of technological literacy.

IBM Design Thinking

Success in the Inspire program can be attributed to the software design methodologies we taught our students to employ. IBM's version of design thinking (DT) and resources on the process were provided and taught to our students. As previously mentioned, software development strategies such as Agile are very effective but often neglect the needs of the end users. This is where design thinking (DT) is particularly effective.

DT is a development methodology that involves prioritizing the feelings and wants of the intended end user *during* the requirements gathering process as opposed to treating human-centric requirements as an afterthought [7]. Typically when software is developed, the main features are implemented, and accessibility options are then added into the interface. However, in considering users' needs during requirements elicitation and implementation, there is an improved emphasis on satisfying user needs [7].

The phases of DT, also employed by the Inspire program's students, are as follows [7]:

- **Understand:** *Focus on gaining empathy for end users' pain points.*
- **Explore:** *Generate solution ideas while avoiding overly simplistic solutions.*
- **Prototype:** *Iterative series of mockups to visualize ideas from the Explore phase.*
- **Evaluate:** *Gather feedback on generated prototypes.*

DT is unique as it is a human-centered approach to software development that places a heavy emphasis on empathy. For the purposes of DT methodologies, empathy can be defined as the ability for the software development team to truly understand the needs and wants of the product's end users. Due to its human-centric focus, DT relies on a team's potential multidisciplinary talents. For example, in the Inspire program, the ability to program a website was only a small subset of the total skill required to accomplish a project. Students needed to be able to establish good communication with their community partner, negotiate project scope in a professional manner, and learn critical requirements gathering techniques such as conducting focus groups and interviews when dealing with highly sensitive topics around vulnerable persons.

Practices of Design Thinking

In this section of the chapter, we discuss several requirements gathering techniques that are unique to DT and that were used in the Inspire program. Each of these techniques introduces a new way to empathize and understand potential end users, thus giving software teams a better perspective on the needs of their clients.

Empathy Mapping

The development of effective communication skills was critical to the results of highly inclusive software that we observed. DT has several requirements elicitation techniques; a popular practice is empathy mapping. When creating an empathy map, the development team discusses pain points with a potential end user of a software product. Utterances the end user makes during such a session are then categorized into feelings, thoughts, actions, and statements. An empathy map helps the development team identify how an individual is currently navigating their pain points and helps yield creative ideas for the prototyping phase. A participant witnessing the development team categorizing their statements also helps curate a space where they feel cared for and heard, encouraging the participant to share their thoughts more openly and honestly.

Students in the Inspire program used empathy mapping frequently for their projects and were able to gain a deeper understanding into how their clients were navigating their pain points. Figure 10-1 shows the students using an empathy map to identify the common data points from the collected data. One student said empathy mapping really helped their team *“figure out what we wanted to get out of our interviews and helped us figure out how we could craft those questions in a respectful way.”*



Figure 10-1. Inspire students presenting on what they have learned about their end user's needs after empathy mapping

Hopes and Fears

Another common practice is called *hopes and fears*. In this exercise, the development team makes a list of all their greatest aspirations and worries for the designated project. When this exercise was done in the Inspire program, an overwhelming amount of both hopes and fears from students were related to the satisfaction of the end users. For example "*I hope that we can build a product that actually makes a difference for people facing homelessness.*" This demonstrates that a lot of strong emotions toward the project are centered around satisfying end user needs. Identifying a team's greatest hopes and fears for the project may be helpful in deciding which requirements elicitation methods to use and which to avoid. For example, if a software is to be designed for vulnerable groups such as individuals facing homelessness, and a team is nervous about interviews with such persons, then a greater emphasis can be placed on fine-tuning an interview structure or finding another requirements gathering technique altogether. This is critical in creating an inclusive requirements gathering experience, so that clients feel they are in a safe space to share their true needs and wants for an ultimately inclusive end product.

Journey Mapping

An important technique in DT that helps developers empathize with how end users interact with software is journey mapping. Simply put, this is a visualization of what a user experiences when they interact with the software to accomplish a goal. This timeline of steps is then embellished with things like the user's thoughts or emotions at specific moments in this journey. In the Inspire program, our students learned how overwhelming it could be for vulnerable individuals to use the software they were currently using through journey mapping. In Figure 10-2 the students can be seen using a journey map. This catalyzed discussions around simplifying user interfaces and having clearer instructions and even subtler discussions surrounding simplistic fonts and colors. Creating inclusive software is more than just creating the software; other elements such as customer communication and even marketing must also be inclusive, and journey mapping is a way to achieve this [4].



Figure 10-2. Inspire students explaining insights from journey mapping

Playbacks

Playbacks are informal presentations that are done at regular intervals to update sponsors, clients, and other project members on the progress of the project. Within the Inspire program, these playbacks were a safe space to receive feedback from other students on angles the team might have missed about their end users and from their community partners. These sessions were also conducted to help keep Inspire teams on track in terms of scope and feasibility of their prototypes. These sessions were particularly insightful as our students shared what they had learned from their clients with the Inspire teams. It was incredible to see how their assumptions about their end user needs and wants had been disproven and how the practices in DT had led them to uncover the true depth of many of these community problems.

Revisiting Previous Practices

In addition to these unique DT practices, revisiting some traditional techniques such as interviews and focus groups with a design thinking lens is just as important. In Part 2, [Chapter 6, “Elicitation Revisited for More Inclusive Requirements Engineering”], Tizard et al. acknowledge that diverse end users may not respond as well to traditional elicitation methods. With respect to DT, while traditional methods are still employed,

they take on much more forethought than the typical preparation for an interview or focus groups. Figure 10-3 shows the students at a school event to gather requirements from teens for their project.

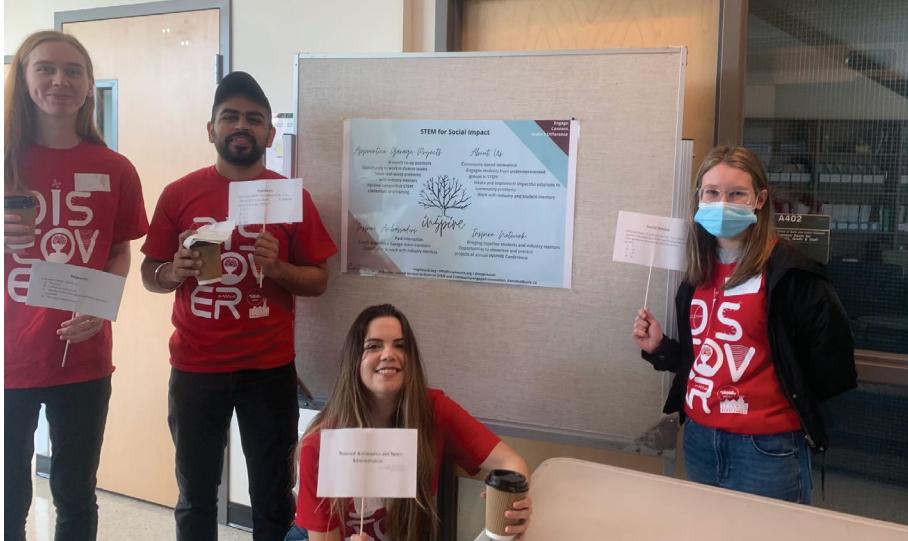


Figure 10-3. Students in the Inspire program experienced much success revisiting traditional elicitation techniques with end users in mind. The ClimAct team (pictured) was especially concerned with engaging youth and ended up gathering requirements by gamifying the interview process.

Our students in the Inspire program spent ample time focusing on the language surrounding their focus group and interview questions and received project-specific training on how to interact with their clients. For example, interviews conducted with patients with acquired brain injuries were much different than interviews conducted with neurotypical individuals. Similarly, gender-neutral language was imperative in sessions involving women+ individuals facing homelessness. Students in the Inspire program initially described having to interview vulnerable participants as “*overwhelming*” and “*intimidating*,” but found that the time spent finessing the interview dialogue and environment helped them “*understand their participants better* and “*really get our end users to open up*” to them.

Inclusive Teams, Inclusive Software

The point of inclusive software is to design a product that caters to diversity within an intended end user base. In one of our Inspire projects, for example, the website designed for patients with acquired brain injuries was not designed with the average, neurotypical person in mind. There can be an incredible amount of diversity even within a group of specific end users in mind, and creating inclusive software means acquiring an understanding of their specific characteristics and needs.

In Inspire we found that the true significance of the inclusive design thinking practices outlined previously is actually found within the software development team itself, not the end users.

While inclusive requirements elicitation is indeed critical in creating inclusive software, we discovered that a prerequisite to inclusive software is inclusion within the software development team. For example, the hopes and fears exercise catalyzed discussions within the team of each other's strongest emotions regarding the project, facilitating a sense of empathy within team members. While we recognized and focused on the diversity of our student teams, it is the inclusion within these teams that made diversity significant. In a heterogeneous group, the many advantages of diversity such as greater possibility to understand and represent the final user needs [3] and generation of more unique ideas when compared with homogeneous teams [9] cannot be leveraged without inclusion.

Our observations suggest that the empathy-based design thinking processes facilitated a sense of empathy within team members and a greater ability to understand different perspectives. In one student's words, *“...We have really developed so much empathy for each other in the group; understanding each others strengths, weaknesses, and emotions has been crucial so far.”* Such empathy for each other allowed teams to ultimately be empathetic to their end users. An Inspire student mentioned that the empathy they had for their team members *“really helped highlight how important the design thinking practices are. It even helped with creating our interview and focus group questions with an empathetic lens to make our clients more comfortable too.”* Another student mentioned, *“In using the design thinking methods we were taught in Inspire, I could see that there was more than one user group; there were actually three main categories of users based on their life stages and personal journeys. It really made me think about whether or not the platform we were building would be viable or useful for each of these user groups.”* This suggests that students were developing a heightened sense of empathy within their teams throughout the design thinking process, which they were able to then apply to requirements elicitation with their end users.

Empathy and Team Morale

The development of empathy within teams that was facilitated by DT techniques assisted the teams in empathizing with their end users and thus helped the teams create inclusive software. Creating inclusive software this way can be an arduous task; many iterations and pivots (i.e., changes in project directions) occur, and team frustration is rather inevitable. A student mentioned that their team experienced many pivots and expressed that at some points along the project they felt “*defeated*.”

Aside from the direct relationship to creating inclusive software, the sense of empathy that the students developed also dramatically helped with team morale. It was a mitigating factor when teams experienced turbulent times. The same student who described feeling “*defeated*” subsequently said, “*I would not have been able to overcome this without my team. They are very empathetic and can tell if I'm having a bad day and need to take on a lighter workload for the time being.*” Such a sense of empathy thus increased team morale overall, which also improved motivation. As the students were able to work well together in terms of team dynamic, they were able to fully focus on the parts of the project that excited them the most, thus boosting their overall motivation [8].

A Step Toward Inclusive Software

Recall that inclusive software largely encompasses an end user’s emotions [5]; thus, DT techniques such as empathy and journey mapping can contribute dramatically to making the entire software development process more user focused in order to create software that is more inclusive. The first year of the Inspire program was a stepping stone in learning how software teams approach inclusive software development. Applying the DT practices in real-world scenarios for diverse end users allowed the students to learn and empathize. The students recognized their personal biases and emphasized the users’ needs throughout the product development. While our teams superseded our initial expectations and maximized the potential of DT practices, much more work in the area of inclusive software has to be done to truly understand how we can curate an inclusive digital environment.

Conclusion and Takeaways

Overall, an analysis of performance in the Inspire program suggests that empathy-based approaches are in fact effective at instilling empathy in software development teams. Moreover, we see *why* this is so important: it allows for the creation of software that caters to a diverse set of end users. While fully introducing DT methodologies into existing software development pipelines may not be feasible for all organizations, we leave the reader with the following points of consideration:

- A sense of empathy within the software development team must be fostered before inclusive software can be successfully developed.
- Some design thinking practices can still be implemented to facilitate this sense of empathy, even if the full design thinking methodology is not adopted.
- A developed sense of empathy is critical to high team morale, especially in diverse software engineering teams.
- Creating inclusive software via these empathy-based techniques is ultimately required to repair the digital divide observed today.

Bibliography

- [1] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, January 2006. Publisher: Routledge eprint: www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa.
- [2] Sasha Costanza-Chock. *Design Justice: Community-Led Practices to Build the Worlds We Need*. The MIT Press, 2020.
- [3] John Grundy, Ingo Mueller, Anuradha Madugalla, Hourieh Khalajzadeh, Humphrey O. Obie, Jennifer McIntosh, and Tanjila Kanij. Addressing the influence of end user human aspects on software engineering. In *International Conference on Evaluation of Novel Approaches to Software Engineering*, 241–264, Springer, 2021.

- [4] Tharon Howard. Journey mapping: A brief overview. *Communication Design Quarterly Review*, 2(3):10–13, 2014.
- [5] Hourieh Khalajzadeh, Mojtaba Shahin, Humphrey O. Obie, and John Grundy. How are diverse end-user human-centric issues discussed on github? Preprint at *arXiv:2201.05927*, 2022.
- [6] Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, and Soo Fun Tan. Software development life cycle agile vs traditional approaches. In *International Conference on Information and Network Technology*, volume 37, 162–167, 2012.
- [7] Percival Lucena, Alan Braz, Adilson Chicoria, and Leonardo Tizzei. IBM design thinking software development framework. In *Brazilian Workshop on Agile Methods*, 98–109, Springer, 2017.
- [8] Michael Miles, David Melton, Michael Ridges, and Charles Harrell. The benefits of experiential learning in manufacturing education. *Journal of Engineering Technology*, 22(1):24, 2005.
- [9] Charlan J. Nemeth. Differential contributions of majority and minority influence. *Psychological Review*, 93(1):23, 1986.
- [10] John Noll, Sarah Beecham, Abdur Razzak, Bob Richardson, Ann Barcomb, and Ita Richardson. Motivation and autonomy in global software development. In *International Workshop on Global Sourcing of Information Technology and Business Processes*, 19–38, Springer, 2017.
- [11] Anthony Savidis and Constantine Stephanidis. Inclusive development: Software engineering requirements for universally accessible interactions. *Interacting with Computers*, 18(1):71–116, 2006.
- [12] Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, and Margaret Burnett. From gender biases to gender-inclusive design: An empirical investigation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–14, 2019.
- [13] Adrienne Yapo and Joseph Weiss. Ethical implications of bias in machine learning. 2018.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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.

PART III

Diversity and Inclusion in Development Teams

CHAPTER 11

Gender Diversity on Software Development Teams: A Qualitative Study

Karina Kohl, Uber, Brazil.*

Rafael Prikladnicki, Pontifical Catholic University of Rio Grande do Sul, Brazil.

This chapter presents the results of a qualitative study that aimed to understand how people in software development teams feel about gender diversity in software engineering, the perceived benefits, and the perceived difficulties. To achieve that, we applied a qualitative survey. We found out that gender-diverse workplaces are prone to have better ideas sharing, better decision-making, creativity, and innovation. Women inspire other women, and some men reported being touched by the subject and diligently are deconstructing their prejudice and misconceptions about women in technology. However, it is still common to not see women in teams or to see just a few. A white, cisgender man is the pattern most of the time. The same pattern repeats itself in leadership positions leading to male protectionism and privileges. Additionally, other dimensions of diversity pervaded the answers, like intersectionality and race/ethnicity.

The Survey

Software engineering involves real people in real environments. People create software, people maintain software, and people evolve software. Accordingly, to truly understand software engineering, it is imperative to study people – software practitioners as they solve real software engineering problems in real environments. It means conducting studies in field settings [13].

This study aims to understand how people in software development teams feel about gender diversity in software engineering. So we applied a qualitative, unsupervised, largely open-ended survey, using a web self-administered questionnaire. Our target population consists of people who identify themselves as part of a software development team worldwide.

The survey questions¹ for this work were formulated based on the Empathy Map Canvas, created by Dave Gray (2009) as a tool that helps teams develop deep, shared understanding and empathy for other people [7]. Henschel et al. [8] say empathy corresponds to the ability to understand others' minds, feel their emotions outside our own, and respond with kindness, concern, and care to their emotions.

The participation was voluntary and confidential. Using the mechanisms of the social networks, we estimated the survey link reached around 1,280 people, and 149 answered totally or partially. We evaluated that 60 respondents answered less than 20% of the questions, mostly only the demographics. We could not use these answers to extract information. In the end, we used answers from 89 respondents, corresponding to 6.88% of the people impacted by the post/link to the survey.

We had 44 respondents self-declared as man (49.44%) and 43 as woman (47.19%). Only one respondent self-declared as non-binary. One respondent self-declared as bisexual, which is a sexual orientation, not gender. Once it was self-declared and we did not have a way to identify gender, we considered it important to keep the information. About the age of respondents, on average, the respondents were 35 years old (36.09 for men, 35.15 for women). The self-declared non-binary was 31 years old and the self-declared bisexual 24 years old. The oldest respondent was 61 and the youngest 23 years old. Race and ethnicity was also a self-declared question. We had 65 respondents self-declaring white (73.86%), 5 brown (5.68%), 5 black (5.68%), and 13 diluted between Greek, Latin, Brazilian, Chinese, Swedish, Indian/Asian, and not informed (14.79%).

We also asked some professional demographics. About the role the respondent had in the software development team, we had 80 responses where 40 said they were software engineers. The other 40 answers were spread in a spectrum of scrum masters, product managers, engineering managers, DevOps, business analysts, data scientists, directors, professors, quality engineers, researchers, software architects, systems analysts, and trainees. About years in technology, we had 79 answers for this question, and, on average, respondents worked in technology for around 13 years (max, 30 years; min, 0.3 years).

¹<https://figshare.com/s/5e7720e891939da3c7a3>

To analyze the survey data and gain deeper understanding of the data content, we used *thematic analysis* [2]. Through the process, we created 429 codes for 709 segments of the answers of the 88 respondents. We grouped the codes into 18 themes, and for five of them, we split them in sub-themes, better supporting to answer our research questions. Table 11-1 presents a summary of the themes and sub-themes.

Table 11-1 Summary of themes and sub-themes

Themes	Sub-Themes	Frequency	Description
Benefits of Diversity		10	Straightforward observations about diversity in the workplace: innovation, creativity, and better decision-making, better team working, better decisions and a better product.
Diversity as a whole		36	Broader observations about diversity in the workplace: can create a more democratic workplace; try to give visibility to unconscious bias.
Companies trying to be Diverse and Inclusive		23	The importance of having Diversity as a value of the company and clear actions
Men		38	
	“Bro” culture	21	Characteristics and actions (good or bad ones) identified as coming from men and which impact gender diversity in companies
	Desconstruct Prejudice	14	
	Men Characteristics	3	
Women		125	
	Small number of women	15	
	Women are disrespected	60	Characteristics and situations (good or bad ones) identified as related to women and which impact gender diversity in companies
	Women Supporting Women	43	
	Women Characteristics	7	
No Diversity		79	
	Far away from diversity	31	
	No Equity	5	People do not identify diversity around the workplace
	All men	36	
	Old Patterns	7	

(continued)

Table 11-1. (continued)

Themes	Sub-Themes	Frequency	Description
Leadership and Management Roles		21	How women in leadership and management roles feel in terms of gender diversity
Sexism and Prejudice		26	How microaggressions impact gender diversity in teams
Professional Insecurities		53	The kind of insecurities women mainly feel related to the workplace
Companies do not support D&I		47	Companies that do not have any kind of diversity actionable initiatives
Missing Affirmative Actions and Initiatives		41	Companies that do not have enough diversity actionable initiatives
Family and Personal Life		37	How people that identify themselves as in some diversity groups feel related to their personal lives
Intersectionality		25	How different types of diversity relate to each other and impact the software engineering teams on a daily basis
Meritocracy & Elitism		4	How social/financial diversity impacts the software engineering teams on a daily basis
Unawareness		69	The insensitivity about the importance of diversity issues
Emotions and Emotional Responses	Stress	3	
	Anticipation	1	How people respond to emotions related to diversity issues
	Trust	1	
	Fear	3	
	Sadness	5	
Overload		12	How gender diversity impact the work load
Financial Concerns		40	The fears about money issues coming from diverse groups

The following sections we present the themes we defined during the thematic analysis and the perceived benefits and perceived difficulties of gender diversity on software development teams reported by individuals.

Perceived Benefits of Gender Diversity on Software Development Teams

Diversity is good beyond ethical reasons; it is recognized as valuable, and a lot of studies have been done about it [16]. Large technology companies have been increasing their efforts to have a more diverse workforce, increasing minority numbers through recruiting, working to minimize unconscious bias, and also investing in programs to increase representativeness [12].

From the **Benefits of Diversity** theme, respondents see gender-diverse environments as more prone to innovation, creativity, and better decision-making. Nonaka and Takeuchi [15] emphasize that a team made of members with different backgrounds, perspectives, and motivations is critical for organizational knowledge creation to take place. Knowledge creation is the basis of repeatable innovation in companies. *“An organization’s internal diversity must match the variety and complexity of the environment in order to deal with the challenges posed”* [20].

*That having a healthy gender mix results in **better team working, better decisions, and a better product**. A lot of people are tired of the “bro” culture around startups, and having a healthy gender mix ensures you don’t end up with a laddish/pizza and beer feel to the team.*

—#5, Man, Software Engineer

From the **Diversity as a Whole** theme, respondents believe diversity can create a more democratic workplace. They try to give visibility to unconscious bias (the social stereotypes about certain groups of people that individuals form outside their own conscious awareness [14]) and have a neutral workplace. Judy [10] says in a performing team, each member relates to the other as equals. The team will inspect its behavior and continuously improve the social skills required to communicate. It is a requirement for trust and the kind of collaboration that leads to cohesion and self-direction.

*I believe that greater diversity would make the environment **richer in ideas, dynamic, and democratic.***

—#4, Man, Software Engineer

From **Companies Trying to Be Diverse and Inclusive**, we had respondents mentioning the cultural codes of the companies as important tools to address ambiguous situations. Most codes of conduct aim to protect members from harassment (aggressive pressure or intimidation) [21]. As such, workplace harassment is any offensive, belittling, or threatening behavior toward an individual worker or group of workers. It results in an unpleasant, humiliating, or intimidating environment employees feel uncomfortable in and consequently damages the effective work and productivity of employees [21].

*Our company has a **beautiful culture code** that encourages us to be the best person every day. If someone feels bad about a situation, it should be addressed to an anonymous channel, and HR will take care of it.*

—#36, Woman, Scrum Master

The theme **Men** and the sub-theme **Deconstruct Prejudice** helped us understand that some men are not only empathetic with the subject but they are also to identify their gaps to support better work environments. Viana et al. [23] analyzed the stereotypes attributed to “egalitarian men,” men who support gender equality in relation to domestic and family responsibilities as well as inclusion in the workforce. They found out the egalitarian man is perceived as fragile, sensitive, incompetent, and feminine. On the other hand, he is also seen as more competent and social than egalitarian women and traditional men [23].

*I am always willing to learn more and more. In the end, I'm a cisgender man, and with that, everything I say or think is projected through the search for information. I'm not in my place of talking about gender diversity, so **I always try to put myself in a learning position, deconstruction of my prejudices and construction of a space for diversity.***

—#29, Man, Software Engineer

The theme **Women** and the sub-themes **Women Characteristics** and **Women Supporting Women** helped to identify some characteristics that are more prominent in women, such as empathy, flexibility, and collaboration. Turley and Bieman [22]

identified specific competencies of knowledge, personality, and attitudes as significant factors influencing software engineers' performance. Darley and Smith [6] say females pay more attention to details and disparate, multiple cues for information processing in simple and complex tasks.

Empathetic, flexible, collaborative.

—#28, Woman, Engineering Manager

Also, respondents emphasized the importance of women supporting other women. Research indicates that the social aspect of Agile practice, particularly routine face-to-face meetings and pair programming, reduces women developers' sense of isolation and raises their satisfaction and confidence [10]. It also reduces feelings of internal competition and builds trust [10]. Studies have shown that mentors can help women motivate them and improve their self-confidence to achieve their goals [24].

I always try to share and praise the women I admire and who motivated me to continue in the software development area.

—#14, Woman, Business Analyst

Perceived Difficulties of Gender Diversity on Software Development Teams

The first theme is **No Diversity** where respondents mentioned few women in the teams and no other gender than man. The pattern repeats itself in leadership positions, where there are, most of the time, white men, who are also cisgender (a person whose gender identity matches their sex assigned at birth), linking to the theme **Leadership and Management Roles** where respondents mentioned the difficulties for women to foster their careers to leadership positions.

Our findings are aligned with the previous work of other researchers. The stereotype of a computer scientist is a middle-class white man who is often geeky and antisocial [17]. While stereotypes are not necessarily representative of the general population, they do impact the perception of who belongs in the field and can act as exclusionary forces for people who do not fit the stereotype [17]. However, respondents identify characteristics in women considered leadership ones: collaboration, conciliation, determination, empathy, flexibility, organized way of working, and will to win. Jetter and

Walker [9] say, on average, women are more likely to avoid competition, underperform in competitive environments, and exhibit higher risk aversion than men. Persistent social phenomena, such as the gender wage gap or the underrepresentation of women in highly competitive occupations and job positions, have been linked to such observations. One prominent hypothesis to explain this phenomenon relates to the idea that the gender of one's opposition could influence competitive behavior. More generally, people may behave differently when competing against adversaries from the opposite sex. Women are especially underrepresented in jobs generally associated with high-pressure environments and large stakes and also technology where women usually occupy less than 20% of positions [9].

*With a lot of effort, we managed to advance some steps in terms of representation. However, **in more strategic positions, the participation of women is almost nil.***

—#25, Woman, Engineering Manager

The themes **Sexism and Prejudice, Men, and Professional Insecurities** continue the previous ones. Respondents reported male protectionism and privileges preventing women from advancing in their careers. Together, they lead to a chain of difficulties for women:

- Women suffer from **sexism and prejudice** that can be veiled and also can lead to harassment.

*Some time ago, my wife reviewed a code, and even though she was a senior, **their colleague said that he did not accept her review because she was a woman.** She complained, and the company did nothing. There was also the case that his manager called her to dinner at a one-on-one meeting.*

—#18, Man, Software Engineer

- There are two sub-themes from the theme **Men**, related to attitudes observed in men: the **Men Characteristics** and the **“Bro” Culture.** Both are related with what the respondents called the male protectionism.

*A network of **male protectionism**, which prevents women from advancing in strategic positions (or advancing with great difficulty, even requiring a certain masculinization to do so).*

—#24, Woman, Engineering Manager

- Women go through **Professional Insecurities**. They feel they need to prove they are as capable as men, and it generates the feeling of never being competent enough, and their opinions are worthless and disrespected.

I often feel that I need to go beyond my peers concerning training like I'm never competent enough.

—#27, Woman, Engineering Manager

The “*brogrammer culture*” is a term that acts as shorthand for pointing to sexism in the tech industry. The term “*brogrammer*” began as a satirical term to refer to a man who can code and succeeds with the behaviors of a stereotypical “frat boy” and ambition to become rich fast [17]. By definition, women are excluded from this group and often objectified and pushed out of workplaces because of the fraternity-like environments created as a result of *brogrammers* being in the space. Women who succeed in these environments may also face difficulties due to their gender, as being better than men can be seen as threatening. In addition, how women perceive being gendered in male-dominated spaces indicates how they relate their experience to gender. Women’s beliefs about the reason for the lack of women at high levels in companies, for example, influence whether they are motivated to fight structural barriers for other women or if they reify glass ceilings [17].

Themes **Companies Do Not Support Diversity and Inclusion** and **Missing Affirmative Actions and Initiatives** relate to one another once companies seem to have difficulty supporting diversity and inclusion and affirmative initiatives for diversity are scarce: too much talking and no action. More than that, the perception is that companies are hidden behind a facade of employers’ branding initiatives that happen only from the door out. Simmonds et al. [18] observe that affirmative action programs to boost female enrollment in programs can have positive effects for science, technology, engineering, and mathematics undergrad programs; however, the initiatives yield weaker results for computer science/software engineering majors.

Too much talk and little action, nonrecognition of women in development teams, the lack of looking at different aspects of diversity.

—#20, Woman, Software Engineer

The theme **Intersectionality** permeates diversity. Intersectionality [5] is an analytical framework for understanding how aspects of a person's social and political identities combine to create different modes of discrimination and privilege [5]. For some women, in particular women of color who are doubly and sometimes triply excluded because of their skin color and language practices, the discourse on gender may itself be harmful [4].

Intersectionality is a complex and relevant approach that is little known in the scope of the software engineering research field. Respondents brought to the light they see the number of opportunities increasing for women; however, the initiatives seem to be focused on white women. More than that, they mentioned they have few black and transgender workmates. Another face of diversity mentioned was ageism, the process of systematic stereotyping or discrimination against people because they are old, just as racism and sexism accomplish with skin color and gender [3].

In the previous company, I was the only woman on the team I was on, there was only one black man, and diversity was not encouraged; not everyone had a voice in the company.

—#16, Woman, Software Engineer

Gordón and Palacios [19] say the diversity crisis is not limited to women it is about social identities that go beyond gender and race, but it is mainly about power. There are cases in which software workers belong to two or more underrepresented groups. It is clear that systems of privilege and oppression often converge for underrepresented groups, that is, there are organizational power dynamics that have historically privileged some groups and marginalized others in the software engineering field [19]. They also say an intersectional approach invites software engineering researchers to read data in different ways and ask other questions that increasingly demonstrate the flaws of a race-only or gender-only approach.

Also, **Meritocracy and Elitism** are also understood as problems related to diversity. Meritocracy is a social system in which advancement in society is based on an individual's capabilities and merits rather than based on family, wealth, or social

background [11]. Nowadays, meritocracy is a subject with a lot of attention; however, the “*if you want you can*” speech opposes the lack of financial resources some individuals have to enter the software engineering field. The area is defined as elitist once equipment and training are usually expensive. If an individual does not have the means to be trained as a software developer, the discourse of meritocracy does not apply.

It is an elitist area. And the elite has one color and one gender. The market would easily accept black or trans people with the appropriate technical skills, but why does the market have almost no such people? Because these people are marginalized, they are on the periphery, they are without access to our world. Equipment, courses, internet, everything is expensive, so only the same group, the same elite as ever, qualify well. Companies are more open to diversity, but to what extent will this bring a truly diversified team?

—#28, Man, Software Engineer

The last theme is **Unawareness**. There were respondents who reported that diversity does not matter or should not be considered when talking about software development teams. According to these respondents, people should be treated equally, no matter their gender, and what must be considered is their ability, not their identity. Also, there were comments that the career is not for women.

I've heard several people, mostly men (professors, college friends, lecturers), saying that [diversity], it's not important, that this profession is not for women. There are still a lot of people who are resistant to this issue.

—#29, Woman, Software Engineer

Alba [1] says gender equality does not mean pretending that “male” and “female” do not exist. Gender equality also does not mean that males and females must always be treated the same. In some cases, what is required is not equal treatment but equitable treatment. Equity means recognizing that differences in ability mean that fairness often requires treating people differently so that they can achieve the same outcome [1].

Conclusion

We conducted this survey to qualitatively understand the perceived benefits and difficulties of gender diversity in software development teams. The answers we had aligned with different related work performed by researchers from the software engineering community. We also found studies from other knowledge fields, such as psychology, that studied gender issues in technology and corroborated our findings.

From a beneficial point of view, we have that gender-diverse workplaces are prone to have better ideas sharing, better decision-making, creativity, and innovation. Characteristics linked to women were highlighted as empathy, flexibility, and collaboration. To achieve the benefits mentioned, respondents reported that some companies worked to improve the hiring process to be more gender-inclusive. To support and guarantee inclusion and safety, solid cultural codes were created.

There is mutual support from women to women. Women support and inspire each other to remain in technology or enter the field. More than that, some men reported being touched by the issue and diligently deconstructing their prejudice and misconceptions about women in technology.

However, there are also difficulties from the point of view of gender diversity in software development teams. There are still cases where the respondents do not see diversity. It is common to see only one woman in teams or just a few. More than that, no other gender than men and women, so the white, cisgender man is the pattern most of the time. The same pattern repeats itself in leadership positions. This reported pattern leads to an issue: male protectionism and privileges. Due to that, women report being frequently disrespected and face difficulties in thriving in their careers. Sexism and prejudice used to happen together.

But there are other dimensions of diversity that pervaded the answers, and intersectionality was mentioned. Intersectionality aims to study how aspects of a person's social and political identities combine to create different modes of discrimination and privilege. So, when racism was mentioned, we can exemplify when respondents brought to the light they see the number of opportunities increasing for women; however, the initiatives seem to be focused on white women, excluding the intersection of race and gender of black women. Ageism, the discrimination against people because they are old, was also mentioned. Some people feel they lost, or even do not have opportunities, due to their age.

A little less explored as a dimension of diversity, social vulnerability is related to meritocracy and elitism. The "*if you want you can*" speech opposes the lack of financial

resources some individuals have to enter the software engineering field. In Brazil, equipment and training in software engineering are usually expensive, so the entry point to the field is difficult for people in social vulnerability.

We also observed a lack of awareness about the need to talk about the subject. The survey respondents reported that diversity does not matter or should not be considered when talking about software development teams. According to the respondents, people should be treated equally, no matter their gender, and what must be considered is their ability, not their identity. It would be the ideal scenario; however, due to unconscious bias that women and girls (and other dimensions of diversity) are routinely subjected to and the different other points we already mentioned in this work, we see it as important to keep an active awareness about the subject. The unconscious bias leads to people arguing that women are “biologically” and “culturally” good and trained at certain tasks such as communication, visual design, and documentation even though they do not feel interested in them.

As the main takeaways, the survey aligned with different works on the area of the importance of diversity in general, to bring more innovation and fairness to the software engineering industry. Also, the survey highlights there are still a lot of issues and prejudice, even knowing about the benefits of having more diverse work environments. More than that, research in diversity in software engineering must consider gender beyond the idea of binary genders. Additionally, intersectionality is a must. The community has been researching aspects of diversity individually. It is important that the intersection of the multiple faces of diversity be considered, considering not only gender, race, sexual identity, etc. but also social economic characteristics that put people away from software engineering positions.

Acknowledgments

Rafael Prikladnicki is partially supported by FAPERGS and CNPq in Brazil.

Bibliography

- [1] Beatrice Alba. To achieve gender equality, we must first tackle our unconscious biases. 2018.
- [2] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 77–101, July 2006.
- [3] R. Butler. Age-ism: another form of bigotry. *Gerontologist*, 243–246, December 1969.
- [4] C. Convertino. Nuancing the discourse of underrepresentation: a feminist post-structural analysis of gender inequality in computer science education in the US. *Gender and Education*, 594–607, June 2020.
- [5] K. Crenshaw. Demarginalizing the intersection of race and sex: A black feminist critique of antidiscrimination doctrine, feminist theory and antiracist politics. *University of Chicago Legal Forum*, 1–31, December 1989.
- [6] William K. Darley and Robert E. Smith. Gender differences in information processing strategies: An empirical test of the selectivity model in advertising response. *Journal of Advertising*, 41–56, April 1995.
- [7] D. Gray. Empathy map. 2017.
- [8] Sébastien Henschel, Jean-Louis Nandrino, and Karyn Doba. Emotion regulation and empathic abilities in young adults: The role of attachment styles. *Personality and Individual Differences*, 109763, April 2020.
- [9] Michael Jetter and Jay K. Walker. The gender of opponents: Explaining gender differences in performance and risk-taking? *European Economic Review*, 238–256, October 2018.
- [10] Ken H. Judy. Agile values, innovation and the shortage of women software developers. In *Proceedings of the Hawaii International Conference on System Sciences*, 5279– 5288, February 2012.

- [11] Chang Hee Kim and Yong Beom Choi. How meritocracy is defined today? Contemporary aspects of meritocracy. *Economics and Sociology*, 112–121, December 2017.
- [12] Karina Kohl and Rafael Prikladnicki. Perceptions on diversity in Brazilian agile software development teams: A survey. In *Proceedings of the IEEE/ACM International Workshop on Gender Equality in Software Engineering*, 37–40, May 2018.
- [13] Timothy C. Lethbridge, Susan Elliott Sim, and Janice Singer. *Studying Software Engineers: Data Collection Techniques for Software Field Studies*. Springer, 311–341, 2005.
- [14] Renee Navarro. Unconscious bias.
- [15] Nonaka and H. Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.
- [16] S. E. Page. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press, 2007.
- [17] Rose K. Pozos and Michelle Friend. “You sound like a good program manager”: An analysis of gender in women’s computing life histories. In *Proceedings of the ACM Technical Symposium on Computer Science Education*, 692–698, Association for Computing Machinery, New York, NY, USA, 2021.
- [18] Jocelyn Simmonds, Maria Cecilia Bastarrica, and Nancy Hitschfeld-Kahler. Impact of affirmative action on female computer science/software engineering undergraduate enrollment. *IEEE Software*, 32–37, February 2021.
- [19] Mary Sánchez-Gordón and Ricardo Colomo-Palacios. A framework for intersectional perspectives in software engineering. In *Proceedings of the IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering*, 121–122, 2021.

- [20] H. Takeuchi and I. Nonaka. *Hitotsubashi on Knowledge Management*. Wiley, 2004.
- [21] Parastou Tourani, Bram Adams, and Alexander Serebrenik. Code of conduct in open source projects. In *Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering*, 24–33, 2017.
- [22] Richard T. Turley and James M. Bieman. Competencies of exceptional and nonexceptional software engineers. *Journal of Systems and Software*, 19–38, January 1995.
- [23] Hyalle Viana, Ana Raquel Rosas Torres, and José Luis Álvaro Estramiana. Egalitarian men: stereotypes and discrimination in the labor market. *Acta Colombiana de Psicología*, 111–128, May 2020.
- [24] E. Vidal, E. Castro, S. Montoya, and K. Payihuanca. Closing the gender gap in engineering: Students role model program. In *Proceedings of the International Convention on Information, Communication and Electronic Technology*, 1493–1496, 2020.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 12

Exploring Intersectional Perspectives in Software Engineering Through Narratives

Mary Sánchez-Gordón, Østfold University College, Norway.*

Ricardo Colomo-Palacios, Technical University of Madrid, Spain.

In this chapter, we explore intersectional perspectives in software engineering by collecting ethnographic histories from three software practitioners from *underrepresented groups (URGs)*. The chapter draws attention to multiple interlocking issues related to diversity and aims to help software engineering researchers and practitioners to become more aware of conscious and unconscious discrimination and to re-story interactions with members of URGs. Without diversity, majoritarian perspectives dominate the software engineering field. This chapter begins with an introduction to intersectionality followed by our research approach, and then three disruptive majoritarian narratives are presented.

Introduction

There is an increasing interest in workforce diversity, but significant inequalities exist in the software industry [10]. It does not seem that gender equality is determinant to explain the scarce presence of women in the IT field [9]. The lack of women entering the industry is not the only problem, as evidence from the literature reveals [4, 5, 12];

there are also retention problems as well as difficulties in career development of women. However, the diversity crisis is not limited to them; it is about social identities that go beyond gender and race, for example, class, ethnicity, sexuality, nationality, age, and dis/ability [32]. Individuals with different **social identities** are affected by interlocking systems of oppression and privilege in different ways [25]. Therefore, how individuals experience their identity is **context-specific** [8], and dismantling the power structures requires complex thinking that goes beyond focusing on one dimension at a time.

Just a few software engineering studies [2, 17, 27] examine more than one of the social identities mentioned previously. Research on multiple levels of analysis is needed to understand the role that multiple and simultaneous categories of differences (social identities) play in an individual's experience and the software that they develop [5]. One way to approach that is through *intersectionality*, a social sciences lens that has been used in fields related to software engineering such as human-computer interaction (HCI) [26]. However, intersectionality is a complex approach that is little known in the scope of software engineering research and practice.

Note The origins of intersectionality can be traced back to early social and political movements of women of color, during the 1960s and 1970s. Crenshaw coined the term *intersectionality* [11] inspired by critical race theory and Black feminism. Crenshaw perceived that those approaches in legal studies were not addressing how gender and race are interwoven in shaping the experiences and struggles of women of color.

Intersectionality can inform multilevel analysis of inequality outcomes, for example, generating more precise information about recruitment, retention, and career progression. This analysis can identify how inclusionary and exclusionary practices shape practitioners' lives and at which intersections. Such evidence will reveal how to reduce intersecting power structures that hinder the achievement of equity.

Research Approach

We realized that there are numerous unheard stories within the histories and lives of URGs. Therefore, we used a qualitative research approach by interviewing three software practitioners with diverse social identities at different career stages – exploration,

establishment, and late. We also decided to select participants from a specific country (Ecuador) to limit the cultural-historical context. Using these criteria, we noted that we could recruit one software professional among our contacts. Based on our previous experience recruiting participants from URGs [24], we contacted one university professor who endorsed our study to encourage participation and recruited one further participant. Then, we systematically searched Google to find software professionals who met our criteria. As a result, we sent invitations to four potential participants, and we got a response rate of 100%. Although all were interested in participating, we canceled one interview due to scheduling conflicts. Therefore, we recruited three participants.

After obtaining informed consent for each participant, one researcher conducted, recorded, and transcribed semi-structured interviews that lasted from two to two-and-a-half hours. During interviews, we asked participants to reflect on their decisions, aspirations, fears, and hopes, because we explored what they did and what had shaped their decisions, work trajectories, and imagined futures. We use pseudonymized participant identifiers by using the anonymous “FM,” “RS,” and “CN.” To identify intersectional perspectives, data were analyzed using the combination of several elements of our proposed framework for intersectionality. In turn, to make sense of participants’ experiences naturally, three storied narratives were written by taking their words as an illustration of their social identity salience. To establish resonance in our findings, we shared a private copy of the narratives with the participants to conduct member-checking. If the participants thought something was misinterpreted in their narratives, they were able to reword, comment, or suggest edits. However, only minor changes were introduced.

Intersectional Perspectives

The growing diversity crisis of software engineering [1, 3] and our self-reflection as individuals and members of this community motivated the design of a conceptual framework for understanding intersectional perspectives in software engineering that was previously published in [25]. In this chapter, we used it as a perspective view because we assume that we cannot give a complete view of an individual. Figure 12-1 shows the high-level view, which consists of three major levels: social identities, societal processes and organizational practices, and cultural-historical context.

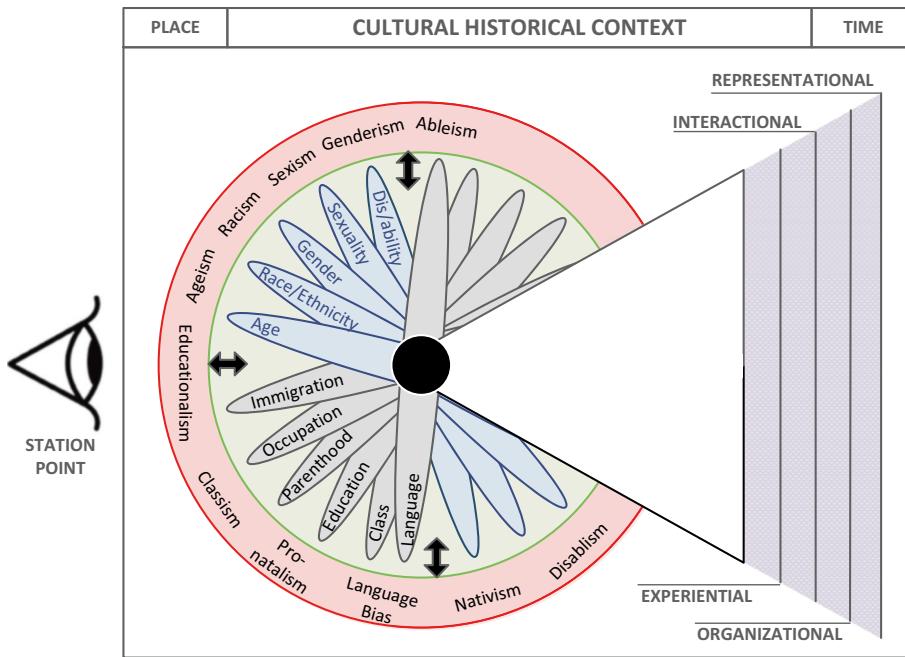


Figure 12-1. Framework for intersectional perspectives in SE adapted from [25]

The **individual** is illustrated in the black inner circle in the center of Figure 12-1. The first level represents the **social identities** (green middle circle) that an individual might hold. By reviewing the software engineering literature, the following two groups of social identities were identified: (1) **internal (personal) characteristics** (blue ellipses), which included but were not limited to age, gender, sexuality, race/ethnicity, and dis/ability, and (2) **external characteristics** (gray ellipses), which included but not limited to occupation, education, immigration, language, social class, parenthood, and religion. These multiple social identities might overlap and combine, denoting certain intersections at the micro level of individual experience so that they not only shape an individual point of view on the world but also they reflect **interlocking systems of oppression** (red outer circle) at the macro, social-structural level, for example, genderism, sexism, and racism. This first level also considers that **identity development** is also strongly influenced by recognition in a role or community [13]. Therefore, it emphasizes how **membership** (participation) in multiple social categories influences the degree to which individuals face impediments to progress in organizational contexts.

The first level allows us to understand that software professionals from URGs experience social structure differently since the junction of their identities (see Table 12-1) reflects an intersection of overlapping oppressions that take place in the

software industry. For example, FM as a cis-man parenting a child with a disability, who is influenced by recognition in a role or group since FM is a project manager that holds a PMP Certification. Before starting the analysis, we recognized our position in the world (for details, see the following for **positionality**) to establish a station point since it changes our perspective and determines to what extent we will be able to generate a multiple-point perspective by navigating the second level of the framework.

The second level is concerned with **societal processes and organizational practices** that influence the formation, salience, perpetuation, and character of social categories. Although it is hard to distinguish the boundaries between the domains of institutional power, we can, at least, consider four types. **Representational** is the degree to which diverse groups and related policies are portrayed in materials depicting the profession (e.g., media about the discipline). Moreover, it is concerned with how stereotypes threatening URGs are produced and sustained over time as their self-consciousness about failure increases. **Organizational** refers to the institutional processes that hinder diverse groups from participating (e.g., structural positions in society such as family, work, and education). **Interactional** focuses on the nature of the interactions between the different social actors (e.g., relationships between members of groups and individuals). It also links how those relationships influence life chances and outcomes that can reify or minimize stereotype threats. **Experiential** is how individuals' sense-making of their vital experiences relates to their perception of their social identity when shaping their opportunities.

Figure 12-1 illustrates the second level as a four-point perspective to the left (purple area). From our positionality, we focus on one area, for example, "organizational," while aiming to generate a one-point perspective. For instance, RS mentioned that "I tried to get a job by sending resumes to see if any company would give me the opportunity ... I did a couple of interviews for ones, and I did some technical tests for others, but they did not call me" (organizational). Although we are standing in the same spot, we can shift our focus from this area to another to generate a two-point perspective, for example, RS also reflected "companies should open their doors to [newcomers] gain experience" (experiential). As deep reflection is required to integrate the different points of view gradually, we used the process of thinking aloud for deeper reflection and understanding of the transcriptions of the interviews we drew upon. To illustrate another example, CN was also afraid that her lack of experience impeded her job search and claimed, "I hope they [potential employers] hire me to have more experience ... In most jobs [ads] ask for 5 years of experience" (experiential), and she sadly recalled, "The remuneration is very little, and there are no job positions for what I have studied" (organizational).

The third level represents the **cultural-historical context** in which the first and second levels are situated. It means that multiple identities are socially constituted and have an impact on how social positions, hierarchies, and divisions are established and reified in society. Figure 12-1 depicts this level as the view plane that situates the participants and their social context in a particular place and time.

As mentioned before, we limited the analysis of this level to one country. However, although the participants did not identify as belonging to an URG and they have a sense of belonging comparable to majoritarian groups as found by [19], our analysis reveals that systems of oppression and privilege are in place and hinder/empower success. In this sense, an unexpected finding was that RS believed that his lack of experience was an issue, but his indigenous identity was not. Indigenous software workers have frequently reported ethnic discrimination [24], but RS's salient identity was "English language," which gives him privilege over more experienced professionals such as FM, who is not proficient in English.

Narratives

The software industry has never been equitable and has arguably served as a social reproduction mechanism, maintaining and perpetuating social inequity [Chapter 1, "Roads Ahead to Diversity and Inclusion by Software Engineering"]. Despite the efforts and perseverance of many underrepresented people in the software industry, their accomplishments have been historically neglected, for example, the hidden stories of Atari women [7]. We believe that an intersectional perspective can help challenge dominant stories that make one narrative the only story. Therefore, we seek diverse perspectives to help break down the power of dominant narratives and stereotypes [22]. Based on the findings from the interviews, we used a type of counter-narrative, which is a third-person authored story of another person's life, to reveal experiences with and responses to interlocking systems of oppression [28]. In this way, we aimed to legitimate their experiences as "ways of knowing."

We found that beyond sexism and genderism, other forms of oppression or "-isms" such as racism, classism, ableism, abuse of power, or dehumanization take place in the software industry. Throughout this book, it is also supported, for example, [Chapter 1, "Roads Ahead to Diversity and Inclusion by Software Engineering", Chapter 3, "The Challenges of Ethnic-Racial Diversity Within the IT Sector" Chapter 24, "Economical Accommodations for Neurodivergent Students in Software Engineering Education: Experiences from an Intervention in Four Undergraduate Courses"].

Whose Story Is It?

Table 12-1 shows the participants' social identities, but we identify nuances like parenting a child with a disability, indigenous belonging to an Andean community, or living with a physical disability because of being born without arms.

Table 12-1. Participants' social identities

Identities	Family Man (FM)	Rising Star (RS)	Courageous Newcomer (CN)
Age	Middle age	Young	Young
Gender	Cis-gender man	Cis-gender man	Cis-gender woman
Nationality	Ecuadorian	Ecuadorian	Ecuadorian
Race/ethnicity	Mestizo	Andean Indigenous	Mestizo
Disability	-	-	Physical
Language	Spanish	Spanish, English, Kichwa	Spanish, English
Education	Engineer	Engineer	Student
Class	Middle	Working	Working
Parenthood	Yes	-	-
Occupation	Developer	Developer	-
Career stage	Mid-career	Establishment	Exploration
Immigration	-	Yes	-

We also disclose our **positionality**. The first author identifies as Latinx and a cis-gender woman who has low vision. The second author identifies as Spaniard and a cis-gender man. Both of us have been parents and immigrants. We are also middle-aged and native Spanish speakers with PhD degrees in computer sciences. Together, we present around 50 years of experience in academia and the software industry.

Disrupting Dominant Narratives

A deficit perspective perpetuates the damaging narrative that the software industry is based on meritocracy and therefore is equitable and appropriate for all professionals. The following three narratives provide a viewpoint that is not represented by dominant narratives.

A Family Man's Story

He likes big challenges without fear of failure. He is passionate about developing software and has gained expertise and recognition in financial software development. Indeed, he thought “the financial sector can be stressful, but it pays well.” During his career path, he has played different roles like developer, tester, and technical leader, including movements to senior positions like a project manager and project management office (PMO) manager. He was part of international software projects and traveled to other South American countries. He has built skills and knowledge via courses, formal training, and on-the-job learning. He is a family man who is satisfied and proud of his career.

He thought that his journey was governed by his choices, and after 25 years, he rarely changed jobs. However, three years ago, he quit without notice from the company that he helped grow for 14 years. Surprisingly for him, it was not a hard decision; he wanted to spend more time with his son and pursued a PMP Certification. He had not taken vacations for four years, and he sometimes left the office at 2:00 a.m. and returned at 9:00 a.m. He had enjoyed working long days and always committed to working as hard as he could on weekdays. When he got promoted to senior positions, it was easier to avoid working weekends or holidays and spending them with his family.

He had acted for over 20 years like he was strong, even when he was not. Some days parenting was his hardest job. His son had a late autism diagnosis; he and his wife were always busy with therapies and appointments. They chose to see it as a gift that opened their eyes to new wonders and taught them to be better people. He learned to celebrate the victories no matter how small they were. He not only developed organization, adaptability, and dependability but also empathy, compassion, resilience, and other transferable social skills.

After a longer break, he realized that obtaining a PMP Certification would be easier than finding a full-time job that demands, at least, 40 hours per week. Finally, he accepted a job with a lower salary for which he was overqualified, but suddenly

everything changed when the COVID-19 pandemic started. Remote work was a relief as previously he would spend more than two hours in traffic to get to the office. Now, he shares parenting duties and home chores with his wife. Although he accepts that there is no “perfect” work-life balance, he aspires to get a more challenging and better-paid job in a flexible workplace.

A Rising Star’s Story

He knows what lack of opportunities means because his parents immigrated to Europe in the 1990s. They were undocumented and itinerant Andean Indigenous for almost ten years. The removal of barriers to intra-European mobility in 1992 considerably eased that, and they made a living. He was born and raised far from Ecuador until he was around five years old. At that time, the economic support from a European family allowed his parents to come back and give him a private education in a town surrounded by indigenous villages. A public university near his home was the best option to ease his family’s economic hardship.

He was involved in academic clubs during his university years, and he became part of the Microsoft student developer community. It allowed him to meet other students and visit other universities. Once he carried out a capstone project, he was actively looking for a job, but potential employers did not call him back or said that he did not fulfill the required years of experience. He began to get frustrated because of the lack of opportunities. Then, he sought the advice of an industry professional he met a few years ago. This professional supported and encouraged him by offering advice and knowledge. He gained new perspectives on his career and started training in a set of programming tools while he was developing his undergraduate thesis. Both started to seek a job for him as a junior software developer.

When the COVID-19 pandemic started, a rise in software developer demand substantially impacted hiring methods and business goals. Companies explored international talent pools – hiring remotely. Remote jobs opened opportunities for talented developers without leaving home. After an interview and solving a coding challenge, he started a trial period in a software development outsourcing company located in Central America that has a large presence in North America. Surprisingly for him, the company did not require any certification or degree. However, he had worked hard and learned a lot at the beginning. In fact, his full-time job sometimes requires extra time for acquiring the knowledge that each project demands.

This company has local and international projects, has onboarding and mentoring programs, and organizes virtual bonding events to keep the team connected and engaged. There is also a diversity initiative, but he is not part of it, since it focuses on women and does not consider other underrepresented communities (like male immigrants). After two years, his technical and soft skills have grown. He is a rising star who made the most of his skills. His English language skills allow him to be part of international projects. It seems that hiring software developers in Latin America is significantly more cost-efficient than in the United States. The reality of his employment exceeded his initial expectations, and he is working for his career advancement.

A Courageous Newcomer's Story

She is a very independent young woman with a physical disability whose dream was to study at a university. She knew that studying at a higher education institution would be a new challenge for her. She worked hard and enrolled in a bachelor's in computer science. Although it took her longer, her experience as a student in a public university was like that of any other. Indeed, she has never seen herself as different from others despite her disability.

She always thinks about her mother who put her through school for 12 years and tells herself to "never give up." Initially, she was refused due to her disability, but her mother was persistent and found a Catholic private school where an endowment was established to support her. As she was born without arms, she learned to do everything with her feet – she could do it "just like anyone else." However, classist attitudes from her peers dissuaded her from finishing high school there. This is a clear example of why equality of opportunity is not enough. Indeed, she found relief in the public education that previously rejected her. In the beginning, she was timid and had little contact with the other students, but her disability was not an obstacle to enjoyable studies during those years. She learned to use a keyboard and a mouse to program, and her interest in computing grew.

The hard road to her education began in university when the domino effect of failing a course delayed her graduation for three years. She had not even heard about the "Student Welfare Service" during the first four years. After those years, she was tired but still determined. However, she could not overcome mental exhaustion. There were many deeply ingrained barriers, both subtle and obvious, that she had to face. The long way home and lack of sleep created an imbalance in her body. Her student life was stressful

during the next two years, but she refused to give up. The COVID-19 outbreak disrupted her education in such a way that brought her time to rest and flourish. The online classes started four months later, and she was only enrolled in a couple of courses at a time. Her mother never treated her any differently from her three younger siblings. So she has never wanted to be treated differently in any way and thrived, thanks to her spirit. In a bid for future success, she is currently seeking a job while finishing her graduation project, which is about weed detection using machine learning. Although she is putting time and effort into finding a job opportunity like any other newcomer, she is a little disappointed because professional experience is required and there is no room for her preferences – at least so far. Her passion still burns, and she expects to have more job opportunities than other unqualified disabled people.

Discussion

The perspectives of software developers from URGs have not yet been thoroughly explored in many previous empirical studies of software engineering as mentioned [Chapter 29, “Strategies for Reporting and Centering Marginalized Developer Experiences”]. In this chapter, the proposed conceptual framework provides a prism through which to view how different types of inequality interact and exacerbate one another. There are many ways to use it, but turning to the preceding narratives, they illustrate the first level of the framework by introducing the participants’ social identities and material differences in conditions of life.

FM mentioned, “In 2019, I decided to quit. I just quitted. He [my son] always had a personal tutor … I didn’t see the need for [to be there], and my wife has always been with him.” He also regretted, saying, “I had to have negotiated time and money” and recalled, “I talked to my wife, and she said, ‘Our son is the most important thing. I will support you in the decision you make.’ I hadn’t taken a vacation for a while, four or five years.” FM also reflected on disabled dependent care by saying, “It creates [further] expenses” and claimed, “It is complex in the sense that nobody can understand what is happening,” but “He [his son] is not a burden … I think I’ve learned to understand life in a different way.”

CN smiled and said, “I was born without arms, so I do everything with my feet.” Then, she added, “She [her mom] has never felt sorry for me” and recalled her mother’s voice saying, “Come help me … Come do this. You can do it.” CN also stated, “At high school, I learned to disassemble computers and programming. That’s when I realized that it is hard, but I liked programming … That’s why I decided to follow the engineering

career." CN was also reminded of her university years. "I went by bus to the university. It lasted around 70 minutes, and it took me about 30 minutes to get to the bus stop." CN also claimed, "My body gets tired, and there comes a time when it doesn't give anymore" and added, "The truth is hard, [there is] a lot of pressure, many times without eating." CN reflected, "I was so exhausted ... I was mentally exhausted" and claimed about the COVID-19 pandemic, "I had already been through so much [at the university] that I wanted to be home." CN displayed her determination saying, "I have always thought that my arms are missing, not my head ..."

RS stated that "I was part of the clubs [at the university] and I developed skills [there]" and claimed, "It was an enriching experience to me, but it was not enough to have my first job." RS remembered a European family that helped his parents to pay for a private education by saying, "Foreign aid was influential during my education at primary and secondary schools." Then, RS also mentioned, "My parents worked hard to give me an education ... I had a computer from an early age." RS also pointed out, "I had a good level of English when I enrolled at the university." RS recalled the months when he was looking for a job and said, "I felt frustration. I knew it was not because of ethnicity but because of experience."

The narratives also present some societal processes and organizational practices that affect the formation, salience, perpetuation, and character of the social categories. The idea of privilege is inherent in the three stories, but it operates differently in the background to create the context in which the stories developed. The main character in each story represents each participant in this study. They are subject to various inequalities, and their individual experiences are not simply the sum of their parts.

The FM story unveils the dominant story about a male-dominant software industry that welcomes men. In this story, FM is a man who experienced a toxic culture and burnout. He claimed, "I used to work on the weekend, when necessary, but I am no longer willing to work late. I do not know how to explain this, but I want to spend time with my son." That might sound odd in this type of culture, but FM's son is a young adult with disabilities and high support needs. In this case, the personal story of FM shows that he is dealing with work-life balance issues comparable to those reported by female ICT practitioners regarding motherhood [23, 31]. It suggests that further research is needed to better understand the perceived impact of parenthood on software engineering since an individual's social identities profoundly influence one's experience. Thus, this inequality cannot be separated from other forms of inequality such as age. FM is also a middle-aged professional who followed two employability strategies mentioned in [6]: moving to a management role and specialization. Despite that, he found it hard

to find a new job, just as any other “older” developer would in this competitive industry. The lived experiences of FM also reveal organizational practices “glorifying working late or on weekends” [6].

The RS story challenges the majoritarian story on stereotypes and perceptions about indigenous people as unmotivated, low-qualified workers and academically untalented [24]. In fact, a recent study [15] about societal biases shows that applicants from URGs would be perceived as less favorable and capable. This study examined if racial and gender biases may be reduced through the embodiment of video game characters. The preliminary results show that an intersectional effect of race and gender simultaneously influences an applicant’s rating.

The analyses of the RS story revealed that compared with the narratives of other indigenous male and female software professionals with racism [24], RS recounted having no experiences with racism. One possible interpretation of this finding is that racism is not an issue for indigenous professionals in his region; however, this view is superficial at best. Therefore, a more compelling interpretation goes beyond the individual narrative. RS also said that he lives “near an indigenous community,” and RS was a student at a regional university with a population of about 10% indigenous students. In addition, RS mentioned that “my English teacher at the primary school was a native speaker. He was a retired foreigner ... As you know, foreigners like to live here.” In this case, background education shaped RS’s experiences of intersecting identities of racial or ethnic identification. Thus, the interpretation of the “seemingly unmeasurable and un-analyzable data” is an essential component for applying an intersectionality approach as mentioned by [8].

Additionally, the RS story highlights the critical role of mentoring in his career. Although the theoretical framework of Intersectional Capital [29] focuses on women, it is worth noting that mentoring is one of the three tenets central to that framework. The RS story suggests that further research is needed to understand the extent to which mentoring has a particular impact on professionals from URGs. RS also pointed out that “I did not have the years of experience that employers demand when I started looking for work after graduation.” It also suggests that “companies should take their share of responsibility, encouraging practices that welcome developers of any age” as recommended by [6].

The CN story draws attention to the silence prevailing around conversations about disability and the necessity of ensuring the inclusion and participation of students with disabilities in higher education institutions. The lack of policy and provision for students

with disabilities negatively impacts their self-efficacy. In line with this, a previous study [20] revealed that policies like competitive enrollment were a negative predictor of a sense of belonging and self-efficacy. However, this study is not focused on disability.

Additionally, the CN story illustrates that the effect of class and status is beneficial for some students, but they are barriers for others of lower socioeconomic status, such as CN, to pursue computer careers [14]. It highlights a need for educational strategies that engage undergraduate women in computing across their multiple identities like the theoretical framework of Intersectional Capital [29] or the critical framework for analysis in CS education called Intersectional Computing [21]. However, it is worth noting that those frameworks do not consider disability; therefore, further research is needed to address disabled students. Moreover, although previous studies like [18] were designed to counter-act oversimplification of the complex issues affecting workforce diversity, their focus is on the intersection of gender, race, and ethnic group but fails to include disability. Therefore, disability should be understood in the context of power relations embedded in social identities too. In particular, [18] aims to understand women's career progression and identify six themes of which four were mentioned by CN – bias, credibility and legitimacy, support, and technical skills. For instance, CN stated that "People, just based on my disability, assign a bit of a value or assumptions about my [technical] skills and scope of responsibility. There's all this worry about whether you're capable or qualified." In addition, CN noted that employers are crucial at this early stage of her career and said that "I would like to have only one job opportunity to prove who I am." CN also emphasized the importance of technical skills to gain the respect of her peers and professors.

Despite all difficulties, CN still chose to stay in the computer discipline as the 11 Black women interviewed in [30]. Although CN did not mention having effective mentors like these Black women, she, like these women, remains true to her personal and professional goals and the inspiration from her mother, for example, she shared that "I am the first person in my family to attend university." The analysis of societal processes allows us to identify that these factors contributed to her strategies of resistance and "successful" journey.

These stories remind us that being on the advantaged side provides more than just avoidance of oppression or disadvantage since it also provides access to status, rewards, and opportunities that other intersections do not have. Moreover, an intersectional position may be disadvantaged in one group while advantaged in another. RS, as an indigenous man, may be disadvantaged because of racism, but he enjoys privilege over other indigenous and non-indigenous professionals due to his English language proficiency.

From an intersectional perspective [16], it is a problem if we consider the inequalities presented in these stories or other inequalities as a “them” or “unfortunate other” problem since it hinders our ability to understand and conduct a proper intersectional analysis. Our experience has been that even having this disposition, it is challenging and complex to incorporate an intersectional perspective because it is related to our mindset.

Summary

In this chapter, we explored intersectional perspectives through narratives from three software practitioners at different career stages. As people are as important as the process and technology in software engineering, the intersectional experiences of professionals from URGs must be legitimate as “ways of knowing” that offer a perspective not covered by dominant narratives [21]. To make sense of these complex and uncertain situations in a natural way, we used stories. Particularly, we used counter-narrative to challenge majoritarian narratives by communicating the lived realities of professionals from URGs.

To promote broadened participation from URGs, there is a critical need to understand their narratives, and interventions must carefully consider power dynamics between dominant and non-dominant social groups as also mentioned by Kohl and Prikladnicki in Chapter 11, “Gender Diversity on Software Development Teams: A Qualitative Study.” We hope this chapter inspires other researchers and practitioners to explore the soft side of software engineering and challenge the deeply engrained oppression systems that are in place by taking seriously the unique and specific knowledge of the software professionals from URGs.

Acknowledgments

We would like to thank the software professionals who participated in this study and shared their lived experiences. Moreover, we acknowledge The European Commission and ERASMUS+ Programme for partially supporting this research work as part of the project EduTech (609785-EPP-1-2019-1-ES-EPPKA2-CBHE-JP). The European Commission’s support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Bibliography

- [1] Bram Adams and Foutse Khomh. The Diversity Crisis of Software Engineering for Artificial Intelligence. *IEEE Software* 37, 5 (September 2020), 104–108. DOI: <https://doi.org/10.1109/MS.2020.2975075>
- [2] Swati Agarwal, Nitish Mittal, and Ashish Sureka. Minority ethnic groups in computer science research: what is the bibliography data telling us? *SIGCAS Comput. Soc.* 47, 2 (June 2017), 5–15. DOI: <https://doi.org/10.1145/3112644.3112646>
- [3] Khaled Albusays, Pernille Bjorn, Laura Dabbish, Denae Ford, Emerson Murphy-Hill, Alexander Serebrenik, and Margaret-Anne Storey. The Diversity Crisis in Software Development. *IEEE Software* 38, 2 (March 2021), 19–25. DOI: <https://doi.org/10.1109/MS.2020.3045817>
- [4] Catherine Ashcraft, Brad McLain, and Elizabeth Eger. 2016. *Women in Tech: The Facts*. National Center for Women and Information Technology, Boulder, CO.
- [5] William Aspray. 2016. *Women and Underrepresented Minorities in Computing: A Historical and Social Study*. Springer International Publishing. DOI: <https://doi.org/10.1007/978-3-319-24811-0>
- [6] Sebastian Baltes, George Park, and Alexander Serebrenik. Is 40 the New 60? How Popular Media Portrays the Employability of Older Software Developers. *IEEE Software* 37, 6 (November 2020), 26–31. DOI: <https://doi.org/10.1109/MS.2020.3014178>
- [7] Pernille Bjørn and Daniela K. Rosner. Intertextual design: the hidden stories of Atari women. *Human–Computer Interaction* 37, 4 (July 2022), 370–395. DOI: <https://doi.org/10.1080/07370024.2020.1861947>
- [8] Lisa Bowleg. When Black + Lesbian + Woman ≠ Black Lesbian Woman: The Methodological Challenges of Qualitative and Quantitative Intersectionality Research. *Sex Roles* 59, 5 (September 2008), 312–325. DOI: <https://doi.org/10.1007/s11199-008-9400-z>

- [9] Ricardo Colomo-Palacios and Cristina Casado-Lumbreras. 2019. National cultures and gender balance in ICT: a preliminary study. In *Proceedings of the 13th European Conference on Software Architecture – Volume 2* (ECSA '19), Association for Computing Machinery, Paris, France, 76–81. DOI: <https://doi.org/10.1145/3344948.3344965>
- [10] Committee on Information Technology, Automation, and the U.S. Workforce. 2017. *Information Technology and the U.S. Workforce: Where Are We and Where Do We Go from Here?* National Academies Press, Washington, D.C. DOI: <https://doi.org/10.17226/24649>
- [11] Kimberle Crenshaw. Demarginalizing the Intersection of Race and Sex: A Black Feminist Critique of Antidiscrimination Doctrine, Feminist Theory and Antiracist Politics. *University of Chicago Legal Forum* 1, 8 (1989), 31.
- [12] Edna Dias Canedo, Heloise Acco Tives, Madianita Bogo Marioti, Fabiano Fagundes, and José Antonio Siqueira de Cerqueira. Barriers Faced by Women in Software Development Projects. *Information* 10, 10 (October 2019), 309. DOI: <https://doi.org/10.3390/info10100309>
- [13] Allison Godwin, Geoff Potvin, Zahra Hazari, and Robynne Lock. 2013. Understanding engineering identity through structural equation modeling. In *2013 IEEE Frontiers in Education Conference (FIE)*, 50–56. DOI: <https://doi.org/10.1109/FIE.2013.6684787>
- [14] Jane Margolis, Rachel Estrella, Joanna Goode, Jennifer Jellison Holme, and Kimberly Nao. 2017. *Stuck in the Shallow End: Education, Race, and Computing*. MIT Press.
- [15] Marie Jarrell, Reza Ghaiomy Anaraky, Bart Knijnenburg, and Erin Ash. 2021. Using Intersectional Representation & Embodied Identification in Standard Video Game Play to Reduce Societal Biases. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (CHI '21), Association for Computing Machinery, New York, NY, USA, 1–18. DOI: <https://doi.org/10.1145/3411764.3445161>

- [16] Katy Steinmetz. 2020. She Coined the Term “Intersectionality” Over 30 Years Ago. Here’s What It Means to Her Today. *Time*. Retrieved April 29, 2023, from <https://time.com/5786710/kimberle-crenshaw-intersectionality/>
- [17] Antonio M. Lopez, Kun Zhang, and Frederick G. Lopez. 2008. Gender and race: Stereotyping, coping self-efficacy and collective self-esteem in the CSET undergraduate pipeline. In *2008 38th Annual Frontiers in Education Conference*, F4B-20-F4B-25. DOI: <https://doi.org/10.1109/FIE.2008.4720372>
- [18] Kimberly McGee. The influence of gender, and race/ethnicity on advancement in information technology (IT). *Information and Organization* 28, 1 (March 2018), 1–36. DOI: <https://doi.org/10.1016/j.infoandorg.2017.12.001>
- [19] Catherine Mooney and Brett A. Becker. 2020. Sense of Belonging: The Intersectionality of Self-Identified Minority Status and Gender in Undergraduate Computer Science Students. In *United Kingdom & Ireland Computing Education Research conference.*, ACM, Glasgow, United Kingdom, 24–30. DOI: <https://doi.org/10.1145/3416465.3416476>
- [20] An Nguyen and Colleen M. Lewis. 2020. Competitive Enrollment Policies in Computing Departments Negatively Predict First-Year Students’ Sense of Belonging, Self-Efficacy, and Perception of Department. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ACM, Portland, OR, USA, 685– 691. DOI: <https://doi.org/10.1145/3328778.3366805>
- [21] Yolanda A. Rankin and Jakita O. Thomas. 2020. The Intersectional Experiences of Black Women in Computing. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (SIGCSE ’20), Association for Computing Machinery, New York, NY, USA, 199–205. DOI: <https://doi.org/10.1145/3328778.3366873>
- [22] Yolanda A. Rankin, Jakita O. Thomas, and Sheena Erete. Real talk: saturated sites of violence in CS education. *ACM Inroads* 12, 2 (May 2021), 30–37. DOI: <https://doi.org/10.1145/3463406>

- [23] Larissa Rocha, Edna Dias Canedo, Claudia Pinto Pereira, Carla Bezerra, and Fabiana Freitas Mendes. May 14–15. Investigating the Perceived Impact of Maternity on Software Engineering: A Women’s Perspective. In *ICSEW’23: Proceedings of the IEEE/ACM 45th International Conference on Software Engineering Workshops*, Melbourne, Australia, 12.
- [24] Mary Sánchez-Gordón and Ricardo Colomo-Palacios. 2020. Factors Influencing Software Engineering Career Choice of Andean Indigenous. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings* (ICSE ’20), ACM, New York, NY, USA, 264–265. DOI: <https://doi.org/10.1145/3377812.3390899>
- [25] Mary Sánchez-Gordón and Ricardo Colomo-Palacios. 2021. A Framework for Intersectional Perspectives in Software Engineering. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, IEEE, Madrid, Spain, 121–122. DOI: <https://doi.org/10.1109/CHASE52884.2021.00025>
- [26] Ari Schlesinger, W. Keith Edwards, and Rebecca E. Grinter. 2017. Intersectional HCI: Engaging Identity Through Gender, Race, and Class. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, Denver, Colorado, USA, 5412–5427. DOI: <https://doi.org/10.1145/3025453.3025766>
- [27] Uta Schloegel, Sebastian Stegmann, Rolf van Dick, and Alexander Maedche. Age stereotypes in distributed software development: The impact of culture on age-related performance expectations. *Information and Software Technology* 97, (May 2018), 146–162. DOI: <https://doi.org/10.1016/j.infsof.2018.01.009>
- [28] Daniel G. Solórzano and Tara J. Yosso. Critical Race Methodology: Counter-Storytelling as an Analytical Framework for Education Research. *Qualitative Inquiry* 8, 1 (February 2002), 23–44. DOI: <https://doi.org/10.1177/107780040200800103>

- [29] Breauna Spencer, Audrey Rorrer, Sloan Davis, Sepi Hejazi Moghadam, and Cori Grainger. 2021. The Role of “Intersectional Capital” in Undergraduate Women’s Engagement in Research-Focused Computing Workshops. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, 1–6. DOI: <https://doi.org/10.1109/RESPECT51740.2021.9620576>
- [30] Jakita O. Thomas, Nicole Joseph, Arian Williams, Chan’tel Crum, and Jamika Burge. 2018. Speaking Truth to Power: Exploring the Intersectional Experiences of Black Women in Computing. In *2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, 1–8. DOI: <https://doi.org/10.1109/RESPECT.2018.8491718>
- [31] Bianca Trinkenreich, Ricardo Britto, Marco A. Gerosa, and Igor Steinmacher. 2022. An empirical investigation on the challenges faced by women in the software industry: a case study. In *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society (ICSESEIS ’22)*, Association for Computing Machinery, New York, NY, USA, 24–35. DOI: <https://doi.org/10.1145/3510458.3513018>
- [32] Yanwei Zhu. 2019. “Recoding Gender”: The intersectional experience of female immigrant programmers in the Swedish IT sector. Master’s thesis. Lund University, Sweden. Retrieved from <http://lup.lub.lu.se/student-papers/record/8993439>



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 13

Perceptions of Software Developer Inclusion: A Survey at Google

Maggie Hedges, Google, USA.*

Emerson Murphy-Hill, Google, USA.

As an emerging and growing research field, the ability of researchers in software developer diversity and inclusion to make practical progress hinges on their ability to ask and answer the right questions. To make progress on this task, we surveyed a diverse group of 903 software engineers at Google about their experiences surrounding inclusion and software development. We found that accessibility was frequently mentioned as an important area to address, that negative experiences sometimes impact certain groups more often, and that open and closed source developers use different patterns of help-seeking behavior.

Introduction

As evidenced by this book, a variety of researchers have studied inclusion in software engineering communities, typically through the lens of a specific group of historically marginalized developers such as women [14] or through a specific task such as code review [13]. Such research can provide compelling insights into the inordinate challenges imposed on these developers.

What such lenses lack is a bigger picture on inclusion in software engineering, beyond challenges faced by a specific community or during a specific task. But this lack of a big picture has posed a significant threat to our research group's ability to fulfill our

team's mission at Google, to *advance understanding of diversity and inclusion challenges facing software developers and evaluate interventions that move the needle on creating an inclusive developer culture for all.* Without a big picture of inclusion challenges, both our team and the wider community that researches software developer inclusion cannot be certain that our current focal areas are the most important inclusion areas we can be focusing on. For instance, perhaps our effort to study inequities in code review is misguided, because we could have a bigger impact by studying a topic we haven't thought of yet that is pervasive in practice.

To widen the lens and build a bigger picture of inclusion in software engineering, in 2021, we ran a survey with 903 responses from Google engineers (18% response rate). We constructed the survey by combining newly created questions with questions adapted from Stack Overflow¹ and GitHub² developer surveys to allow for comparisons. Survey topics included experiences with key collaborative development tasks; preferred channels for obtaining support off-team (i.e., a team at Google that a developer is not a member of); help-seeking, help provision, and stuck behavior; and experiences while using asynchronous communication tools. Full survey details are in the section "Appendix."

To select participants, we used a stratified sampling approach to ensure the representation of developers across diverse races/ethnicities, gender identities, and ages, as detailed in the section "Appendix." We additionally invited developer members of LGBTQ+ (29 responses), transgender (8 responses), and disability-focused (4 responses) employee resource groups to respond through their email lists. Because there were relatively few respondents from the transgender and disability groups, we did not provide or analyze breakdowns for those groups, but we did include their responses within our aggregate quantitative and qualitative analysis. In figures throughout this chapter, response breakdowns from members of the LGBTQ+ employee resource group are referred to by the group's name, "Pride at Google."

The other demographic identifiers referenced in this chapter are reflective of the options that were available within employees' internal human resources profiles at the time of data collection (which also include the option to not self-identify). In charts we present, we distinguish between US- and non-US-based employees when referring to race/ethnicity categories because analyzing race/ethnicity data is only permitted for US employees according to company policies based on international regulations. Response

¹<https://insights.stackoverflow.com/survey/2020>

²<https://opensourcesurvey.org>

counts per group are detailed in the section “Appendix” alongside the sampling criteria. Note that most categories we provide breakdowns for are not mutually exclusive, that is, survey participants may appear in multiple demographic categories, such as those for age, race/ethnicity, and gender. The only mutually exclusive categories are Female/Male and US/Non-US. Our project was reviewed by Google’s employee privacy working group, helping ensure that we were using data congruent with employees’ expectations and following privacy best practices.

We summarized overall distributions of responses across questions with descriptive statistics and analyzed data for any significant differences in responses across groups when sample sizes were sufficient to provide statistical power to make such comparisons. The first author analyzed open-ended data to synthesize qualitative themes. We next summarize the results and discuss what those results mean.

Results

Overall Inclusion Sentiment

Overall, we found that engineering tools and processes are meeting most groups’ needs. Thirty-one percent of respondents reported that engineering processes and tools met all of the respondents’ needs, 60% reported most of their needs were met, 8% reported some of their needs were met, and 2% said few of their needs were met. Figure 13-1 breaks down participant responses by demographic category.

Thinking about your background, experiences and demographic characteristics, to what extent do engineering tooling and processes at Google meet:

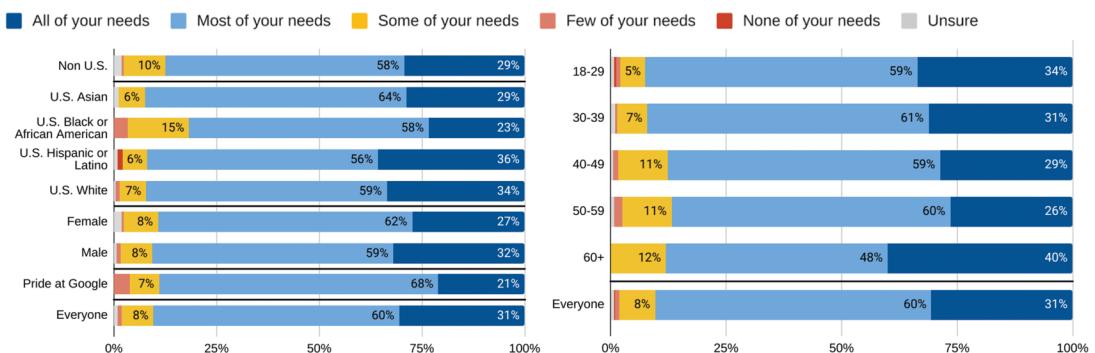


Figure 13-1. Survey response breakdown for software engineering inclusion question

On the other hand, looking at Figure 13-1, more Black developers (18%) reported that only some or few of their needs were met when compared with White, Asian, and Hispanic or Latino developers (8%). Additionally, more developers aged 40 years and older (13%) reported that only some or few of their needs were met than those under 40 years of age (7%). Also, developer members of Pride at Google were the least likely to report all of their needs were met when compared with everyone else (21% vs. 31%).

Most Important Developer Inclusion Areas to Address

To add more depth to the previous quantitative data, we next asked respondents the following open-ended question, “For this question, we’re defining inclusion as the degree to which employees feel part of essential organizational processes, including influence over the decision-making process, involvement in critical work groups, and access to information and resources. In your opinion, what inclusion issues in engineering tools and processes do you view as most important to address, and why?” We coded the 177 responses into several categories, as shown in Figure 13-2. We next give examples of comments from the top four categories and the “better documentation, tutorials, mentorship” category.

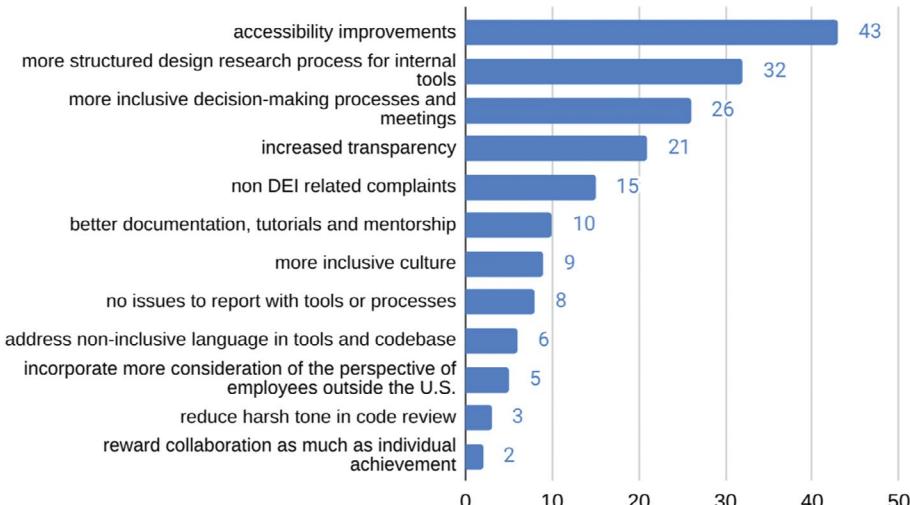


Figure 13-2. Number of open-ended responses in emergent themes from the inclusion issues question

The most common response was a desire for improved accessibility of internal tools. More than half of those comments specifically discussed enhanced supports for visual accessibility, for example:

[C]olor contrast. Not all text is legible. My eyesight is good but not perfect anymore.

Participants also desired improved product design processes. Specifically, the second most common theme was a desire for a more user-centered design process for internal engineering tools, similar to the structured design research approaches applied to consumer products. Respondents wanted the design of these tools to take in account the varied experiences and feedback of their users rather than basing decisions on the intuitions of the team building the product or the loudest voices off-team:

The typical metric that [internal] product teams seem to use for prioritizing features is “How many people are asking for this?” It often feels incredibly difficult to motivate their product decisions based on “How acutely is this affecting a smaller group of users?”

Respondents raised concerns that organizational decision-making was at times too hierarchical and opaque, failing to incorporate a diverse set of perspectives or to clarify the criteria used to reach decisions to the broader impacted groups. This category included concerns around when conversations are not facilitated to solicit input from the range of voices present:

Engineering decisions have become much more top-down than in the past ...and the needs and concerns of the engineers on the receiving end are not really considered.

Respondents also had a desire for increased transparency and open information-sharing. Respondents shared examples of information being difficult to discover and inconveniences related to gaining access to documents in cases where the subject matter was not confidential yet default sharing permissions weren't granted between different parts of the organization:

Inability to access documents in others' verticals might affect ability to provide feedback.

Finally, respondents described how high-quality documentation, user guides, and mentorship can ensure all developers are equally empowered to be successful:

A lot of documentation on engineering tools assumes background or past experiences that may not be applicable to folks from diverse backgrounds ... Certain documentation puts a lot of burden on the reader to dig around for background info.

Experiences During Collaborative Development Work

We next asked respondents to share their experiences with key collaborative development activities. The breakdown of their responses is shown in Figure 13-3. Most respondents reported frequently or always having positive experiences during these activities. About 80% said they always received respectful feedback during design and code reviews, while over half said they always experienced fair task distribution, appropriate credit, and receptiveness to their ideas in the context of their work.

In terms of collaboration, developers of higher age groups reported worse outcomes than those of lower age groups, to a statistically significant degree. Twenty-three percent of developers aged 60 years or greater reported having experienced unfair distribution of development tasks compared with only 6% of developers under 30 years old. Nineteen percent of developers between the ages of 50 and 59 years old and 35% of developers 60 years or greater in age reported instances of not receiving appropriate credit for their engineering contributions, compared with 6% of developers under 30 years old. Apart from these age differences, no other statistically significant differences between demographic groups emerged.

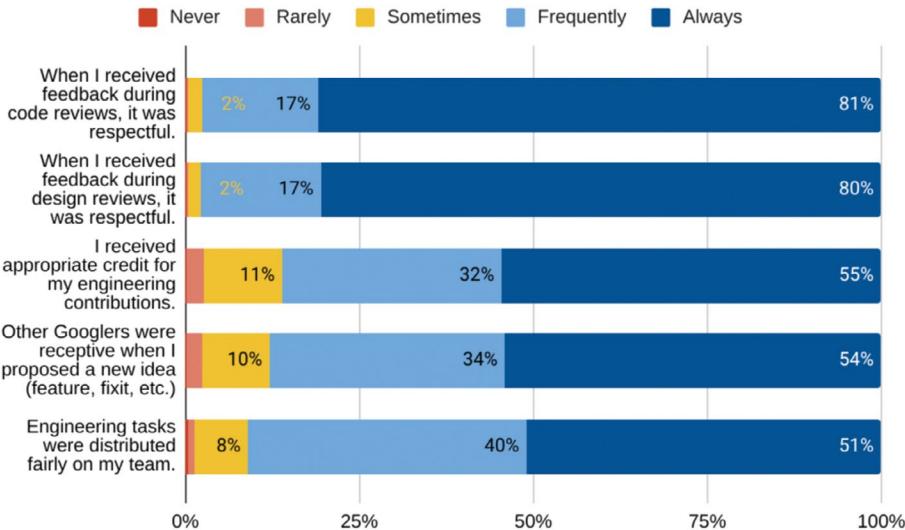


Figure 13-3. Survey response breakdown for experiences during collaborative development activities question

Respondents provided examples of negative experiences during these collaborative development activities, as a follow-up to the prior question. In the following are some of the examples of the 110 follow-up responses we received:

There is a lot of behind-the-scenes work that gets done (like task tracking, note-taking, organizational planning, mentoring, etc.) that I as a female was either asked to do or just did that helped the team run smoothly but was never acknowledged.

Even if I contributed heavily to the design through reviews, if my name isn't listed as an author, I probably wouldn't take credit. Might be nice to have guidelines about second [and] third authors and publicize more.

I was offended when my coworker responded that my idea was "not important" and that I was "missing the company's priorities." Even if that is 100% true, I would have appreciated him saying, "That's an interesting idea, but we don't have time for that right now. Maybe we can revisit this later."

Work is not particularly proactively distributed or tracked on my team. Nobody has particular visibility into how much work anyone is doing...and at [performance review] it's clear our manager has limited visibility too.

Level of Comfort Reaching Out Off-Team

Most participants reported feeling comfortable using various channels when reaching out for off-team support, but comfort did vary based on the channel. More respondents reporting being “comfortable” or “very comfortable” reaching out to an individual they were directly referred to or to listed code, document, or bug owners (88% and 84%, respectively) than they did using an internal question forum called YAQS or reaching out to a mailing list (77% and 70%, respectively).

When asked to elaborate on reasons for discomfort, the most frequently cited reasons were fear of asking what others would perceive as an obvious or unnecessary question, doubts that the answers would be timely enough or sufficient to solve their problem, reticence to engage due to the tone in which previous questions were responded to within the forum, and concerns about spamming too wide of a group or disrupting others in their work. For example, one developer explained in an open response:

I have major imposter syndrome and I'm afraid of asking dumb questions. Questions on YAQS and mailing lists will be visible forever, and it's open to a huge audience.

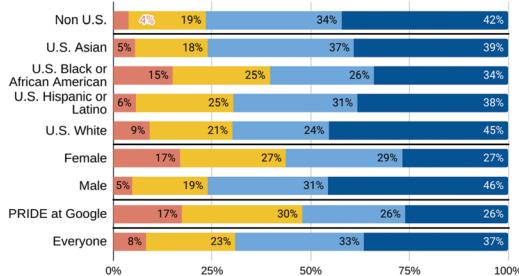
As suggested by Figure 13-1, how different demographic groups responded to questions was generally similar. However, the level of comfort seeking off-team technical support was one of the two areas of the survey where differences did exist, as evidenced by adjusted Wald confidence intervals.

Figure 13-4 shows these differences across groups. Using non-overlapping confidence intervals of non-favorable responses as the criterion, we observe the following differences:

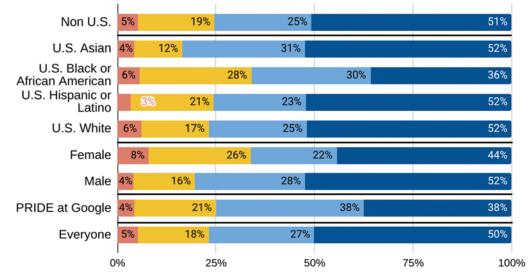
In the past 3 months, how comfortable did you feel reaching out off-team via the following channels?

■ Not at all comfortable ■ Somewhat comfortable ■ Comfortable ■ Very comfortable

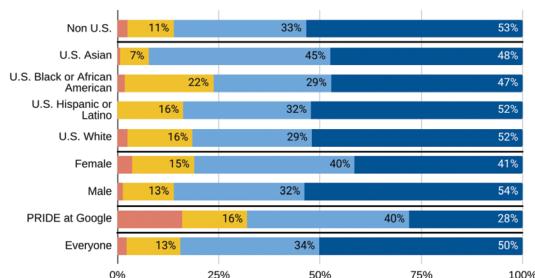
Mailing List



YAQS



Listed code/doc/bug owners



Directly to another individual I'm referred to

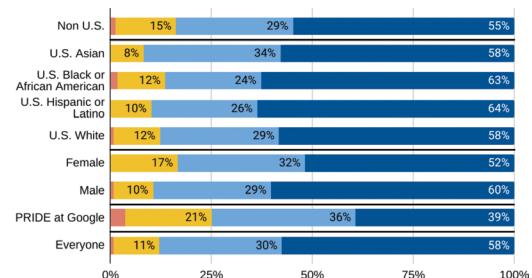


Figure 13-4. Comfort reaching out off-team, broken down by demographics

- Thirty-four percent of female developers reported being “not at all” or “somewhat” comfortable reaching out via YAQS compared with 20% of male developers.
- Forty-four percent of female developers reported being “not at all” or “somewhat” comfortable reaching out via a mailing list compared with 24% of male developers.
- Thirty-four percent of Black developers reported being “not at all” or “somewhat” comfortable using YAQS compared to 16% of Asian developers.
- More Black (23%), White (18%), and female (18%) developers, as well as developer members of Pride at Google (30%), reported being “not at all” or “somewhat” comfortable reaching out to listed code/doc/bug owners when compared with Asian developers (8%).

Comparing Google to Stack Overflow and GitHub

Because some of our questions were modified versions of inclusion questions asked by Stack Overflow and GitHub, we can compare those results. When asked about a recent experience finding help, directly reaching out to another person for help is the most common way developers reported providing and seeking assistance at Google; 63% of Googlers reported asking a specific person for help, while only 14% of GitHub users did. Google developers were less likely to report unsolicited help or asking for help in an external forum; 0% of Googlers reported doing this, whereas 74% of GitHub users did.

Methods for getting un-stuck also differed between Google and Stack Overflow.

When asked about what developers do when they get stuck

- Eighty-eight percent of Google respondents “ask a teammate for help” at least sometimes, while only 50% of Stack Overflow respondents “call a coworker or friend” with the same frequency.
- Seventy-one percent of Google respondents “investigate the issue using external documentation (Stack Overflow and others)” at least sometimes, while 91% of Stack Overflow respondents “visit Stack Overflow” with the same frequency.
- Twenty-three percent of Google respondents “watch help/tutorial videos” at least sometimes, while 53% of Stack Overflow users do.

Many of the differences observed between Google and GitHub responses can likely be attributed to the inherent differences in developer workflows in a private company vs. an open source project. For example, within a private company, developers primarily collaborate with colleagues they know, whereas within an open source context, developers are more likely to collaborate with strangers whom they don’t have a professional relationship with.

Discussion

Returning to our original question, are we as researchers studying the right thing? There are many ways to answer this question, but turning to Figure 13-2, we can compare what the research community focuses on vs. what developers report as the most important to address.

For instance, many in the research community – ourselves included – have focused on inclusion in the code review process (e.g., [7, 9, 10, 11, 13]), but it was not mentioned by many respondents as the most important issue to address. On the other hand, vision and other accessibility issues were frequently mentioned, providing some support for research in the topic (e.g., [1, 2, 3, 8, 12]). As another example, a more structured design research process for internal tools was the next most mentioned inclusion issue, but we know of little research in that area. To the extent that there's a disconnect between practitioners' perceptions of inclusion issues and what the research community investigates, we speculate that part of the problem may be that the community tends to investigate highly structured and instrumented processes (like code reviews), which are easier to study than unstructured or un-instrumented processes (like design reviews).

Beyond asking respondents directly what they believe to be the most important inclusion issues to address, we also asked them about their experiences during common collaborative development activities and evaluated whether experiences varied across demographic groups. We found some groups were less comfortable reaching out for help using mailing lists and question forums. Prior research has also found evidence that experiences in public technical forums vary by gender (e.g. [4, 5, 6]). Scaling the exchange of information and expertise often requires diverting questions to common support channels. Further research aimed at enhancing the psychological safety of these online knowledge-sharing spaces could potentially reduce barriers to obtaining technical support that may be disproportionately experienced by underrepresented groups within those spaces.

We believe strongly in amplifying marginalized voices, as we have tried to do with this survey. But at the same time, we also believe that a research roadmap need not be constructed strictly using a histogram of frequently mentioned issues. Solving real inclusion issues is clearly critical, yet not every issue is salient and easily articulated when filling out a survey, so researchers should also apply other prioritization mechanisms. For instance, a researcher in a historically marginalized group may choose to study that group, bringing valuable personal motivation and experiences to the table.

Additionally, although we found few significant differences in terms of the frequency of positive experiences across demographic groups, we did find that a small proportion of engineers across all groups had negative experiences while collaborating on development work. While frequency of negative occurrences may be similar, the impact of those negative experiences may not be equal across all groups. For those who are less represented in the field of engineering, these experiences, like failing to

get appropriate credit for their contributions or receiving disrespectful feedback, may have greater negative impact. Thus, research aimed at reducing the incidence of these unfavorable experiences in general, through more fair and transparent processes for distributing and tracking team work, for example, may still serve to enhance inclusion for marginalized groups.

We believe that Google is a reflection of the larger tech industry and that our survey is replicable in other organizations. At the same time, our results have very limited generalizability because our sample of survey respondents was stratified such that the respondent pool is purposefully *not* representative of the population of developers at Google. Given the differences we observed comparing our results to GitHub and Stack Overflow's, results may be different in other development contexts as well. Nonetheless, the results presented in this chapter provide a unique examination of developer inclusion issues across a diverse sample of engineers.

Conclusion

As the field advances, researchers can use our findings to help guide decisions on what inclusion problems to study in software development. When asked to volunteer priorities to address in this space, developers frequently mentioned issues related to accessibility and product design processes, but since frequency isn't the same as importance, we also encourage researchers to study other inclusion challenges, challenges that might not yet be salient to practitioners, but that negatively impact software developer diversity and inclusion nonetheless.

Acknowledgments

Thanks to Carolyn Egelman, Ciera Jaspan, Claire Taylor, Collin Green, Dalain Williams, Jill Dicker, Jingjing Chen, Liz Kammer, Madison Stamos, Sarah D'Angelo, Sarah Inman, Google's Core Developer, and anonymous reviewers for their feedback and support.

Appendix

In this appendix, we list our survey design, sampling criteria, and survey questions.

Survey Design

Our survey was divided into three parts: the frontmatter, high-level inclusion experience questions, and questions about specific processes. In what follows, we briefly describe the design of each section; full questions appear later in this appendix. We asked six questions designed to assess developer sentiment toward engineering inclusion. The questions covered whether the company’s engineering tooling and processes meet engineers’ own needs, as well as to what extent the respondent believes the company is committed to product inclusion and accessibility when developing Google’s internal engineering tools. We designed these questions to assess software developer inclusion at a high level, with a focus on developer tools because our team’s place in the company enables it to provide research findings directly to teams that build and maintain such tools.

The remainder of the questions focused on specific development processes and tasks. The first two questions were about the nature of feedback during code and design reviews, whether engineering tasks are distributed fairly, peers’ receptiveness to new ideas, and receiving appropriate credit for engineering contributions. We chose these questions based on prior work in computer and social sciences, which, for instance, suggest that women are more likely to face more pushback during code review [9, 13].

The next three questions asked about respondents’ comfort reaching out off-team via various channels and frequency of employing various methods for getting unstuck. These questions are based on a similarly worded, comparable question for Stack Overflow. The next four questions asked about a respondent’s most recent specific help-giving and help-receiving experience, digging into how help was obtained, the relationship between the help giver and help receiver, and the nature of the problem. These questions are based on similarly worded questions from GitHub. The final three questions asked about discouraging behaviors that respondents may have encountered, such as lack of responses to questions, dismissive responses, and unexplained delays. These questions were also adapted from GitHub’s survey.

Sampling

To ensure inclusion of diverse groups and perspectives in our sample, we invited

- 300 developers who self-identify as Black or African American
- 400 developers who self-identify as Hispanic or Latino
- 500 developers who self-identify as Asian
- 500 developers who self-identify as White
- 10 developers who self-identify as American Indian or Alaskan Native
- 10 developers who self-identify as Native Hawaiian or Other Pacific Islander
- 500 developers who self-identify as male
- 500 developers who self-identify as female
- 500 developers who self-identify as less than 29 years of age
- 500 developers who self-identify as between 30 and 39 years of age
- 500 developers who self-identify as between 40 and 49 years of age
- 400 developers who self-identify as between 50 and 59 years of age
- 70 developers who self-identify as 60+ years of age

While we would have ideally sampled 500 software engineers from each group, for several groups listed here, we invited fewer engineers, so as not to overburden these communities with research requests.

We received 903 total responses, but 41 of these were in response to survey invitations sent by email to three employee resource groups and do not have demographic data connected to them. We received 862 responses from the stratified sample that we invited by individual email invites. Demographic data is available for these 862 responses, but we only report breakdowns in this chapter for the groups that had sufficient sample sizes. We received responses from

- 171 developers based outside the United States (race/ethnicity data not available)
- 197 developers who self-identify as Asian

- 64 developers who self-identify as Black or African American
- 105 developers who self-identify as Hispanic or Latino
- 306 developers who self-identify as White
- 2 developers who self-identify as Native Hawaiian or Other Pacific Islander
- 1 developer who self-identifies as American Indian or Alaska Native
- 12 developers who self-identify as more than one of the preceding races/ethnicities
- 28 developers within the United States who chose not to self-identify a race/ethnicity

Breaking down responses by gender, we received responses from:

- 178 developers who self-identify as female
- 681 developers who self-identify as male
- 3 developers who chose not to self-identify a gender

Across age groups, we received responses from

- 252 developers under the age of 30
- 283 developers aged 30–39 years
- 175 developers aged 40–49 years
- 127 developers aged 50–59 years
- 25 developers aged 60 years or more

Responses from some groups (e.g., Male) exceed the original strata sample sizes because respondents from other samples contribute to response group sizes.

Survey Frontmatter

We designed the frontmatter to the survey to explain the context of the survey, how the results would be used and shared, and how to reach out to the researchers. At the end of the frontmatter, we asked two questions about where respondents' code resides: in the company's monolithic repository (e.g., where Google Maps and Docs reside), in its

external git repository (e.g., where Android and Chromium reside), on private cloud git repositories (source.cloud.google.com), or on public git repositories on GitHub. These questions were not generalizable outside of Google, so we do not provide any further analysis of them here. More than 80% of respondents said that Google's monolithic codebase is where the majority of their code is hosted.

Refinement and Analysis

To refine the survey's validity and clarity, we piloted the survey by observing 11 developers filling it out. Pilot participants were recruited using a small stratified sample similar to the one used for the full survey. During the pilot, the first author observed the participants answer the survey questions over video conferencing during a 45-minute session. The first author occasionally probed with additional questions about how pilot participants selected their responses or how they interpreted the question and answer options.

Survey Questions

[Q1] Thinking about your background, experiences, and demographic characteristics, to what extent do engineering tooling and processes at Google meet?

- All of your needs
- Most of your needs
- Some of your needs
- Few of your needs
- None of your needs
- Unsure

[Q2] Google/Alphabet is committed to product inclusion within internal engineering tools (i.e., intentionally incorporating underrepresented perspectives at key points in the product design process).

- Strongly disagree
- Disagree
- Neither agree nor disagree

- Agree
- Strongly agree
- Unsure

[Q3] Optional: Please explain your answer to the previous question and elaborate on any ways Google/Alphabet is or is not demonstrating commitment to product inclusion within internal engineering tools.

[Q4] Google/Alphabet incorporates accessibility (i.e., the needs of people with disabilities) into the design and development process of internal engineering tools.

- Strongly disagree
- Disagree
- Neither agree nor disagree
- Agree
- Strongly agree
- Unsure

[Q5] Please complete the following statement. The engineering tools I use regularly within my work at Google/Alphabet have

- All of the accessibility supports I require
- Most of the accessibility supports I require
- Some of the accessibility supports I require
- Few of the accessibility supports I require
- None of the accessibility supports I require

[Q6] Optional: For this question, we're defining inclusion as the degree to which employees feel part of essential processes, including influence over the decision-making process, involvement in critical work groups, and access to information and resources. In your opinion, which inclusion issue in engineering tools and processes at Google do you see as most important to address, and why?

[Q7] During the past three months, how often did each of the following things happen? (Options: Never, Rarely, Sometimes, Frequently, Always, Not applicable)

- When I received feedback during design reviews, it was respectful.
- When I received feedback during code reviews, it was respectful.
- Engineering tasks were distributed fairly on my team.
- Other Googlers were receptive when I proposed a new idea (feature, fixit, etc.).
- I received appropriate credit for my engineering contributions.

[Q8] Optional: Please elaborate on any experiences in which you did not receive respectful feedback during design/code reviews, engineering tasks weren't distributed fairly, other Googlers weren't receptive when you proposed an idea related to your engineering work, or you did not receive appropriate credit for your engineering contributions.

[Q9] During the past three months, how comfortable did you feel reaching out for off-team help through the following channels? (Options: Not at all comfortable, Somewhat comfortable, Comfortable, Very comfortable, Not applicable)

- YAQS (an internal Q&A system)
- Mailing list
- Listed code/doc/bug owners
- Directly contacting another individual I was referred to

[Q10] Optional: Please describe any reasons you did not feel comfortable when using YAQS, mailing lists, listed owners, or referrals to reach out for assistance off-team.

[Q11] During the past three months, how frequently did you do each of the following when you got stuck during an engineering task? (Options: Never, Rarely, Sometimes, Frequently, Always)

- Investigate the issue using internal documentation (YAQS, MoMA, intranet search, etc.).
- Investigate the issue using external documentation (Stack Overflow and others).
- Do other work and come back later.

- Watch help/tutorial videos.
- Ask a teammate for help.
- Ask a colleague off-team for help.
- Do something non-work related and come back later (e.g., take a walk).
- Other:

[Q12] Thinking of the most recent case where someone helped you at work, how did you find someone to help you? (Choose one.)

- I asked for help in an internal forum (e.g., in a YAQS thread, project mailing list, etc.) and someone responded.
- I asked for help in an external forum (e.g., Stack Overflow) and someone responded.
- I asked a specific person for help.
- Someone offered me unsolicited help.
- A standard or prescribed process led us to interact (e.g., code review).
- Other:

[Q13] Which best describes your relationship with the person who helped you?

- We knew each other well.
- We knew each other a little.
- I knew of them through their contributions to projects, but I didn't know them personally.
- Total strangers, I didn't know of them previously.

[Q14] What kind of problem did they help you with?

- Writing code or otherwise implementing ideas
- Debugging code
- Installing or using a tool, application, or piece of infrastructure
- Understanding community norms (e.g., how to submit a change, how to communicate effectively)

- Introductions to other people
- Other:

[Q15] Thinking of the most recent case where you helped someone at work, how did you come to help this person?

- They asked for help in an internal forum (e.g., in a YAQS thread, project mailing list, etc.) and I responded.
- They asked me directly for help.
- I reached out to them to offer unsolicited help.
- A standard or prescribed process led us to interact (e.g., code review).
- Other:

[Q16] Which best describes your relationship with the person you helped?

- We knew each other well.
- We knew each other a little.
- I knew of them through their contributions to projects, but I didn't know them personally.
- Total strangers, I didn't know of them previously.

[Q17] What kind of problem did you help them with?

- Writing code or otherwise implementing ideas
- Debugging code
- Installing or using a tool, application, or piece of infrastructure
- Understanding community norms (e.g., how to submit a change, how to communicate effectively)
- Introductions to other people
- Other:

[Q18] Optional: Please share any additional context about your answers to the previous questions regarding recent instances of helping others and receiving help or your general experience as an engineer of helping and receiving help.

[Q19] Thinking about the code review tools you've used in the past three months at Google/Alphabet, how often have you experienced the following? (Options: Never, Rarely, Sometimes, Frequently, Always)

- Unexplained delay in getting a CL reviewed
- Lack of response to questions
- Withholding of LGTM on CLs without explanation
- Dismissive responses to CLs
- Dismissive responses to questions
- Conflict or interpersonal tension with another engineer
- Language or other content that made you feel uncomfortable (e.g., profanity, inappropriate jokes, etc.)

[Q20] Optional: Please elaborate on any of the preceding experiences that you said happened rarely to always while using code review tools.

[Q21] Thinking about the asynchronous communication tools you've used in the past three months at Google/Alphabet (e.g., chat, email, YAQS), how often have you experienced the following? (Options: Never, Rarely, Sometimes, Frequently, Always)

- Unexplained delay in getting a response
- Lack of response to questions
- Dismissive responses to questions
- Conflict or interpersonal tension with another engineer
- Language or other content that made you feel uncomfortable (e.g., profanity, inappropriate jokes, etc.)

[Q22] Optional: Please elaborate on any of the preceding experiences that you said happened rarely to always while using asynchronous communication tools.

Bibliography

- [1] Khaled Albusays and Stephanie Ludi. Eliciting programming challenges faced by developers with visual impairments: exploratory study. In *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering*, 82–85, 2016.
- [2] Ameer Armaly, Paige Rodeghero, and Collin McMillan. AudioHighlight: Code skimming for blind programmers. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 206–216, IEEE, 2018.
- [3] Catherine M. Baker, Lauren R. Milne, and Richard E. Ladner. StructJumper: A tool to help blind programmers navigate and understand the structure of code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 3043–3052, 2015.
- [4] S. J. Brooke. Trouble in programmers paradise: gender-biases in sharing and recognising technical knowledge on stack overflow. In *Information, Communication and Society*, 2092–2112, 2021.
- [5] Sian Brooke. Condescending, rude, assholes: Framing gender and hostility on stack overflow. In *Proceedings of the Third Workshop on Abusive Language Online*, 172–180, 2019.
- [6] Denae Ford, Alice Harkins, and Chris Parnin. Someone like me: How does peer parity influence participation of women on stack overflow? In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, 2017.
- [7] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Garden, Emerson Murphy-Hill, and Kelly Blincoe. Destructive criticism in software code review impacts inclusion. In *Computer Supported Cooperative Work*, 2022.

- [8] Sean Mealin and Emerson Murphy-Hill. An exploratory study of blind software developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 71–74, IEEE, 2012.
- [9] Emerson Murphy-Hill, Ciera Jaspan, Carolyn Egelman, and Lan Cheng. The pushback effects of race, ethnicity, gender, and age in code review. *Communications of the ACM*, 65(3):52–57, 2022.
- [10] Reza Nadri, Gema Rodriguezperez, and Meiyappan Nagappan. On the relationship between the developer’s perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering*, 2021.
- [11] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 57–60, 2020.
- [12] Kevin M. Storer, Harini Sampath, and M. Alice Merrick. It’s just everything outside of the IDE that’s the problem: Information seeking by software developers with visual impairments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–12, 2021.
- [13] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*, 3:e111, 2017.
- [14] Bianca Trinkenreich, Ricardo Britto, Marco A. Gerosa, and Igor Steinmacher. An empirical investigation on the challenges faced by women in the software industry: A case study. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 24–35, IEEE, 2022.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 14

How Much Do Women Build Open Source Infrastructure?

*Huilian Sophie Qiu**, Northwestern University, USA.

Zihe H Zhao, Rice University, USA.

Tielin Katy Yu, Carnegie Mellon University, USA.

Laura Dabbish, Carnegie Mellon University, USA.

Bogdan Vasilescu, Carnegie Mellon University, USA.

There have been many past reports of women being underrepresented among contributors to open source, from surveys and analyses of repository data. In this chapter we take a fresh, comprehensive look at the representation of women in open source, focusing on historical trends among *infrastructure* projects – the libraries and packages indexed by popular package managers that so much of the world relies on. We start by compiling and synthesizing existing empirical data from the literature and then use an automatic name-based gender inference technique to capture population level across 20 open source package manager ecosystems. Our results reveal a promising upward trend in the percentage of women among both highly active (“core”) and general repository contributors over time, but also high variation in the percentage of women contributors across ecosystems. The chapter is based on a short paper we presented at ICSE SEIS 2023 [44].

Introduction

The economic value and importance of open source software (OSS) to the economy and society as a whole are, by now, well recognized. Companies big and small, nonprofits, government entities, scientists, students, and hobbyists all use OSS libraries and packages [18]. To maintain all this digital infrastructure, a constant supply of effort is needed, often by volunteers, to fix bugs, patch vulnerabilities, and implement new features. Prior research has repeatedly shown that the availability of this effort should not be taken for granted – open source contributors can choose to disengage at any time for a variety of reasons [37], and even widely used, popular projects can end up being maintained by no one at all [4, 12].

Among the challenges to open source software sustainability, low gender diversity is particularly problematic because it hinders the benefits that a team could have possessed otherwise. It is beyond being a problem of social justice, as there is plenty of evidence demonstrating the benefits of having a gender-diverse team. For example, evidence shows that having a gender-diverse team in public code collaboration could enhance productivity and lower community smells [11, 46]. One reason behind the better performance is that men and women tend to display different personalities [49]. Leveraging positive personality traits that are associated with better team performance can lead to more successful teams [59]. At the same time, a diverse team can better understand the needs of their users, which are often diverse [38].

Practitioners and researchers have been working on solving the problem of low gender diversity in OSS. Many studies and reports in the past two decades showed low representation of women in OSS (see the section “Related Work” for a review). Active research areas include identifying roles women play in OSS development [54], detecting barriers that women face when entering OSS [17], and quantifying biases women face when making contributions [53]. In practice, there are initiatives to remove barriers for women and to create more inclusive communities, such as Open Source Diversity,¹ Outreachy,² and Rails Girls Summer of Code.³

There have been many attempts to assess the gender representation in the open source software community. Although prior studies reached a general agreement on the overall low fraction of women in the population, the reported percentages have a

¹<https://opensourcediversity.org>

²<https://www.outreachy.org>

³<https://railsgirlssummerofcode.org>

high variance, the possible causes of which could be unrepresentative samples, different subpopulations, different time periods, or different methods. In this chapter, we add one large-scale study to the literature while fixing the method and looking over time and across ecosystems. This chapter is a descriptive study that reports the representation of women in OSS slicing by three dimensions. We first slice data over time to show how the gender distribution evolves. Then we slice data by ecosystem, since each of them has different management practices [6]. We also segment the population vertically to analyze women’s distribution among core contributors, those who are more experienced and responsible for the majority of the contributions [28].

When investigating gender distribution, we followed many previous studies [48, 57] and used automated gender inference tools to infer genders based on the information disclosed by contributors, oftentimes names. These methods have certain known limitations and biases, including the imperfect accuracy and the assumption of *binary* gender, which does not reflect the current perception of gender [50]. We are aware that the use of the inference on individuals can be harmful [26, 29]. Therefore, our study only uses name-based gender inference on the population level and treats the results as only an approximation of the real situation [35].

Related Work

Automatic Gender Inference Tools

Researchers have explored various techniques to automatically infer gender of individuals. This section discusses the approaches available to our GitHub source data. Note that all classifiers here assume binary gender, and their benchmarks also consist of only data of binary gender.

Appearance-based gender inference has been extensively studied in the field of computer vision, where many classifiers can achieve an accuracy higher than 90%, even 99% [2] or nearly 100% [62]. However, a large number of GitHub users are using default profile pictures, and there is no guarantee that a contributor’s profile picture is a picture of themselves. Hence, we did not use appearance-based inference because the results would be very unreliable.

Researchers have also explored *text-based gender inference*, which relies on vocabulary and frequency of words [34] and even style markers and structural characteristics [13]. However, our text pieces on GitHub, such as commit messages, are usually short, and the accuracy of this technique is low.

To the best of our knowledge, *name-based gender inference* is the most commonly used approach in the software research community. Certain tools perform the inference based on only an individual's first name. For example, Gender-guesser is a Python package that uses the first name to assign "unknown," "andy" (androgynous), "male," "female," "mostly_male," or "mostly_female" to an individual. In comparison, several tools incorporate one's geolocation or cultural origin into their inference. For example, both Namsor and NameAPI are paid services that infer one's cultural origin based on their last name. Based on benchmark evaluations by Santamaría and Mihaljević [50] and Sebo [51], Gender API and Namsor are the most accurate tools with accuracy higher than 90%. Thus, we pick Namsor as our gender inference tool.

Researchers have started reflecting on the negative impact of automatic binary gender inference tools. Hamidi et al. [26] criticized the tools' assumption of binary gender as "gender reductionism." We acknowledge and agree that the limitation also exists in name-based gender inference, including ours, and caution against using such technology to make individual-level inferences. As we argued previously, we only make population-level inferences to get a general sense of global trends and differences among ecosystems.

Gender Distribution from Prior Studies

With rising awareness of the low gender diversity problem, many studies have attempted to estimate the gender composition in the OSS community. Although all studies report a low percentage of women contributors, these numbers have wide variation ranging from 1% to 12%. Building on the overview of women ratios across years by Trinkenreich [55], we provide an overview of the results reported by prior studies grouped by methods.

Surveys: The first section of Table 14-1 lists the studies that rely on survey data to measure gender distribution. Surveys can capture people's self-identified gender and arguably increase the precision of gender identification [36]. However, survey data, albeit more reliable and accurate, are prone to selection bias [5]. Moreover, survey samples are usually small, making it hard to generalize.

Table 14-1. Women ratios in prior works grouped by data sources and methods

Year	Source	Sample Size	Ratio	Citation	Project
Gender Ratios Reported from Survey Data					
2001	Online survey	5,478	0%	Robles et al. [47]	
2002	Online survey	2,784	1.1%	Ghosh et al. [24]	
2001– 2002	Email	684	2.5%	Lakhani et al. [32]	
2002	Email	79	5%	Hars and Ou [1]	
2003	Online survey	1,588	1.6%	David et al. [15]	
2013	Online survey	2,183	10.35%	Robles et al. [46]	
2015	Online survey	816	24%	Vasilescu et al. [59]	
2017	Online survey	6,000	5%	GitHub [23]	
2017	Online survey	64,000	7.6%	Stack Overflow [52]	
2019	Online survey	119	10.9%	Lee et al. [33]	
2021	Online survey	242	7.6%	Gerosa et al. [22]	
Gender Ratios Reported from Mining Software Repositories					
2012	Email subscribers, US census	1,931	8.27%	Kuechler et al. [31]	
2012	Stack Overflow	2,588	11.24%	Vasilescu et al. [57]	
2015	GitHub, genderComputer [57]	1,049,345	8.71%	Kofink [30]	
2015	GitHub, genderComputer	873,392	9%	Vasilescu et al. [60]	
2017	GitHub, social media	328,988	6.36%	Terrell et al. [53]	

(continued)

Table 14-1. (continued)

Year	Source	Sample Size	Ratio	Citation	Project
2017	OpenStack, genderize.io ⁴	-	10.4%	Izquierdo et al. [27]	
2019	GitHub, Namsor [9]	300,000	9.7%	Qiu et al. [42]	
2019	Gerrit, genderComputer, social media	4,543	8.8%	Bosu and Sultana [7]	
2020	GitHub, genderComputer, Namsor	1,954 core	5.35%	Canedo et al. [8]	
2021	GitHub, genderComputer, Simple Gender [21]	1,634,373	5.49%	Vasarhelyi et al. [56]	
2021	GitHub, genderize.io	65,132	10%	Prana et al. [41]	
2022	Software Heritage [40], gender-guesser ⁵	21.4M	10%	Rossi et al. [48]	
Gender Ratios Reported from Different Ecosystems or Projects					
2014	Mailing list	3,342	9.81%	Vasilescu et al. [58]	Drupal
2014	Mailing list	3,611	7.81%	Vasilescu et al. [58]	WordPress
2016	Online survey	765	5.2%	Sharan [20]	Apache
2005–2016	GitHub	14,905	8%	Cortázar [14]	Linux
2016	Online survey	1,479	2%	Raissi et al. [45]	Debian
2019	GitHub, Namsor	1,601	3.4%	Asri and Kerzazi [3]	Angular.js
2019	GitHub, Namsor	1,824	3.5%	Asri and Kerzazi [3]	Moby

(continued)

⁴www.genderize.io⁵<https://pypi.org/project/gender-guesser/>

Table 14-1. (continued)

Year	Source	Sample Size	Ratio	Citation	Project
2019	GitHub, Namsor	3,723	4.2%	Asri and Kerzazi [3]	Rails
2019	GitHub, Namsor	1,672	5.3%	Asri and Kerzazi [3]	Django
2019	GitHub, Namsor	1,127	4.2%	Asri and Kerzazi [3]	Elasticsearch
2019	GitHub, Namsor	1,735	5.8%	Asri and Kerzazi [3]	TensorFlow
2019	Gerrit, genderComputer	258 core	3.87%	Bosu and Sultana [7]	Android
2019	Gerrit, genderComputer	151 core	3.97%	Bosu and Sultana [7]	Chromium OS
2019	Gerrit, genderComputer	24 core	4.17%	Bosu and Sultana [7]	Couchbase
2019	Gerrit, genderComputer	90 core	7.77%	Bosu and Sultana [7]	Go
2019	Gerrit, genderComputer	68 core	1.47%	Bosu and Sultana [7]	LibreOffice
2019	Gerrit, genderComputer	60 core	10%	Bosu and Sultana [7]	OmapZoom
2019	Gerrit, genderComputer	34 core	2.94%	Bosu and Sultana [7]	oVirt
2019	Gerrit, genderComputer	159 core	3.12%	Bosu and Sultana [7]	Qt
2019	Gerrit, genderComputer	73 core	4.1%	Bosu and Sultana [7]	TYP03
2019	Gerrit, genderComputer	19 core	0%	Bosu and Sultana [7]	Whamcloud
2021	Online survey	2,350	14%	Carter et al. [10]	Linux

Mining software repositories: The second section of Table 14-1 lists the studies that rely on data mining to report gender distribution. In these quantitative studies, researchers often need to infer gender because not all platforms collect users' gender and not all users disclose their genders online. Thus, automatic gender inference tools have become a common practice. Despite the limitations, gender inference based on mined user information provides a more representative, larger-scale sample than the survey approach. It also eliminates the burden on the survey respondents and the efforts taken to collect survey results.

Ecosystems: The last section of Table 14-1 lists studies that report gender ratios in specific software ecosystems. The percentages of women range from 0% (Whamcloud) to 10% (OmapZoom) [7]. However, to the best of our knowledge, there is not a study that covers all major ecosystems, and many of the previous studies focus on a selection of projects rather than the entire ecosystem.

Methods

To conduct an ecosystem-level census, we used data from GHTorrent and retrieved the list of projects in the 20 largest package managers on libraries.io,⁶ a service collecting data of open source packages. We only selected the 20 biggest package managers out of the total 38. Because our automatic gender inference is not perfect and can be used only as a population-level approximation, results in smaller ecosystems can fluctuate and become unreliable. We used data from GHTorrent [25], which provides trace data from GitHub between January 2008 and March 2021. However, we note the limitation that the data between June and December 2019 are missing.

Data Processing Pipeline

Extracting the list of open source infrastructural projects: We consider a GitHub project that is registered at libraries.io as an OSS project. Using the January 12, 2020, version of the dataset from libraries.io, which consists of entries of open source projects registered by the date, we parsed out 1,550,273 unique, valid projects that can be found on GHTorrent.

⁶<https://libraries.io>

Collecting contributions: Due to data traceability, we consider only commits, both code and documentation, as contributions. We acknowledge that this simplification neglects contributions such as management, avocation, and mentorship [54, 55]. However, many of these non-code activities are either untraceable or hard to quantify. Therefore, at this moment, we focus on only tractable contributions.

De-aliasing user entries: Because developers sometimes use different accounts when authoring commits in a project, we perform identity merging through a set of heuristic rules to ensure that we do not over-count users. Our de-aliasing method relies on user-level information, for example, emails and names [19, 61].⁷ For example, if two accounts use the same email and similar names, that is, some or all parts are the same but in different orders, or the same name with similar emails, that is, their emails contain part of their names, their commits could most possibly be credited to one author.

Removing bots: To reduce the impact of bot contribution, we manually evaluate the activity of all users who made at least 1,000 commits in each ecosystem [16]. We found 511 unique bot accounts, which made 5,828,940 commits in total.

Aggregation granularity: To study how women’s participation changes over time, we aggregate data into *three-month windows*, which ensures sufficient interactions among contributors since activities on GitHub are more sparse than those in companies. For windows that have less than 30 contributors whose genders can be inferred, we consider those windows as no activity, as the percentage of women might surge and become an outlier in the data.

Identifying core contributors: Adapting from the validated count-based methods by Joblin et al. [28] and Bosu et al. [7], we identified core contributors in the following way. For each ecosystem, within each three-month window, we first identified projects whose number of commits ranked top 10% in the ecosystem. Then, within each of the top projects, we identified each project’s core developers as those who made more than 10% of the commits within that three-month window. In summary, in our analysis, a core contributor makes more than 10% of the commits in a project whose number of commits ranks top 10% in that ecosystem. We are specifically interested in core developers because cores typically take more responsibility for public project code contribution.

⁷https://github.com/bvasiles/ght_unmasking_aliases

Gender Inference

Of the 45,838,860 GitHub users in GHTorrent, 53.65% do not provide a name, and 3.84% are organizational accounts. We label these users' gender as *Unknown*. We also label users whose names have more than four parts (71,367 (0.16%)) as *Unknown* since a manual checking showed that most of them are names of organizations. We preprocess the remaining users' names by removing punctuations, common titles or prefixes, emails, and URLs.

Then, we infer the gender of each user with Namsor [9], one of the name-based gender inference tools with the highest accuracy [50, 51]. The tool makes inferences based on the first name and the cultural origin of the last name.

Namsor also provides a confidence level that a user's gender is correctly identified. We denote users whose gender inference confidence is lower than 0.7 as *Unknown* gender. Removing inferences with low confidence can increase the overall accuracy of our gender classification, yet setting a high confidence threshold cuts down our data size. Thus, we choose 0.7 as the threshold to retain 83.81% of the gender data. Of 1,823,414 users who have contributed to OSS projects, 911,990 (50.02%) are labeled as men and 54,859 (3.01%) as women. To reduce the effect of *Unknown* gender on our result, we calculate women fraction by

$$\frac{\text{Number of Women Contributors}}{\text{Number of Women} + \text{Men Contributors}}$$

Results

Gender Distributions in OSS and Different Ecosystems

Figure 14-1 shows the overall gender distribution in OSS libraries and its evolution over time. Overall, the percentage of women has been constantly low – no higher than 5.0%. Moreover, the percentage of women among all contributors in OSS projects is lower than that among core contributors.

For the gender distributions in the top 20 most popular OSS ecosystems and their evolution, we observed different patterns in different ecosystems. Due to the space limit, we display only plots from four more representative ecosystems in Figure 14-2: npm, CRAN, PlatformIO, and CPAN.

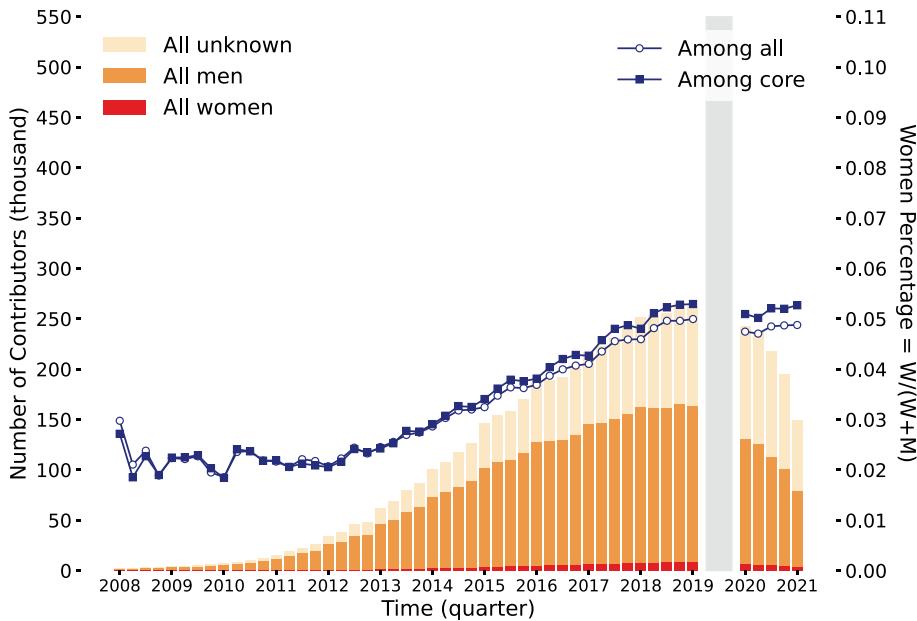


Figure 14-1. Gender representation in OSS contribution overall. The gray bar covers the period where GHTorrent has missing data.

For more figures, please visit our GitHub page.⁸

Figure 14-2a shows the trend of women percentage in the npm ecosystem. The pattern of npm's women percentage change is representative of many ecosystems, such as PyPI, Bower, and Go. Although the overall women percentage has been low all the time (lower than 6%), there is a steady increase overtime.

While most ecosystems exhibit increasing women percentage, the numbers are all lower than 10%, with the exception of CRAN, which reached 10.02% in 2021 (Figure 14-2b). CRAN is the package manager for the R programming language, which is widely used among academic researchers. The higher women percentage in CRAN may be due to the fact that the population of R users is more diverse because they come from various disciplines other than computer science [6].

⁸<https://github.com/CMUSTRUDEL/OSS-gender-census-SEIS2023>

CHAPTER 14 HOW MUCH DO WOMEN BUILD OPEN SOURCE INFRASTRUCTURE?

Moreover, as shown in Figure 14-2c and 14-2d, PlatformIO and CPAN display a puzzling periodicity and minimum growth over the years. This pattern can be due to the fact that PlatformIO is a smaller ecosystem in our dataset. As a result, a small change in team composition can result in a large fluctuation. This also explains why we chose to only present results for the 20 larger ecosystems: the smaller the ecosystem is, the more likely it would be influenced by small changes.

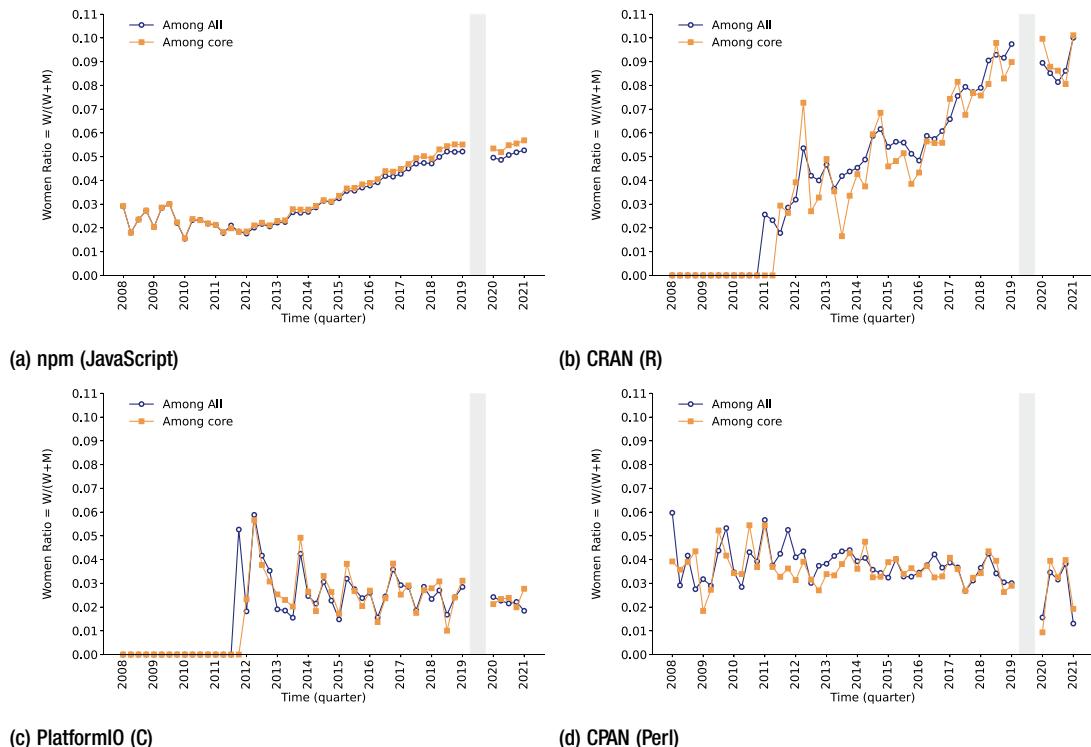


Figure 14-2. Women distributions overall and in selected ecosystems. Gray bars cover the period with missing data on GHTorrent.

For most ecosystems, the percentage of women exhibited an uphill pattern and reached its peak between 2018 and 2021. However, some languages commonly used for system programming – Perl, Rust, and C++ – reached their maximum percentage before 2014. Table 14-2 shows the percentages of women at the end of our data (January–March 2021) and the window during which the maximum percentage of women contributors occurred.

Gender Distributions Among Core Contributors

Starting with women percentage of 2.13% among core contributors and 2.25% among all contributors, the number has been steadily growing between 2008 and 2021. We observed that, while the women percentage among all contributors was higher than among cores in 2008, the difference between them was less than 0.01% in 2014. Between 2014 and 2021, we found that the women percentage among cores has surpassed that among all, leaving a slight but approximately stable margin of 0.3%.

Lastly, comparing the percentage of women among core contributors and among all contributors in 2021 in Table 14-2, we noticed that, in most ecosystems the percentage among core contributors is higher than that among all contributors, with few exceptions such as Meteor, Pub, Cargo, and Hex, which have very small number of women contributors overall.

Table 14-2. Women's participation by package managers (sorted by the number of projects)

Ecosystem	Programming Language	# of Projects	% Women (2021)	Max % of Women	Window of the Max Pct	% Core Women (2021)	Max % Core Women
npm	JavaScript	568,116	5.36%	5.39%	Apr–Jun 2019	5.83%	5.83%
Packagist	PHP	250,687	3.23%	3.58%	Apr–Jun 2018	3.42%	3.89%
Go	Go	236,902	4.33%	4.59%	Oct–Dec 2019	4.49%	4.84%
PyPI	Python	116,819	5.33%	5.61%	Jan–Mar 2019	5.78%	6.03%
RubyGems	Ruby	94,561	5.7%	5.77%	Jul–Sep 2020	6.17%	6.24%
Bower	CSS	57,885	5.48%	5.48%	Jan–Mar 2021	5.76%	5.76%
CocoaPods	Objective-C	52,109	4.5%	4.85%	Oct–Dec 2018	4.66%	4.94%

(continued)

Table 14-2. (continued)

Ecosystem	Programming Language	# of Projects	% Women (2021)	Max % of Women	Window of the Max Pct	% Core Women (2021)	Max % Core Women
NuGet	C#	44,283	4.01%	4.01%	Jan–Mar 2021	4.63%	4.63%
Maven	Java	29,187	5.3%	5.36%	Apr–Jun 2019	5.84%	5.84%
Cargo	Rust	18,466	3.87%	4.52%	Apr–Jun 2014	3.65%	4.66%
Clojars	Clojure	12,551	4.79%	4.95%	Jul–Sep 2020	5.33%	5.33%
Atom	CSS	10,685	4.51%	5.82%	Jul–Sep 2019	5.75%	6.8%
CPAN	Perl	10,365	1.37%	6.15%	Jan–Mar 2008	2.04%	5.26%
Hex	Elixir	7,821	3.81%	3.81%	Jan–Mar 2021	3.64%	3.82%
Meteor	JavaScript	7,795	6.93%	6.93%	Jan–Mar 2021	6.25%	6.25%
Hackage	Haskell	7,570	3.4%	4.05%	Jan–Mar 2019	3.80%	4.09%
Pub	Dart	6,355	3.88%	6.25%	Oct–Dec 2012	3.74%	7.69%
CRAN	R	5,322	10.02%	10.02%	Jan–Mar 2021	10.51%	10.51%
Puppet	Puppet	3,943	1.49%	3.87%	Oct–Dec 2017	1.59%	4.15%
PlatformIO	C++	3,637	1.74%	4.55%	Apr–Jun 2012	2.63%	4.28%
Others	-	23,021					

Main Takeaways

The gender diversity is improving. We observed a slow but steadily increasing trend of women's participation in open source infrastructural projects. Our observation agrees with prior findings [41, 48]. The increasing trend is also observed in most of the ecosystems. While the reasons behind this change over time are beyond the scope of our study, we speculate that some of the past efforts to encourage and support marginalized groups in OSS have taken effect.

Gender distributions vary across ecosystems. Specifically, many ecosystems related to web development, especially front end, for example, Meteor and RubyGems, have higher women percentages. In comparison, several ecosystems related to system programming, for example, CPAN and PlatformIO, have lower gender diversity. Our finding agrees with Vasarhelyi et al.'s finding [56] that contributors in front-end programming languages are more likely to be women.

There are more core women contributors among big open source projects. When computing women's percentage among core contributors, we focused on only the biggest projects, whose commits are ranked top 10% in that ecosystem. We found that, among the biggest projects, whose commits are ranked the top 10% in that ecosystem, the percentage of women among core contributors is higher than that of among all contributors.

Open Research Questions

Reasons behind the increase: While our analysis and several recent studies [41, 48] reported a similar trend of increasing percentage of women among open source contributors, we do not yet understand how this has happened. Is it by chance or because some prior diversity efforts have been effective? Are hackathons [39], coding camps [43], or conferences effective in attracting and retaining women contributors? Future research can analyze the reasons behind the increased women's percentage and reflect on the outcome of prior efforts to improve diversity. Such studies can inform the design and deployment of future diversity and inclusion activities.

Ecosystem difference: Our study provides another piece of evidence that the differences in gender representation could be due to the functions of the programming languages. However, more in-depth and targeted studies are needed to test the speculation or provide a reasonable explanation. Is the disparity due to the nature of the programming languages or some community practices?

A fine-grained examination on women's representation across open source:

Although our analysis found differences in gender representation across ecosystems and the level of contributions, there are more ways to slide the data and pinpoint the places with skewer gender distribution. For example, we examined the percentage of women core contributors among big projects and found that the percentage is higher than among all contributors. This is different from a prior result where the percentage of women among core contributors is much lower than that among all contributors [8]. Future studies can further investigate the relationship between gender distributions and project sizes. Our study also did not investigate the non-code contributions. Future researchers can consider adding contributors who only contributed to issue discussions. There are also non-code contributions that are not visible on social coding platforms. Quantifying the gender distribution among these hidden contributors is an open research question.

Bibliography

- [1] Shaosong Ou and Alexander Hars. Working for free? Motivations for participating in open-source projects. *International Journal of Electronic Commerce*, 6(3):25–39, 2002.
- [2] Luís A. Alexandre. Gender recognition: A multiscale decision fusion approach. *Pattern Recognition Letters*, 31(11):1422–1427, 2010.
- [3] Ikram El Asri and Noureddine Kerzazi. Where are females in OSS projects? Socio technical interactions. In *Working Conference on Virtual Enterprises*, 308–319, Springer, 2019.
- [4] Guilherme Avelino, Eleni Constantinou, Marco Tulio Valente, and Alexander Serebrenik. On the abandonment and survival of open source projects: An empirical investigation. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–12, IEEE, 2019.
- [5] Jelke Bethlehem. Selection bias in web surveys. *International Statistical Review*, 78(2):161–188, 2010.
- [6] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdinand Thung. How to break an API: cost negotiation and community values in three software ecosystems. In *Proceedings of the 2016 24th ACM*

SIGSOFT International Symposium on Foundations of Software Engineering, 109–120, 2016.

- [7] Amiangshu Bosu and Kazi Zakia Sultana. Diversity and inclusion in open source software (OSS) projects: Where do we stand? In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–11, IEEE, 2019.
- [8] Edna Dias Canedo, Rodrigo Bonifácio, Márcio Vinicius Okimoto, Alexander Serebrenik, Gustavo Pinto, and Eduardo Monteiro. Work practices and perceptions from women core developers in OSS communities. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–11, 2020.
- [9] Elian Carsenat. Inferring gender from names in any region, language, or alphabet. *Unpublished*, 10, 2019.
- [10] Hilary Carter and Jessica Groopman. The Linux Foundation report on diversity, equity, and inclusion in open source. <https://www.linuxfoundation.org/tools/the-2021-linux-foundation-report-on-diversity-equity-and-inclusion-in-open-source/>, 2021. Accessed on March 10, 2022.
- [11] Gemma Catolino, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, and Filomena Ferrucci. Gender diversity and women in software teams: How do they affect community smells? In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 11–20, IEEE, 2019.
- [12] Jailton Coelho and Marco Tulio Valente. Why modern open source projects fail. In *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 186–196, ACM, 2017.
- [13] Malcolm Corney, Olivier De Vel, Alison Anderson, and George Mohay. Gender-preferential text mining of e-mail discourse. In *18th Annual Computer Security Applications Conference, 2002, Proceedings.*, 282–289, IEEE, 2002.

- [14] Daniel Izquierdo Cortázar. Gender-diversity analysis of the Linux kernel technical contributions. <https://speakerdeck.com/bitergia/gender-diversity-analysis-of-the-linux-kernel-technical-contributions?slide=48>, 2016. Accessed on January 20, 2022.
- [15] Paul A. David, Andrew Waterman, and Seema Arora. Floss-us the free/libre/open source software survey for 2003. *Stanford Institute for Economic Policy Research, Stanford University, Stanford, CA* (www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf), 2003.
- [16] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. *Detecting and Characterizing Bots That Commit Code*, 209–219. ACM, New York, NY, USA, 2020.
- [17] Edna Dias Canedo, Heloise Acco Tives, Madianita Bogo Marioti, Fabiano Fagundes, and José Antonio Siqueira de Cerqueira. Barriers faced by women in software development projects. *Information*, 10(10):309, 2019.
- [18] Nadia Eghbal. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation, 2016.
- [19] Hongbo Fang, Daniel Klug, Hemank Lamba, James Herbsleb, and Bogdan Vasilescu. Need for tweet: How open source developers talk about their GitHub work on Twitter. In *Proceedings of the 17th International Conference on Mining Software Repositories*, 322–326, 2020.
- [20] Sharan Foga. ASF committer diversity survey. [https://cwiki.apache.org/confluence/display/COMDEV/ASF+Committer+Diversity+Survey +-+2016](https://cwiki.apache.org/confluence/display/COMDEV/ASF+Committer+Diversity+Survey+-+2016), 2016. Accessed on January 20, 2022.
- [21] Denae Ford, Alisse Harkins, and Chris Parnin. Someone like me: How does peer parity influence participation of women on stack overflow? In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 239–243, IEEE, 2017.
- [22] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. The shifting sands of motivation: Revisiting what drives contributors in open source. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 1046–1058, IEEE, 2021.

- [23] GitHub. Open source survey. <https://opensourcesurvey.org/2017/>, 2017. Accessed on March 10, 2022.
- [24] Rishab A. Ghosh, Ruediger Glott, Bernhard Krieger, and Gregorio Robles. Free/libre and open source software: Survey and study, 2002.
- [25] Georgios Gousios and Diomidis Spinellis. GHTorrent: GitHub's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, 12–21, IEEE, 2012.
- [26] Foad Hamidi, Morgan Klaus Scheuerman, and Stacy M. Branham. Gender recognition or gender reductionism? The social implications of embedded gender recognition systems. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–13, 2018.
- [27] Daniel Izquierdo, Nicole Huesman, Alexander Serebrenik, and Gregorio Robles. OpenStack gender diversity report. *IEEE Software*, 36(1):28–33, 2018.
- [28] Mitchell Joblin, Sven Apel, Claus Hunsen, and Wolfgang Mauerer. Classifying developers into core and peripheral: An empirical study on count and network metrics. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 164–174, IEEE, 2017.
- [29] Os Keyes. The misgendering machines: Trans/HCI implications of automatic gender recognition. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–22, 2018.
- [30] Andrew Kofink. Contributions of the under-appreciated: Gender bias in an open-source ecology. In *Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity*, 83–84, 2015.
- [31] Victor Kuechler, Claire Gilbertson, and Carlos Jensen. Gender differences in early free and open source software joining process. In *IFIP International Conference on Open Source Systems*, 78–93, Springer, 2012.
- [32] Karim R. Lakhani and Robert G. Wolf. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *Open Source Software Projects (September 2003)*, 2003.

- [33] Amanda Lee and Jeffrey C Carver. Floss participants' perceptions about gender and inclusiveness: a survey. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 677–687, IEEE, 2019.
- [34] Feng Lin, Yingxiao Wu, Yan Zhuang, Xi Long, and Wenyao Xu. Human gender classification: a review. *Int. J. Biom.*, 8(3/4):275–300, 2016.
- [35] Jeffrey W. Lockhart, Molly M King, and Christin Munsch. What's in a name? Name-based demographic inference and the unequal distribution of misrecognition. 2022.
- [36] Mike Medeiros, Benjamin Forest, and Patrik Öhberg. The case for non-binary gender questions in surveys. *PS: Political Science & Politics*, 53(1):128–135, 2020.
- [37] Courtney Miller, David Widder, Christian Kästner, and Bogdan Vasilescu. Why do people give up FLOSSing? A study of contributor disengagement in open source. In *International Conference on Open Source Systems, OSS*, 116–129, Springer, 2019.
- [38] Dawn Nafus. “Patches don't have gender”: What is not open in open source software. *New Media & Society*, 14(4):669–683, 2012.
- [39] Lavinia Paganini and Kiev Gama. Engaging women's participation in hackathons: A qualitative study with participants of a female-focused hackathon. In *International Conference on Game Jams, Hackathons and Game Creation Events 2020*, 8–15, 2020.
- [40] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. The software heritage graph dataset: public software development under one roof. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 138–142, IEEE, 2019.
- [41] Gede Artha Azriadi Prana, Denae Ford, Ayushi Rastogi, David Lo, Rahul Purandare, and Nachiappan Nagappan. Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in OSS. *IEEE Transactions on Software Engineering*, 2021.

- [42] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. Going farther together: The impact of social capital on sustained participation in open source. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 688–699, IEEE, 2019.
- [43] Huilian Sophie Qiu, Yang Wen, and Alexander Nolte. Approaches to diversifying the programmer community – the case of the girls coding day. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 91–100, IEEE, 2021.
- [44] Huilian Sophie Qiu, Zihe H. (co-first author) Zhao, Tielin Katy Yu, Justin Wang, Alexander Ma, Hongbo Fang, Laura Dabbish, and Bogdan Vasilescu. Gender representation among contributors to open-source infrastructure – an analysis of 20 package manager ecosystems. In *International Conference on Software Engineering – Software Engineering in Society, ICSE SEIS*, IEEE, 2023.
- [45] Mahin Raissi, Molly de Blanc, and Stefano Zacchiroli. Preliminary report on the influence of capital in an ethical-modular project: Quantitative data from the 2016 Debian survey. *Journal of Peer Production*, (10):1–25, 2017.
- [46] Gregorio Robles, Laura Arjona Reina, Jesús M González-Barahona, and Santiago Dueñas Domínguez. Women in free/libre/open source software: The situation in the 2010s. In *IFIP International Conference on Open Source Systems*, 163–173, Springer, 2016.
- [47] Gregorio Robles, Hendrik Scheider, Ingo Tretkowski, and Niels Weber. Who is doing it. *A Research on Libre Software Developers*, 2001.
- [48] Davide Rossi and Stefano Zacchiroli. Worldwide gender differences in public code contributions: and how they have been affected by the COVID-19 pandemic. *Proceedings of the 44th International Conference on Software Engineering (ICSE 2022) – Software Engineering in Society (SEIS) Track*, 2022.

- [49] Daniel Russo and Klaas-Jan Stol. Gender differences in personality traits of software engineers. *IEEE Transactions on Software Engineering*, 2020.
- [50] Lucía Santamaría and Helena Mihaljević. Comparison and benchmark of name-to-gender inference services. *PeerJ Computer Science*, 4:e156, 2018.
- [51] Paul Sebo. Performance of gender detection tools: a comparative study of name-to-gender inference services. *Journal of the Medical Library Association: JMLA*, 109(3):414, 2021.
- [52] Stack Overflow. Developer survey results. <https://insights.stackoverflow.com/survey/2017>, 2017. Accessed on May 1, 2022.
- [53] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Comp Sci*, 3:e111, 2017.
- [54] Bianca Trinkenreich, Mariam Guizani, Igor Wiese, Anita Sarma, and Igor Steinmacher. Hidden figures: Roles and pathways of successful OSS contributors. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2):1–22, 2020.
- [55] Bianca Trinkenreich, Igor Wiese, Anita Sarma, Marco Gerosa, and Igor Steinmacher. Women’s participation in open source software: A survey of the literature. Preprint at *arXiv:2105.08777*, 2021.
- [56] Orsolya Vasarhelyi and Balazs Vedres. Gender typicality of behavior predicts success on creative platforms. Preprint at *arXiv:2103.01093*, 2021.
- [57] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study of Stack Overflow. In *2012 International Conference on Social Informatics*, 332–338, IEEE, 2012.
- [58] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study. *Interacting with Computers*, 26(5):488–511, 2014.

- [59] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 3789–3798, 2015.
- [60] Bogdan Vasilescu, Alexander Serebrenik, and Vladimir Filkov. A data set for social diversity studies of GitHub teams. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 514–517, IEEE, 2015.
- [61] Bogdan Vasilescu, Alexander Serebrenik, Mathieu Goeminne, and Tom Mens. On the variation and specialisation of workload – a case study of the gnome ecosystem community. *Empirical Software Engineering*, 19(4):955–1008, 2014.
- [62] Ji Zheng and Bao-Liang Lu. A support vector machine classifier with automatic confidence and its application to gender classification. *Neurocomputing*, 74(11):1926–1935, 2011.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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.

PART IV

Across the Gamut of Opportunities: Initiatives and Interventions

CHAPTER 15

Beyond Classroom: Making a Difference in Diversity in Tech

Barbora Buhnova, Masaryk University, Czechia.*

With all the opportunities and risks that technology holds in connection to our safe and sustainable future, it is becoming increasingly important to involve a larger portion of our society in becoming active co-creators of our digitalized future – moving from the passenger seat to the driver seat. Yet, despite extensive efforts around the world, little progress has been made in growing the representation of certain communities and groups in software engineering. This chapter shares one successful project, called Czechitas, triggering a major social change in Czechia, involving 1000+ volunteers to support 50,000+ women on their way toward software engineering education and career.

Introduction

The past decade has witnessed the emergence of hundreds of initiatives around the world supporting various underrepresented groups on their pathway toward software engineering, whether connected to universities [13] or companies [15] or run as independent nonprofit organizations [14]. Although the initiatives often start with a great vision and high volunteering commitment, after a few years into the activities, it becomes challenging to sustain the volunteering energy and commitment in the face of the very slow progress toward the better. In those moments, the success cases by others can be what helps us keep going.

The initiative featured in this chapter, called Czechitas [6], started in 2014 in Czechia, with a simple idea to bring tech closer to girls and girls closer to tech, in reaction to the strong underrepresentation of women in tech in the country (see Figure 15-1). The prompt snowball effect helped us build a community around the joint vision to empower and encourage girls and women to engage in computing education and career transition and to show them that software engineering is an interesting career direction that is not necessarily difficult nor limited to one gender. Initially established to provide women in Czechia with an opportunity to put their hands on programming, it now contributes to a major social change in the country.

Over time, Czechitas has become a movement that has attracted a strong community of tech professional volunteers (over 1,000) and companies (over 100) and given rise to a portfolio of women-tailored courses in various areas of software engineering, such as programming, web development, mobile app development, data science, cybersecurity, or testing (over 1,500 courses delivered so far). We have influenced over 50,000 women (over 30,000 via live events and over 20,000 via online tutorials) who graduated from our courses to use their new tech skills to change their education path or advance their careers.

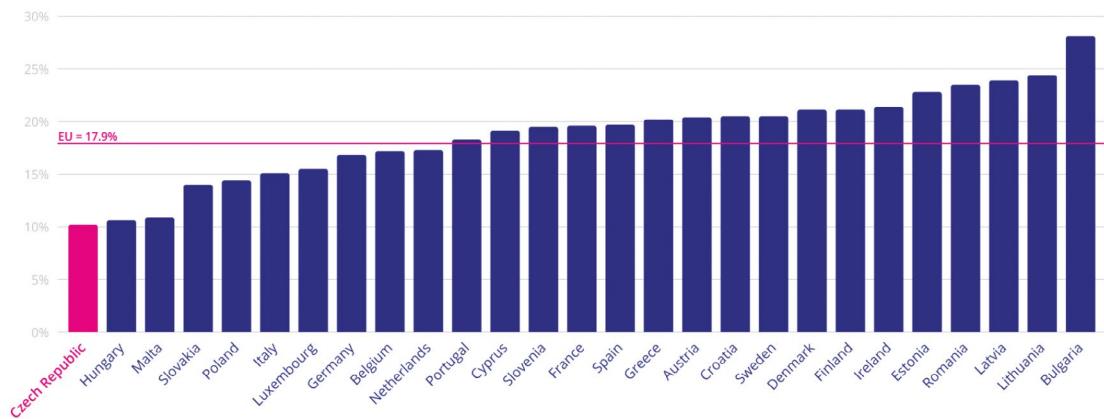


Figure 15-1. Women ICT professional (Eurostat, 2019 data) [8]

Czechitas Mission We inspire, train, and guide new talents toward stronger diversity and competitiveness in tech.

Thanks to the success of our education activities with hundreds of events a year (each receiving more registrations than its capacity), we have become recognized as the leading platform in Czechia actively addressing gender diversity in tech. In this chapter, we share the lessons we learned about the low representation of women in tech and effective strategies in supporting women on their way to software engineering and discuss the ingredients that helped us succeed, the obstacles and challenges we faced, and the progress yet to be made.

Why Are There So Few Women in Tech?

Across Europe, only 19.1% of tech professionals are women (according to 2021 data) [8], with Czechia being the last on the list. The major reasons behind the trend in our region according to our recent study (with 70% of participants from Czechia and Germany) [9] are

1. **Access:** The first hole in the leaky pipeline on girls' pathway toward software engineering is linked to the *missing access to encouragement and support*, together with the *missing access to suitable education* that would be able to build on the interests of girls that often span across multiple disciplines.
2. **Stereotypes:** The ability to *see herself as a software engineer* is then challenged by the *perception of the software engineering* as a field not leading to a purpose the girl would like to dedicate her future to. Often, the close family and friends step in, in this moment to direct girls away from software engineering *with the intention to protect them* from a future where they cannot really imagine the girls becoming successful. Interestingly, the intentions are meant well, to protect the girls, which shows how crucial it is to help parents (and mainly mothers) to understand that software engineering can be a great career choice for their daughters.

3. **Confidence:** The next hole in the leaky pipeline comes when girls find themselves in the classroom, often *surrounded by more experienced learners* (typically boys). For the little girls who often excel in other subjects, it can be hard to fall in the category of a slow novice learner. The girls often mention frustrations of *low self-efficacy, inadequacy, and missing experience of success* in the presence of a classroom dynamic being monopolized by the earlier technology adopters.
4. **Sense of belonging:** The girls who resist through the earlier three challenges and find themselves on the education pathway toward software engineering find themselves in classrooms surrounded predominantly by boys. While this is a comfortable environment for some, many in the study reported *not feeling comfortable to express themselves, facing sexism or unwanted attention, and missing relatable role models and mentors*, which led them to reconsider whether this was the environment they would be willing to spend the rest of their lives in.
5. **Feeling valued:** The last hole in the leaky pipeline challenges the women who entered software engineering careers, as some of them emphasize the struggle of not feeling valued at the workplace. The reasons are different for the women with *stereotypical talent spectrum* (that matches the talent spectrum typical among their men colleagues, typically being very technical) and *non-stereotypical talent spectrum* (bringing not-that-common talents to the table, typically more multidisciplinary and human-oriented). While the first group feels *tired of proving them wrong*, the second group feels frustrated from *their strengths viewed as second class* and from *missing appreciation*.

Supporting Women on Their Way to Tech

In Czechitas, we understand that plumbing the leaky pipeline can hardly be done by isolated and uncoordinated efforts. This section discusses the interlinked pillars of our activities (see Figure 15-2), listing examples of the activities and events we delivered in 2022.

Czechitas Pillar I: Awareness

One of the crucial success factors for a change toward improving gender balance in software engineering is the actual understanding that we are in a disbalanced state that further reinforces itself due to the factors discussed earlier. The efforts toward encouraging women to join software engineering cannot make a difference unless the society, education system, and corporate environment welcome and support the change (understanding it as a push toward the real equilibrium, not a push out of it).

In Czechitas, we are investing substantial effort in awareness around the topic. In 2022 alone, we participated in over 20 conferences and panel discussions; gave numerous interviews in TV, radio, and other media; and organized talks to students and teachers at high schools and to tech professionals in our partner companies. We were visible with a booth at 15 festivals and family days across Czechia. Over 2022, Czechitas was mentioned in 508 articles, reaching a major part of the Czech population. In 2021, we also launched a Czechitas podcast, which in 2022 reached over 14,676 listeners. Furthermore, our website was in 2022 visited by 123,785 unique visitors, and our newsletter was followed by 25,983 subscribers.

The next step in raising awareness among the general public is to make it as easy as possible to get the first exposure to coding in a fun, enjoyable, and community way. To this end, we, for instance, organize an Advent Christmas Coding campaign (following the tradition of an advent calendar, in which instead of a sweet treat, each day holds a coding assignment along a story of bringing Mr. Gingerbread home for Christmas), which is being followed by hundreds of people. Furthermore, in collaboration with the Ministry of Education, Youth and Sport, we, for example, co-organized the #DigiEduHack hackathon. And in collaboration with Czech universities, we run the Czechitas Thesis Award to give visibility to exceptional bachelor theses authored by girls. All these activities typically repeat every year.



Figure 15-2. *The pillars of Czechitas activities*

Czechitas Pillar II: Training

Since the start of our activities in 2014, we have been improving the education design of our courses to reflect the needs of our audience – women and girls who are very often later technology adopters or career changers – with an emphasis on providing suitable first contact with software engineering, creating a safe and supportive environment for novice learners, accommodating differences in the learning speed of each student, building self-confidence, and supporting sustaining long-term interest, which we also publish [2, 10]. In 2022, we delivered 242 live software engineering courses with 8,082 participants (see Figure 15-3 for participation since 2015), with the courses around web development and data science scoring as the most popular ones.

Although most of the training is targeted to women and girls, we are also investing in training elementary-school and high-school teachers (irrespective of gender). And some mixed-gender activities were organized also for children (seven week-long summer camps in the summer of 2022, besides others) and high-school kids, although in case of high schools, it is already important to offer also girl-only courses (three week-long summer schools for high-school girls were given in 2022). Besides, training courses for mixed audience are also provided on events such as family days (we were present at over 20 such events in 2022).

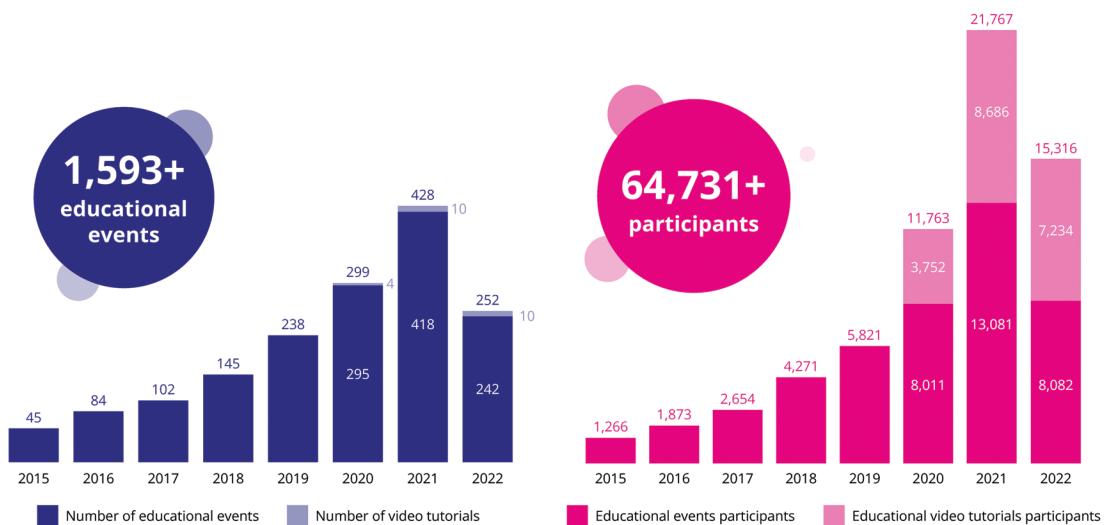


Figure 15-3. Czechitas participation

Czechitas Pillar III: Career Transition

As many women in our community intend to enter software engineering as their future profession, some of our activities are intentionally designed to facilitate this journey, whether software engineering is to become their first job or they intend to change their career [3].

In cooperation with our partner companies, we have identified three career pathways that appear to be the most suitable entry points to software engineering in Czechia. These are (1) *web development* (including courses on JavaScript, React, HTML/CSS, Bootstrap, Git, UX design, etc.), (2) *data analytics* (including courses on Python, databases, SQL, statistics, Power BI, etc.), and (3) *testing* (including courses on requirements engineering, Agile processes, manual testing, issue tracking, regression testing, smoke testing, basics of automated testing, browsers, API, databases, version control, etc.).

For the three directions, we have developed a complex career transition support within so-called Digital Academies. A Digital Academy is a four-month program for a group of 30 women (and involving around 5–15 partner companies), which, besides individual courses covering the topics outlined previously and taking place three to four times a week (evenings on working days, full days on weekends), includes also pairing of the students with mentors from the companies to support them in developing their

own projects, a hackathon, career support, and further events offered by the partner companies. In 2022, we have run ten Digital Academies across four major cities in Czechia, with over 60% of the graduates receiving a job offer within three months from graduating from the academy.

To facilitate the career transition also for the women who opt to customize their training journey (not attending a Digital Academy), our career consultants provide hundreds of career consultations each year (327 in 2022), and we twice a year organize a Czechitas Job Fair, where our graduates can meet the representatives of our partner companies (each job fair attended by about 350 graduates and 30 companies).

Czechitas Foundation: Community

The foundation that supports all our activities is the community, which involves the participants and graduates of our courses, tech professionals who teach with us, mentors, course facilitators, and our partner companies. The fact that many members in our community are men helps us not only engage more tech professional allies in our vision but also influence a more supportive environment in tech companies where our graduates land. To support the blending of the community and increasing the sense of belonging of our graduates also in the mixed-gender environment, we regularly engage in organization of tech meet-ups and hackathons, as well as informal CzechiPubs that regularly take place in ten different cities across Czechia.

Making a Difference

The positive influence of Czechitas activities in Czechia is already visible in the shifted perception of software engineering as an education pathway and career choice to be considered by any gender. That not only motivates many girls to consider software engineering in their choice of a university study field (with the representation of women among ICT students changing from 12% in 2016 to 17% in 2021 in Czechia [5, 7], moving the country closer to the European average; see Figure 15-4) but is likely also having secondary influence on all who so far hesitated to join software engineering.

What Helped Us Succeed

Building Czechitas was only possible thanks to a coordinated effort of hundreds of people (90 employees and over 1,000 volunteers). Over the past eight years of our existence, we came to understand the ingredients without which this would not be possible:

- **Great leadership and love for what we do** is giving us the sense of purpose, energy, and direction, holding us together and keeping us focused. Mentors from partner companies and beyond have been of great help to guide us through the design of our leadership and expansion strategy.

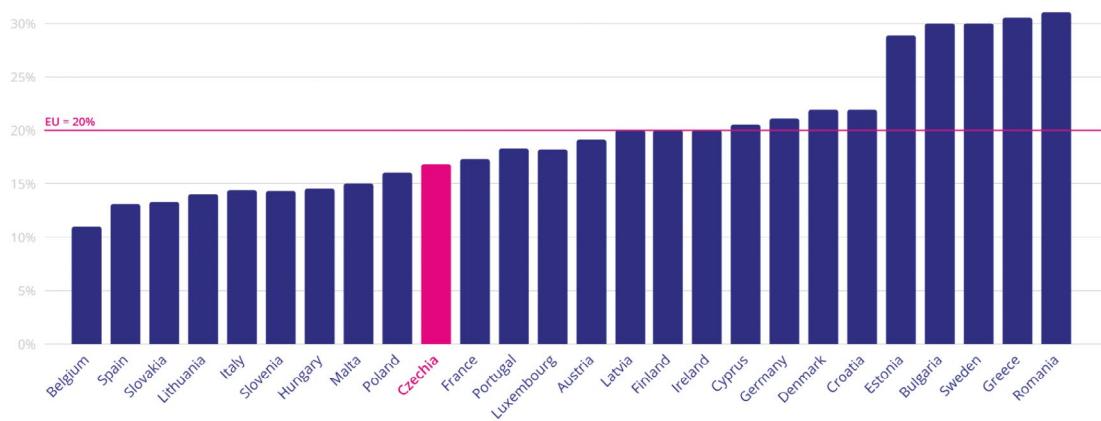


Figure 15-4. Women ICT students (Czech Statistical Office, 2021 data) [5]

- **Visual and playful communication** is giving us the fresh flavor of fun and joy that we all (students as well as trainers and volunteers) enjoy joining even after a tiring day at school or work. The informal and visually attractive communication helps us share the love for our brand.
- **Community and sense of belonging** is crucial for connecting those who strive to learn with those who strive to share and teach and those who want to support the connection. It helps our student feel home and make it easier for them to keep going even when learning gets hard.

- **Inclusive environment and encouragement** makes it safe for our students to make mistakes and experience success, have the opportunity to exchange knowledge, collaborate, and get personalized feedback and guidance. Specific strategies and interventions we have developed to support novice learners and their self-efficacy have been key in this direction [2].
- **Knowledge and understanding** is crucial for us to design our activities with insight into the frustrations steering women away from software engineering [9] and effective strategies to support girls and women in tech education [10] and career transition [3]. We invest our time in sharing the lessons we have learned [2, 3, 9] and learning from other initiatives from across the world (e.g., within the EUGAIN network; see <https://eugain.eu/>).
- **Creating and sharing stories** helps us inspire our students, bring them closer to relatable role models, and give them hope and confidence that with some work and dedication, a transition into software engineering is possible. The stories (each featuring an inspiring woman who changed her career toward tech) are published in our blog, communicated via social networks, and used in media articles. These women inspire others as speakers and panelists in our events and as guests in the Czechitas podcast.
- **Sustainable financial model** helps us sustain a team employed to run the organization. The model stands on financial participation of the students, partner companies, foundations, and individual donors, with an intention to reach out also to the government level in the future. The most crucial pillar of our financial sustainability is the partner companies, which are beside their yearly partnership contributions (depending on the level of partnership) helping us cover certain costs (e.g., offering their office spaces for events, motivating their employees to volunteer as mentors) and opening doors toward further funding opportunities (e.g., with global foundations connected to their company).

Obstacles and Challenges We Faced

As any organization that has substantially outgrown its own plans and expectations, Czechitas has undergone numerous changes and readjustments over its course of existence. And although we are trying to publish the effective setup that works for us now [2, 3, 4], our first steps were highly organic and experimental, which was key to learning what worked for the context we were in. With our enthusiasm and “always yes” spirit, we walked many paths that we failed and rolled back, but we also faced numerous obstacles and challenges that we withstood:

- **Scaling the organization:** Turning a nonprofit start-up into a scale-up is a challenge on its own, as the means for achieving stability are different from traditional companies – besides the discussed financial stability, also in terms of sustained volunteering involvement and brand building. We needed to learn to manage the mix of the innovative and largely self-sacrificing founding community with the necessary systematic and organized spirit of new employees. We needed to learn to prioritize and say no to some activities that the team felt strongly for.
- **Being misunderstood:** As a large organization, we needed to learn to communicate our mission well so that it is not misunderstood, knowing that anything that damages the brand may sink the whole boat. Namely, we needed to help our partner companies understand what level of expertise is realistic to achieve in our students, help our students understand what time investment and commitment it takes to change direction toward tech, and help our society understand why our focus on women is key to the success of our society as a whole.

- **Quantifying the impact of our activities:** One of the important challenges that we are still facing is our ability to quantify the impact of our individual interventions and activities, as it is difficult to isolate the effects of each one of them – more so that the impact is often very subtle and propagates over long periods of time (e.g., a woman making a few steps toward tech education inspiring her friend to make a major shift toward tech, who then inspires her daughter to study CS at university). So although we have a Data & Impact team at Czechitas, with substantial data available, the numbers we have (e.g., the number of women who change their career to tech each year) are still only the tip of the iceberg of the real impact we strive for, which is the shift in the collective mindset of the entire society, leading to a sustained change.

Progress Yet to Be Made

With the increasing number of Czechitas graduates who are joining the software engineering industry, often as very junior (in terms of their software engineering expertise) and diverse (in terms of their talents and competencies) members, we find it crucial to assist the companies to improve the inclusiveness of their environment to integrate and leverage the new diverse talent. In 2020, we made the first step toward that goal via designing a *Diversity Awareness Training*, which was since then delivered to over 300 managers (mostly from Central and Eastern Europe) across some of our partner companies. The concepts that have shown to be the most crucial to discuss and understand during these trainings are outlined in the following:

- **Diversity does not come easy, but it pays off.** Avoiding diversity is natural to human individuals, but dangerous to humankind.¹ The same is true for corporate environment. We need to acknowledge that diverse teams might have a harder time at the start (as illustrated with the Tuckman's Model of Team Dynamics in Figure 15-5), but in the long term, diversity is firmly correlated with higher performance [11, 12].

¹ Our quote inspired by the statement “Diversity is the new Darwinism” by the Great British Diversity Experiment [1].

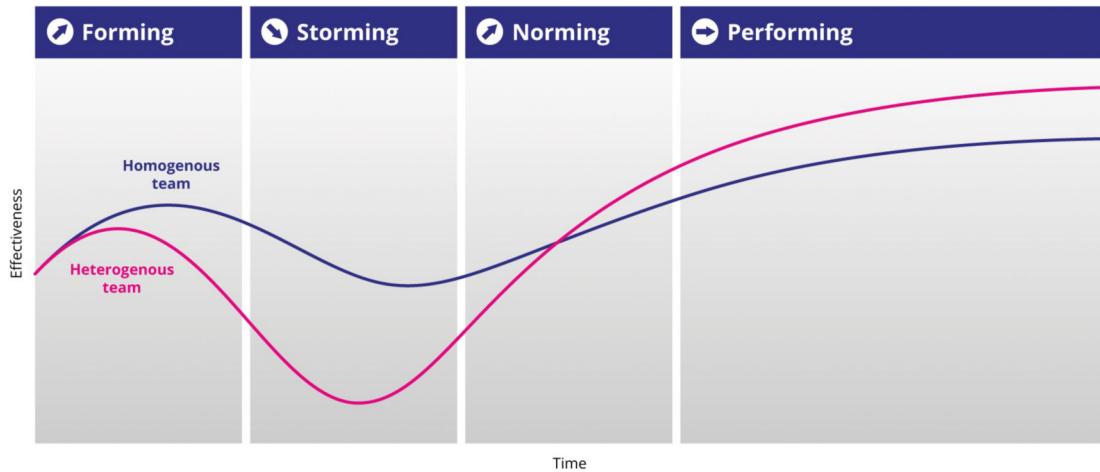


Figure 15-5. Tuckman's Model of Team Dynamics with an illustration of different dynamics observed in homogeneous and heterogeneous teams

- **We too often lose talented people by missing the talent in them.**
We are all talented, in many diverse ways. It is the task of the manager to recognize and direct the talent toward team success. The fact that a person uses a different talent spectrum (approaches problems and situations differently) does not make them more/less suitable for software engineering as such. There is no such thing as a second-class citizen when it comes to the talents we need in software engineering.
- **Biases evolve to help us navigate complexity, but they are not serving us well when making assumptions about the potential in people.** The dark side of biases is that we tend to judge people's potential based on how their talent spectrum matches the talent of already-successful ones, without realizing that the successful ones embody the skills and conditions that worked when they joined the field (in the past) while we are now choosing the software engineers for the future.

- **Connection is built through communication.** There are many unhealthy communication patterns around diversity, which often go against the purpose of making us all feel the sense of belonging. It is important to create safe space, in which we can learn to communicate our differences but also ask about the differences of others. Mistakes are part of that learning, and forgiveness of the mistakes shall be encouraged if the mistakes were done in the process of learning and not repeated blindly. It is important to create a safe space to acknowledge our biases and stop shaming one another for them.
- **Avoid the quick fixes; remove the barriers instead.** Encourage curiosity about why certain communities are underrepresented in software engineering. What are the barriers they face, and what can we do to remove them or make their journey lighter in the presence of the barriers (e.g., the care-taking on the side of most women)? Avoiding the conversation and looking away from the differences in our experiences might lead the community to assume that the underrepresentation is the lower-fit problem, which is dangerous because it leads to pushback on any diversity support one might try to introduce.
- **Change takes time.** Promoting I&D is more complex than it might seem at first. It is crucial to know how to start to see the first positive effects soon and be able to use them to get more people on board toward promoting I&D further. Choose your first steps well and invest in them. The investment will pay off.

Summary

Making a difference in improving gender balance in software engineering on the scale of the whole country is not easy, but is possible. And it is very rewarding to be part of such a movement. In 2021, the social impact of Czechitas activities was recognized at the European Union level via winning the EU Social Economy Award (over 180 organizations nominated) in the Digitalisation and Skills category and in 2022 winning the global Equals in Tech Award (155 organizations nominated) in the Skills category. We hope our example can inspire others, which is also why we are eager to share the lessons learned from our journey.

Acknowledgment

This chapter was made possible thanks to the great dedication and support of the entire Czechitas team. Besides, it has been supported by the COST Action CA19122 – European Network for Gender Balance in Informatics (EUGAIN).

Bibliography

- [1] Amanda Bennett. Case study: The great British diversity experiment. FairPlay Ltd., 2016.
- [2] Barbora Buhnova and Lucia Happe. Girl-friendly computer science classroom: Czechitas experience report. In *European Conference on Software Architecture*, 125–137, Springer, 2020.
- [3] Barbora Buhnova, Lucie Jurystova, and Dita Prikrylova. Assisting women in career change towards software engineering: experience from Czechitas NGO. In *Proceedings of the 13th European Conference on Software Architecture – Volume 2*, 88–93, 2019.
- [4] Barbora Buhnova and Dita Prikrylova. Women want to learn tech: Lessons from the Czechitas education project. In *2019 IEEE/ACM 2nd International Workshop on Gender Equality in Software Engineering (GE)*, 25–28, IEEE, 2019.
- [5] Czech Statistical Office. Human resources in information technology. 2021. Available online at URL www.czso.cz/documents/10180/165376696/063015-21.pdf/c7e96151-b285-4388-9384-532e55f4a318?version=1.2.
- [6] Czechitas. Czechitas annual report 2021. 2022. Available online at URL <https://is.muni.cz/go/u6ji13>.
- [7] Eurostat. Female students under-represented in ICT. 2016. Available online at URL <https://ec.europa.eu/eurostat/web/products-eurostat-news/-/edn-20190425-1>.

- [8] Eurostat. ICT specialists in employment. 2022. Available online at URL https://ec.europa.eu/eurostat/statistics-explained/index.php?title=ICT_specialists_in_employment.
- [9] Lucia Happe and Barbora Buhnova. Frustrations steering women away from software engineering. *IEEE Software*, 39(4):63–69, 2022.
- [10] Lucia Happe, Barbora Buhnova, Anne Kozolek, and Ingo Wagner. Effective measures to foster girls interest in secondary computer science education. *Education and Information Technologies*, 26(3):2811–2829, 2021.
- [11] Dame Vivian Hunt, Dennis Layton, and Sara Prince. Why diversity matters. McKinsey, 2015. Available online at URL www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/why-diversity-matters.
- [12] Rocio Lorenzo and Martin Reeves. How and where diversity drives financial performance. Business Harward Review, 2018. Available online at URL <https://hbr.org/2018/01/how-and-where-diversity-drives-financial-performance>.
- [13] Minerva Informatics Equality Award. Best practices in supporting women. 2022. Available online at URL www.informatics-europe.org/society/minerva-informatics-equality-award/best-practices-in-supporting-women.html.
- [14] Sarah K. White. 19 organizations advancing women in tech. 2022. Available online at URL www.cio.com/article/215709/16-organizations-for-women-in-tech.html.
- [15] Hannah Williams. Best initiatives for women in tech. 2017. Available online at URL <https://techmonitor.ai/technology/hardware/best-initiatives-women-tech>.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 16

Toward More Gender-Inclusive Game Jams and Hackathons

Kiev Gama, Federal University of Pernambuco, Brazil.*

Lavinia Paganini, Eindhoven University of Technology, The Netherlands.

Rafa Prado, Federal University of Pernambuco, Brazil.

Claudia Ferraz, Federal University of Pernambuco, Brazil.

Dayanne Coutinho, Federal University of Pernambuco, Brazil.

Wendy Mendes, Federal University of Pará, Brazil.

Gustavo Pinto, Federal University of Pará and Zup Innovation, Brazil.

George Valença, Federal University of Pernambuco, Brazil.

Game jams and hackathons are time-bounded collaborative events where participants are challenged to gather in teams, ideate a project, and develop a game or another kind of software, respectively [24, 27, 33]. While game jams focus on creating games under a particular theme, hackathons involve developing other types of software applications (e.g., web, mobile) to tackle a problem. Students, professionals, and enthusiasts have different motivations for participating in such events (e.g., learning, networking, and having fun). Regardless of specific motivations, these venues work as an informal setting for learning where participants can gain experience while acquiring technical (e.g., programming, design) and nontechnical skills (e.g., teamwork, presentation skills). These collaboration spaces also offer rich networking opportunities and boost

participants' employability, regardless of being beginners or professionals. Although so many advantages sound attractive to participants, these events typically have an overwhelming majority of men participating. Despite a few isolated initiatives to create more gender-inclusive game jams and hackathons [25, 31, 36, 38], the culture and ethos created around these events are not perceived as welcoming to historically underrepresented groups in the software development field, such as women and LGBTQIA+ people.

These efforts are majorly focused on broadening the participation of women in the field [37] and approach gender from a traditional binary and cisnormative woman/man perspective that ends up unintentionally discouraging or excluding transgender (binary and non-binary) and gender non-conforming (TGNC) people [28]. Transgender and gender non-conforming people tend to be invisible in initiatives around gender diversity in computer science and related courses. An important standpoint to embrace gender diversity in society as a whole is to accept that gender goes beyond a binary woman/man perspective [5].

LGBTQIA+ students on CS-related courses already have a higher probability of dropping out of universities due to their low sense of belonging [44]. When looking into more specific problems of higher education trans-students, many of them consider harassment and violence as other reasons to drop out [16]. The reality in Latin America is of violence against transgenders, with a high murder rate toward them, transphobia, hate and many forms of extreme violence, and micro-aggressions [39]. Since the transgender community has higher unemployment and poverty rates compared with the cisgender population [51], education can be an important tool to attain a job and allow social mobility. By acquiring technical skills in information technology and software engineering, low-income trans-people can be empowered [26]. This population is often overlooked in studies in software engineering.

Under lenses that consider women and LGBTQIA+, with highlights on TGNC people, this chapter consolidates findings from different research initiatives [29, 30, 35] we performed on gender diversity in game jams and hackathons. We combined quantitative and qualitative research methods to analyze a total of 330 responses from surveys and 28 interviews with women and LGBTQIA+ people. We identified some of the main motivators and demotivators for people from these historically underrepresented groups to participate in these events. We also report contrasting perspectives that cis-men have in comparison with the ones from women and TGNC people about gender issues in these events and highlight the typical sexist behavior of men participating in hackathons and game jams that definitely shall not take place. Finally, we propose recommendations

for a more welcoming space and truly gender-inclusive game jams and hackathons. Our experience suggests that these events, if well planned and executed, may become drivers to change attitudes and stereotypes regarding gender in computing.

Background

Gender Beyond the Woman-Man Binary

Gender should not be limited to a binary woman/man perspective. We acknowledge that there are many gender identities and gender expressions that can be very particular to social and cultural contexts (e.g., *muxe* in Mexico, *travesti* in Brazil) [7], and it would be out of scope, as well as challenging, trying to explore that in detail here. For clarification purposes, in this section, we highlight concepts that will be helpful in the context of this chapter.

Transgender (or simply *trans*) refers to “people who move away from the gender they were assigned at birth, people who cross over (*trans-*) the boundaries constructed by their culture to define and contain that gender” [10]. *Gender identity* refers to how one perceives oneself, and it is not associated with sexual attraction. *Cisgender* (or simply *cis*) is when the gender identity aligns with their birth-assigned gender [21]. Note that *straight* is not the opposite of *trans*; a transgender person can be heterosexual too. When someone defines our society as *cisnormative*, it indicates that our common sense is to accept only cisgender behavior and marginalize people that do not follow it [3].

A transgender person can be also *binary* or *non-binary*. Non-binary trans-people are “individuals whose identity is not exclusively man or woman. While some non-binary individuals identify as both men and women, others have identities that are on the spectrum between man and woman, a different gender entirely, or do not identify with any gender” [46]. Non-binary can be used as identity but is more an umbrella term that serves to group identities like *genderfluid* (i.e., the identity of the person flows between other identities) and *agender* (i.e., the absence of gender).

Gender *non-conformity* is when your *gender expression* (i.e., the way of dressing, mannerisms, pronouns, and other characteristics) does not conform to stereotypical gender expectations for your assigned gender [21]. For example, a masculine cisgender woman can define herself as a gender non-conforming person, because she has gender expression associated with the masculine gender, but stills identify herself as a woman.

Gender Diversity in the Information Technology Industry

It is important to recognize that the lack of gender inclusion is present in the broader area of technology, which is one of the sectors that most need diversity in their workforce [18, 20]. The inclusion of women can be beneficial in the market, given that more diverse teams tend to have better results in the same tasks than nondiverse teams – for using collective intelligence [52]. Also, radical company innovations tend to come from more diverse groups, but it is necessary to manage these groups and respect the different present backgrounds [6]. The sense of non-belonging of the LGBTQIA+, which is underrepresented in the technology sector, is also an early barrier that not only contributes to the fear to enter this field of work but also makes the people uncomfortable and more willing to give up their graduation and careers in computing-related courses [44]. LGBTQIA+ people still suffer direct impediments to entering the tech industry because of prejudice and discrimination. Large enterprises are talking about the importance of inclusion, especially because the culture within the companies cannot be changed without working with employees and their perceptions about the LGBTQIA+ community [20].

If we narrow down the perspective to the gaming industry, according to the 2019 International Game Developers Association (IGDA) developer satisfaction survey [49], which collected 1,116 answers, respondents were predominately identified as male (71%), and 24% identified as female, 3% identified as non-binary, and 2% preferred to self-describe other denominations. Women are still one-fourth of the workforce. In a separate question, 4% of respondents identified as transgender.

In regard to sexual orientation, 79% of respondents identified as heterosexual, 4% as homosexual, and 12% as bisexual, and 5% selected the option Other. Although these numbers show that the LGBTQIA+ community is a non-negligible percentage, this group still faces discrimination and prejudice among the majority of cisgender heterosexual men in the gaming industry. For instance, there has been a recent wave of online harassment aimed at female and LGBTQIA+ game designers [23]. Also, the gaming industry has had a persistent history of homophobia, and many LGBTQIA+ employees feel uncomfortable with their jobs because they have to deal directly with homophobic colleagues [41]. As another example, LGBTQIA+ employees who are part of the indie game industry continue to suffer from precarious conditions as they do not have the same support that straight cisgender people have [40].

Low Gender Diversity in Game Jams and Hackathons

Hackathons and game jams offer participants the chance to connect with local or even global communities, in cases of virtual events [12]. These events enable students and professionals to acquire new skills and find new career opportunities. In fact, these events are frequently used as a recruitment strategy by the computing industry. Although hackathons and game jams have been widely adopted to different contexts and flavors, concerns have been raised regarding the lack of gender diversity in these events [8]. This issue can be translated into a loss of opportunity to recruit more women and generate more diversity and gender equity in the area. The average participant profile on these types of events evokes a resemblance with the term *brogrammer*, a portmanteau of “bro” and “programmer” that attempts to represent the knowledge of coding as macho hypermasculine stereotypes [42]. These types of male stereotypes, in combination with stereotypes imposed on the female gender, discourage involvement and a sense of belonging of women in courses related to computer science and STEM in general [4]. In a study on college hackathons, the “hacker culture” and “hardcore ethos” are two of the main reasons that inhibit women from attending these events [48], while in game jams low female attendance is related to a sexist and misogynistic gaming culture [23].

Typical hackathons and game jams have not been successful in significantly increasing women’s participation. As an attempt to change that culture, there are some efforts to change that scenario. In literature, we found reports about hackathons focusing on broader participation to diversify their audience [38] to attract more women and non-binary participants [25, 31] (e.g., StitchFest, Hack Grrrl, T9Hacks). Similarly, women-only game jams [22] have been organized as a way to foster equal participation of women in the game industry. A common issue to address in such events is the so-called competence-confidence gap, which occurs mainly in women who have the competence to perform a task but do not demonstrate the necessary confidence. This phenomenon has been observed in STEM and in collaborative platforms such as GitHub [47]. There is a similar problem related to the confidence of women participants in game jams who feel less confident than men when participating in these events, as we reported in previous work [9].

Recently, companies from different domains have been organizing events in the format of hackathons to recruit new talents [33, 45]. Considering the importance of hackathons and game jams in the learning process beyond classrooms and for the insertion of new professionals in their industries, we are concerned that the lack of women participating in these events may hinder the entry of women into the technology

industry. Motivated by these factors, we attempted to analyze empirical data to understand why women are underrepresented in these events and what are the reasons women are not inspired to participate.

Previous Research on Gender-Related Issues in Hackathons and Game Jams

What we present in this chapter is a consolidation of recommendations, based on different research articles [29, 30, 35] in which we were dedicated to understanding gender-related issues in hackathons and game jams. These studies combined results from quantitative and qualitative research methods. In total, they accumulate 330 responses from surveys and 28 interviews with women and LGBTQIA+ people. Methodological details can be found in the original research articles.

In the work of Paganini et al. [30], we identified some of the main motivators and demotivators for women to participate in hackathons and game jams, collecting survey data from participants of eight events (two game jams and six hackathons) that took place in Recife, Brazil. Among those, one event of each type was focused on women. We also collected data from women who never participated in hackathons, to understand the reasons for that. This research helped gather complementary perspectives of women who have participated in hackathons and game jams as well as women who never joined such events. Women who never participated said they had low confidence in their technical abilities and feared being judged in a predominantly male environment. Another aspect they do not enjoy about these events is the typical format of a weekend-long event. The women who have experienced these events also complained that a predominantly male environment makes them uncomfortable and also generates difficulties in team formation when they have their skills underestimated by men. Also, they complained of not having a voice in their teams and even reported some cases of verbal harassment by men.

In the study performed by Oliveira et al. [29], we investigated the perspective of LGBTQIA+ people that participated in the Global Game Jam (GGJ) 2021. The GGJ is the world's largest game creation event taking place around the world, taking place at a weekend in January since 2009. It involves tens of thousands of participants (jammers) at hundreds of physical and virtual sites in over 100 countries around the world [19]. The purpose of this research was to contribute to gender diversity in game jams and propose some strategies to deal with challenges that LGBTQIA+ may find in game jams. Our

study focused on data collected through a survey and interviews with participants of the GGJ'21 (online due to COVID-19) from Brazilian hubs. Among the interviewees, some participants witnessed situations of homophobia, transphobia, and sexism in other game jams. Participants also highlighted the importance of attending the event with a friend if they did not have any experience yet; there was much hesitation in joining the event alone.

The third study, from Prado et al. [35], focused on difficulties faced by TGNC people when participating in hackathons. We performed a survey and interviews with TGNC people who have participated in hackathons. Their motivations to participate are the typical ones found in hackathon literature, such as learning, networking, and teamwork experience. However, some interviewees reported being a victim of different types of discrimination from the other participants and from the organization staff, mainly because of their gender expression. Part of them does not want to join such events anymore because they are afraid of being victims of LGBTQPhobia again. Survey respondents complained about events not having clear policies on codes of conduct against LGBTQPhobia and gender discrimination.

Recommendations for Gender-Inclusive Events

Some of the intersections in the studies we performed strictly focused on women-inclusive events and others on events that should be more welcoming to LGBTQIA+ people. We merged the recommendations of those studies to generate a set of ten recommendations we believe to be more gender-inclusive, but without a binary man-woman perspective: (1) Start with a gender-inclusive organizing team. (2) Foster inclusive communication. (3) Make safety visible through an explicit code of conduct. (4) Provide equipment to participants and showcase people in the event. (5) Promote events to attract underrepresented genders. (6) Stimulate groups of friends joining together. (7) Introduce elements that underrepresented genders can relate to. (8) Promote learning activities to stimulate both technical and soft skills. (9) Focus more on collaboration and less on competition. (10) Stimulate healthier habits.

- (1) **Start with a gender-inclusive organizing team.** In some events we organized, we observed that the presence of women in the organization can establish a less intimidating environment for female participants creating a sense of belonging for them. We obtained explicit feedback from women regarding the importance

of inviting women to act as mentors and part of the judging panel [30]. The inclusion of staff and especially mentors of different genders and sexual orientations could help bring a sense of belonging [29, 35]. Particularly in the case of TGNC people, their participation is still scarce in these events. Trans-people in the organization would not only assist in creating additional inclusion measures but could also help build a safer space for other TGNC participants [35]. As a positive side effect, specifically inviting trans-professionals as mentors might positively contribute to their professional careers, thus contributing to the inclusion in the technology field as a whole.

- (2) **Foster inclusive communication.** The use of inclusive language (before, during, and after the event) is essential to welcome all audiences [35]. Using neutral language is important especially in languages that have masculine and feminine grammatical genders such as French, German, Spanish, and Portuguese. Advertising can also be inclusive; by focusing on special calls directed to different communities, they can feel included and engaged in registering for the event [9]. It is interesting to include promotional campaigns explaining the opportunities to learn new competencies and meet open-minded people, thus highlighting the typical motivations of learning and networking [13, 30]. In the registration form, allow participants to fill out their preferred names (avoid requesting their registration name) and pronouns. It is important to avoid terms that reduce people to their birth-assigned sex (e.g., male and female) [21]. Rather than having a pre-fixed set of identities, leave open the gender identity field. Provide identification badges in which participants can put their names and chosen pronouns. For online events, make pronouns visible on the participants' profiles. For organizers, make sure that participants, mentors, and staff respect the information that participants provided.
- (3) **Make safety visible through an explicit code of conduct.** Talking more about diversity and providing guidelines on how people can deal with differences is key to a participant's experience in these events. So it is fundamental to have a code of conduct explaining

such aspects and making it visible to every participant since the beginning of the event [29]. A code of conduct must clearly state that sexism, LGBTQPhobia, and other discriminatory attitudes (not only from participants) are not tolerated [35]. Organizers should establish practices that guarantee all participants have an equal voice and have the opportunity to play the role they feel capable of and prevent discrimination. The staff must also verbally reinforce the code of conduct during the event, which would remind that everyone (including mentors and staff) needs to be respectful of each other. In addition, staff should be effective in preventing inappropriate situations and negative attitudes of participants [30]. It is important to create a safe way to denounce bad behavior, so any code of conduct violation can result in expulsion from the event. Such attitudes could ultimately create a safe space for everyone.

- (4) **Provide equipment to participants and showcase people in the event.** LGBTQIA+ people face many social issues, including homelessness, because many people from that community are kicked out or asked to leave the home of their parents, relatives, foster, or group homes because of prejudice toward them [43]. Transgender and gender-diverse people live in financial strain [50], making up a high number of people in many low-wage sectors [17]. In addition to financial difficulties, many members of that group already drop out of higher education for many reasons such as a lack of a sense of belonging, harassment, and violence [16, 44]. The fact of not having proper equipment to join in a hackathon or game jam may refrain them from participating. Although this is almost non-existing practice, organizers could offer proper equipment or propose projects that could be done without using high-performance technologies, so low-income or socially unprivileged people could participate. As technology is a tool to empower minoritized groups such as transgender people [26], showcasing them in the event or individually awarding them for specific tasks (i.e., best pitch, best UI, best back end) can put them in the spotlight so sponsors and eventual recruiters in the

event can notice these people. The participants' talents can be recognized, and they can be invited to selection processes, which would hardly have happened under normal conditions.

- (5) **Promote events to attract underrepresented genders.** This is a general recommendation that has some caveats. Organizing hackathons and game jams focused on gender diversity can be a good strategy to create a sense of belonging for participants of underrepresented genders (e.g., women, TGNC) in these events. This makes them lower their barrier to later participate in typical mixed-gender events. In our experience with gender-focused events, women felt more comfortable participating since one of the obstacles is that an event predominantly composed of men discourages them from joining [30]. This is not only a matter of women-focused hackathons and game jams but events that welcome LGBTQIA+ too. Arguments on that direction were highlighted in one of our studies where one participant cited the name "diversity" in a Global Game Jam hub he joined, which made him feel in a safe space [29]. However, just putting a diversity label in an event is not enough. We identified cases of a woman who did not fit in on a women-focused event and a transphobia case in another women-focused event. It showed us that something more is needed. Such aspects pointed out in the studies we have performed suggest that the other recommendations have to be put into practice to ensure both (i) a sense of belonging and (ii) the creation of a safe space.
- (6) **Stimulate groups of friends joining together.** Many participants mentioned the fun and joy of joining in hackathons and game jams, especially with friends [29, 35]. Although networking is one of the motivations to participate in hackathons [30], shy and introverted people may suffer to fit in. Even in a women-focused event, a woman interviewee abandoned participation in a game jam because she was not comfortable working with a team in which she did not know anyone previously [29]. This can bring a discussion about someone coming to an event with a pre-formed group who might lose networking with others as

this person may focus their interactions with their friends. On the other hand, there is a larger benefit: friends can play a great role in encouraging people who perceive these collaborative events as intimidating or in an unknown setting. However, this recommendation shall not be seen as a rule; otherwise, it would end up excluding people who have no friends who attend hackathons and game jams.

- (7) **Introduce elements that underrepresented genders can relate to.** A way to attract women and the LGBTQIA+ public to hackathons and game jams is to introduce in these events some elements that members of these communities can relate to [29]. The relevance of increasing diversity themes (e.g., gender equity, social justice) in hackathons and games jams can attract more people. In general, social and humanitarian topics generate more empathy. For instance, in humanitarian free open source projects, women and underrepresented groups are more attracted by the opportunity to assist others [34]. An element brought by participants was to bring more visibility to LGBTQIA+ organizers and mentors through specific items (e.g., clothes, badges, pinback buttons, event advertisement digital cards) so that participants who are also part of that community can notice that and have a sense of belonging [29].
- (8) **Promote learning activities to stimulate both technical and soft skills.** Hackathons and game jams are powerful tools for participants to learn and put into practice their technical and soft skills. This is common ground among many participants in these types of event [2, 11, 13, 29, 30, 35]. Students can acquire new competencies and have hands-on experience that complements their academic degree. However, we observed some participants have low confidence in their abilities, and this is also a reason for some people not joining these events [30]. To increase their motivation, it would be interesting to include training workshops and provide resources that participants can study before the events. In addition, the fun and learning aspects should be emphasized during the events [30, 35].

- (9) **Focus more on collaboration and less on competition.** Fostering a competitive or collaborative ambience is a design choice made by organizers [33]. The traditional competitive hackathon format is common when there are incentives such as awards and prizes being offered. A cooperative event can be achieved when social elements are introduced – for example, stimulating participants to pitch project ideas or to wander around the premises and discuss with other teams – thus helping participants from different teams to collaborate and network [32]. Since learning and networking are key motivations for participating, we suggest that organizers could plan events that concentrate less on the competition among teams. Such competitive aspects of game jams and especially hackathons may discourage some people. Of course, having a prize is an important aspect of some events, but we believe that winning should not be promoted as their ultimate goal. Women are stimulated to share their experiences and collaborate with peers, but they expect a welcoming environment. It will also address the fear that many women participants have regarding their performance. Men tend to be more competitive than women in hackathons [30]. In game jams, women [9, 29] and LGBTQIA+ [29] prefer a more collaborative environment, and a spirit of collaboration is important in these events.
- (10) **Stimulate healthier habits.** Another aspect that inhibits participation is the intensive format of events. Weekend events may prevent participants to join because of family commitments [1]. Staying overnight may contribute to low productivity and may be impractical especially for women with children. In fact, we observed that some women feel uncomfortable spending the night at the venue [31]. We suggest alternative schedules to attract more women. It is also important to create a friendly space where everyone feels safe. In addition, the availability of mainly junk food was considered a negative aspect by many participants of our events. Providing healthy food and pauses for fun moments and relaxing during social breaks may create a well-being ambience [30].

As a way to validate these guidelines, we applied them in a hackathon aiming to generate digital solutions in the context of an NGO from Brazil working with socially vulnerable people living with HIV [15]. Part of the material made available [14] exemplifies some of these guidelines (e.g., inclusive communication, elements that underrepresented genders can relate to).

Conclusions

Hackathons and game jams create an environment where informal learning takes place, allowing peers with a multidisciplinary background to share ideas and knowledge. Due to the sexist culture around programming, games, and STEM, in general, the typical behavior of men in these events creates a sometimes unpleasant experience for women and LGBTQIA+ participants and also repels those who never participated. Our studies reveal that hackathons and game jams targeted especially at people from underrepresented genders can create a support network where they feel more confident in their competencies. Nevertheless, mixed-gender events need to prevent the "brogrammer stereotype" by successfully embracing women and LGBTQIA+ in a protective and welcoming environment so they can be more gender-inclusive. Changing the culture at the micro-level of hackathons and game jams can be a starting point for a broader gender-inclusive transformation in the computing area. This transformation is beneficial for everyone, as more diversity means a greater range of visions and experiences, increasing collective creativity toward the resolution of problems. Many future studies can be pursued in this area, such as understanding whether the participation of people from underrepresented gender in the events impacted their academic and professional experiences or exploring other potential benefits that hackathons and game jams may bring for their careers (e.g., building core professional competencies). There are limitations in the studied audiences – restricted to some Brazilian hackathons and game jams – as well as in the findings, which may not be generalized to other countries. However, many of the findings are supported by literature focused on Western countries and may serve as a starting point for other research.

Bibliography

- [1] Craig Anslow, John Brosz, Frank Maurer, and Mike Boyes. Datathons: an experience report of data hackathons for data science education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 615–620, ACM, 2016.
- [2] Ali Arya, Jeff Chastine, Jon Preston, and Allan Fowler. An international study on learning and process choices in the global game jam. *International Journal of Game-Based Learning (IJGBL)*, 3(4):27–46, 2013.
- [3] Reginald A. Blockett. I think it's very much placed on us: Black queer men laboring to forge community at a predominantly white and (hetero) cisnormative research institution. *International Journal of Qualitative Studies in Education*, 30(8):800–816, 2017.
- [4] Sapna Cheryan, Victoria C. Plaut, Paul G. Davies, and Claude M. Steele. Ambient belonging: how stereotypical cues impact gender participation in computer science. *Journal of Personality and Social Psychology*, 97(6):1045, 2009.
- [5] Robin Dembroff. Beyond binary: genderqueer as critical gender kind. *Philosopher's Imprint*, 2019.
- [6] Cristina Díaz-Garía, Angela González-Moreno, and Francisco Jose Sá ez-Martínez. Gender diversity within R&D teams: Its impact on radicalness of innovation. *Innovation*, 15(2):149–160, 2013.
- [7] Alessandra Diehl, Denise Leite Vieira, Marina Milograna Zaneti, Ana Fanganiello, Pratap Sharan, Rebecca Robles, and Jair de Jesus Mari. Social stigma, legal and public health barriers faced by the third gender phenomena in Brazil, India and Mexico: Travestis, hijras and muxes. *International Journal of Social Psychiatry*, 63(5):389–399, 2017.
- [8] Jeanette Falk Olesen and Kim Halskov. 10 years of research with and on hackathons. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, 1073– 1088, 2020.

- [9] Cláudia Ferraz and Kiev Gama. A case study about gender issues in a game jam. In *Proceedings of the International Conference on Game Jams, Hackathons and Game Creation Events 2019*, 1–8, 2019.
- [10] Denae Ford, Reed Milewicz, and Alexander Serebrenik. How remote work can foster a more inclusive environment for transgender developers. In *2019 IEEE/ACM 2nd International Workshop on Gender Equality in Software Engineering (GE)*, 9–12, IEEE, 2019.
- [11] Allan Fowler, Foaad Khosmood, Ali Arya, and Gorm Lai. The global game jam for teaching and learning. In *Proceedings of the 4th Annual Conference on Computing and Information Technology Research and Education New Zealand*, 28–34, sn, 2013.
- [12] Allan Fowler, Johanna Pirker, and Ali Arya. Jamming across borders: An exploratory study. In *International Conference on Game Jams, Hackathons and Game Creation Events 2020*, 16–21, 2020.
- [13] Kiev Gama. Crowdsourced software development in civic apps – motivations of civic hackathons participants. In *International Conference on Enterprise Information Systems*, volume 2, 550–555, SCITEPRESS, 2017.
- [14] Kiev Gama, Lavinia Paganini, Rafa Prado, Claudia Ferraz, Dayanne Coutinho, Wendy Mendes, Gustavo Pinto, and George Valena. Supplementary Material About Hackathon HackaGTP – Book Chapter "Toward more gender-inclusive game jams and hackathons." April 2023.
- [15] Kiev Gama, George Valenca, Candy Estelle Marques Laurendon, Ájò Nasidí Marques, Luis Eduardo Ramos, Ravenna Amaral, Clarissa Barros, and Guilherme Xavier. Hackathons as inclusive spaces for prototyping software in open social innovation with NGOs. In *45th IEEE/ACM International Conference on Software Engineering: Software Engineering in Society, ICSE (SEIS) 2023, Melbourne, Australia*, IEEE, 2023.
- [16] Stacey B. Griner, Cheryl A. Vamos, Erika L. Thompson, Rachel Logan, Coralia Vázquez-Otero, and Ellen M. Daley. The intersection of gender identity and violence: Victimization experienced by transgender college students. *Journal of Interpersonal Violence*, 35(23–24):5704–5725, 2020.

- [17] Amber Hollibaugh and Margot Weiss. Queer precarity and the myth of gay affluence. In *New Labor Forum*, volume 24, 18–27, SAGE Publications, Sage, CA: Los Angeles, CA, 2015.
- [18] Vivian Hunt, Sara Prince, Sundiatu Dixon-Fyle, and Lareina Yee. Delivering through diversity. *McKinsey & Company*, 26:2018, 2018.
- [19] Global Game Jam Inc. Global game jam. <https://globalgamejam.org/>.
- [20] Stefanie K. Johnson. What 11 CEOs have learned about championing diversity. *Harvard Business Review*, 2017.
- [21] Alex Kapitan. The radical copyeditors style guide for writing about transgender people.
- [22] Helen W. Kennedy. Game jam as feminist methodology: The affective labors of intervention in the ludic economy. *Games and Culture*, 13(7):708–727, 2018.
- [23] Aphra Kerr, Joshua D. Savage, and Vicky Twomey-Lee. Decoding and recoding game making events for diversity, inclusion and innovation. 2020.
- [24] Marko Komssi, Danielle Pichlis, Mikko Raatikainen, Klas Kindström, and Janne Järvinen. What are hackathons for? *IEEE Software*, 32(5):60–67, 2014.
- [25] Brittany Ann Kos. Understanding female-focused hackathon participants' collaboration styles and event goals. In *Proceedings of the International Conference on Game Jams, Hackathons and Game Creation Events 2019*, 1–4, 2019.
- [26] Elías Cosenza Krell. Is transmisogyny killing trans women of color? Black trans feminisms and the exigencies of white femininity. *Transgender Studies Quarterly*, 4(2):226–242, 2017.
- [27] Annakaisa Kultima. Defining game jam. In *Foundation of Digital Games (FDG)*, 2015.

- [28] Amanda Menier, Rebecca Zarch, and Stacey Sexton. Broadening gender in computing for transgender and nonbinary learners. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, 1–5, IEEE, 2021.
- [29] Dayanne Oliveira, Rafa Prado, Kiev Gama, and George Valenc, a. An exploratory study on the participation of LGBTQIA+ people in the global game jam 2021. In *Sixth Annual International Conference on Game Jams, Hackathons, and Game Creation Events*, 47–54, 2021.
- [30] Laínia Paganini, Cláudia Ferraz, Kiev Gama, and Carina Alves. Promoting game jams and hackathons as more women-inclusive environments for informal learning. In *2021 IEEE Frontiers in Education Conference (FIE)*, 1–9, IEEE, 2021.
- [31] Lavinia Paganini and Kiev Gama. Engaging women's participation in hackathons: A qualitative study with participants of a female-focused hackathon. In *International Conference on Game Jams, Hackathons and Game Creation Events 2020*, 8–15, 2020.
- [32] Ei Pa Pa Pe-Than, Alexander Nolte, Anna Filippova, Chris Bird, Steve Scallen, and James D. Herbsleb. Corporate hackathons, how and why? A multiple case study of motivation, projects proposal and selection, goal setting, coordination, and outcomes. *Human-Computer Interaction*, 2020.
- [33] Ei Pa Pa Pe-Than, Alexander Nolte, Anna Filippova, Christian Bird, Steve Scallen, and James D Herbsleb. Designing corporate hackathons with a purpose: the future of software development. *IEEE Software*, 36(1):15–22, 2018.
- [34] Lori Postner, Darci Burdge, Stoney Jackson, Heidi Ellis, George Hislop, and Sean Goggins. Using humanitarian free and open source software (HFOSS) to introduce computing for the social good. *ACM SIGCAS Computers and Society*, 45(2):35–35, 2015.
- [35] Rafa Prado, Wendy Mendes Galeno, Kiev S. Gama, and Gustavo Pinto. How trans-inclusive are hackathons? *IEEE Software*, 38(2), 2021.

- [36] Tobias Raun. Interview with Kortney Ryan Ziegler of the Trans* h4ck Project. *Transgender Studies Quarterly (TSQ)*, 1(1-2):280–284, 2014.
- [37] Penny Rheingans, Erica D’Eramo, Crystal Diaz-Espinoza, and Danyelle Ireland. A model for increasing gender diversity in technology. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 459–464, 2018.
- [38] Gabriela T. Richard, Yasmin B. Kafai, Barrie Adleberg, and Orkan Telhan. StitchFest: Diversifying a college hackathon to broaden participation and perceptions in computing. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 114–119, 2015.
- [39] Sheilla L. Rodríguez Madera. From necropraxis to necroresistance: Transgender experiences in Latin America. *Journal of Interpersonal Violence*, 37(11-12):NP9115–NP9143, 2022.
- [40] Bonnie Ruberg. The precarious labor of queer indie game-making: Who benefits from making video games better? *Television & New Media*, 20(8):778–788, 2019.
- [41] Alexander G. Ruiz. Gay in the game industry. *Available at SSRN* 3330812, 2015.
- [42] Anastasia Salter. Code before content? Brogrammer culture in games and electronic literature. *Hyperrhiz: New Media Cultures*, (17), 2017.
- [43] Jama Shelton, Jonah DeChants, Kim Bender, Hsun-Ta Hsu, Diane Santa Maria, Robin Petering, Kristin Ferguson, Sarah Narendorf, and Anamika Barman-Adhikari. Homelessness and housing experiences among LGBTQ young adults in seven us cities. *Cityscape*, 20(3):9–34, 2018.
- [44] Jane G. Stout and Heather M. Wright. Lesbian, gay, bisexual, transgender, and queer students’ sense of belonging in computing: An intersectional approach. *Computing in Science & Engineering*, 18(3):24–30, 2016.

- [45] Gabriela Tapia-González. Educational marketing and hackathon for candidate student recruitment. In *International Conference on Applied Human Factors and Ergonomics*, 431–437, Springer, 2020.
- [46] Kieran Todd, Sarah M. Peitzmeier, Shanna K. Kattari, Michael Miller-Perusse, Akshay Sharma, and Rob Stephenson. Demographic and behavioral profiles of nonbinary and binary transgender youth. *Transgender Health*, 4:254–261, 2019.
- [47] Zhendong Wang, Yi Wang, and David Redmiles. Competence-confidence gap: A threat to female developers' contribution on GitHub. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 81–90, IEEE, 2018.
- [48] Jeremy Warner and Philip J. Guo. Hack.edu: Examining how college hackathons are perceived by student attendees and non-attendees. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, 254–262, 2017.
- [49] Johanna Weststar, Eva Kwan, and Shruti Kumar. Developer satisfaction survey 2014. Summary report. 2019.
- [50] Kenneth J. White, Kim Love, Megan McCoy, Miranda Reiter, Desiree M. Seponski, Janet Koposko, and Erica Regan. Factors associated with the financial strain of transgender and gender diverse college students. *Journal of Consumer Affairs*, 2022.
- [51] Sam Winter, Milton Diamond, Jamison Green, Dan Karasic, Terry Reed, Stephen Whittle, and Kevan Wylie. Transgender people: health at the margins of society. *The Lancet*, 388(10042):390–400, 2016.
- [52] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, and Thomas W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 17

Codes of Conduct in Open Source

Hana Frluckaj, University of Texas Austin, USA.*

James Howison, University of Texas Austin, USA.

Codes of conduct (CoCs) have become a hot topic in open source software as contributors and projects increasingly discuss their messaging, presence, and importance. This chapter aims to provide a holistic overview of CoCs and research on them. We first provide the history, context, and controversies surrounding CoCs and demonstrate why CoCs are an important document and tool for OSS projects. We then showcase findings from the literature on CoCs and finally identify open research questions and call for their exploration.

Introduction of CoCs in OSS

Free/libre open source software (FLOSS/OSS) is an important form of digital infrastructure and technical career pathway for many developers [4, 20]. Despite OSS's origin as an alternative technology movement subverting privatization and commercial forces, OSS's roots in libertarianism, masculine technologies, and techno-determinism have resulted in an apathetic attitude toward social issues [7]. This philosophy has contributed to a notoriously antagonistic environment for women and minorities [14, 16, 23], with CoCs as a particular sore spot.

In 2014, Coraline Ada Ehmke created the Contributor's Covenant (hereafter referred to simply as CC), after reports of sexualized language and assaults at events and a general lack of governance [15]. The CC is the first of many CoCs designed to govern projects and discourage what advocates refer to as “toxic” behavior. To OSS traditionalists, this was interpreted as a threat to free speech. To increasingly

diverse contributors, it was a needed provision of protections and accountability. The introduction of the CC thus began the first major, and ongoing, socio-cultural war of OSS.

Cisgender women and other underrepresented groups often bear the brunt of the labor of advocacy and thankless community-oriented tasks (e.g., documentation and organizing community events) [25]. Women's technical contributions are often undervalued or nitpicked in comparison to men's [17], and women often experience biases and harassment at in-person events. Unsurprisingly, women tend to leave projects earlier than men [13]. While advocacy for CoCs and inclusive efforts has increased over the past few years, there is still pushback among traditionalists who believe the inclusion of any guardrails represents an attack on cis-, white, hetero-men and their free speech; this reaction has even included cases of contributors leaving projects due to the inclusion of a CoC [2]. Having discussed the history of CoCs, we now present their content before turning to examine current research on their role and function in OSS projects.

Structures of Conduct

CoCs are community-wide governance documents that establish the core values, expected behaviors, and commitment to inclusivity within OSS projects. Recently, OSS projects hosted on the popular git control platform GitHub are given CoC templates at the project's creation. The most popular templates include the CC¹ and the Python Code of Conduct² [24]. While CoCs vary in prose and content, they all convey a sense of community values and expectations. We observe different CoC approaches by comparing the six most popular: the Contributor's Covenant, Mozilla's Community Impact Guidelines, Google's Code of Conduct, the Python Code of Conduct, the Django Code of Conduct, and the Rust Code of Conduct.

The most significant CoC is the Contributor's Covenant (CC), which serves as a cornerstone of OSS governance documents. It contains a pledge, standards, enforcement responsibilities and scope, enforcement guidelines, and attribution. The CC pledges against discrimination of any form and is geared toward an “open, welcoming, diverse, inclusive, and healthy community.” It defines unacceptable behavior, namely, any

¹www.contributor-covenant.org/

²www.python.org/psf/conduct/

conduct that could “reasonably be considered inappropriate in a professional setting,” for example, doxxing and harassment. Enforcing these standards is the right and responsibility of community leaders. There are four levels of enforcement depending on the frequency/severity of infringements: initial correction with a written warning, a warning resulting in reduced interactions with others in the community, a temporary ban, and finally a permanent ban.

In the early stages of CoC introduction to OSS, it was primarily larger, successful, and more established OSS projects that were the first to adopt them (e.g., Mozilla and Django). Mozilla’s Community Impact Guidelines³ predates the CC and helped shape it. Mozilla’s CoC is more verbose, with additional clarity, details, and examples. Mozilla showcases a section titled “Be Inclusive,” which encourages people to “seek diverse perspectives” and being open to new perspectives and ideas, fostering innovation. It also lists ways to be considerate toward others, for example, respecting and using someone’s self-identified pronouns. Google’s Code of Conduct⁴ borrows heavily from the CC, albeit terser and more direct. The web framework Django is unique in separating its CoC⁵ from its enforcement manual, the former focusing on community ideals and the latter on enforcement. Django states that while their enforcement manual is internal to their Code of Conduct Committee, it is published in the interest of transparency, a key value of DEI efforts. Finally, the CoC⁶ of the Rust programming language only has two sections: Conduct and Moderation. This structure reflects the CoC’s ultimate purpose – to provide a community-wide baseline for expected behaviors and a warning of the consequences should inappropriate behavior be exhibited.

Related Work on CoCs

To provide additional clarity of the usage of CoCs, we consider existing research and feature results from previous and ongoing work, concluding by highlighting open research questions. Previous research has found that the content of CoCs can be value-based, rule-based, or a mixture of both. Rule-based CoCs list concrete examples of unacceptable behaviors, while value-based codes lack explicit rules and instead

³ www.mozilla.org/en-US/about/governance/policies/participation/

⁴ https://opensource.google/documentation/reference/releasing/template/CODE_OF_CONDUCT

⁵ www.djangoproject.com/conduct/enforcement-manual/

⁶ www.rust-lang.org/policies/code-of-conduct

define community values and ideas generally [24]. All CoCs studied by Tourani et al. championed diversity and a welcoming community and encouraged respectful and constructive collaboration [22, 24]. Finally, for a CoC to be effective, disciplinary actions must be clearly spelt out, and any enforcement should be visible to the community. Public enforcement serves two purposes: potential offenders know there is a consequence to actions, and marginal community members feel safer knowing protections exist with the backing of the broader community [1].

CoC Community Conversations

To demonstrate how communities discuss and use CoCs, we look to a recent publication on the typology of these conversations, where we as researchers viewed thousands of GitHub issues and conducted a content analysis on over 400 OSS community conversations [15].

Adoption and Creation

A CoC's adoption and creation are an important step for communities as they commit to inclusivity and move to a central governance model. The proposal to include a CoC can incite different reactions of disapproval, approval, and ambivalence among community members.

As CoCs grow in popularity, disapproval from the broader project community is becoming increasingly uncommon. Those protesting the inclusion of a CoC often consider it an antithesis to OSS philosophy, claiming it limits free speech and free code (assuming that all code is neutral). Others believe CoCs are an infective moderation tool and are skeptical of their capability to curb negative behavior. Admittedly, communities cannot control the emergence of negative behavior, yet project leaders and advocates can exert authority through their reaction and utilizing CoCs for enforcement. CoCs provide rules that can penalize infringers with unseemly consequences (e.g., temporary or permanent ban).

Most sampled projects had either neutral or positive stances toward a proposed CoC. Positive reactions were relayed through encouraging comments (see Figure 17-1), "+1"s, or "👍"s, yet most CoC proposals received minimal interaction from the community, for example, zero comments/likes. In this case, CoCs were often eventually merged. The lack of engagement from the community could potentially spell trouble for later drama, infringements, or miscommunication.

The screenshot shows a GitHub pull request titled "CODE_OF_CONDUCT.md". The code content is as follows:

```

69 + others.
70 +
71 + In the Lux community we strive to go the extra step to look out for each other.
72 + Don't just aim to be technically unimpeachable, try to be your best self. In

```

Below the code, there are two comments:

- Comment 1:** "I love this line" by a user on Jan 30, 2017. It has 3 upvotes.
- Comment 2:** "I agree. This is one of my favorite parts of the Rust CoC." by another user on Feb 3, 2017.

User roles are indicated next to the comments: "Member" and "Author".

Figure 17-1. Support for the wording in the code of conduct

Moderation and Enforcement

Moderation and enforcement vary between projects and depend on factors such as project leadership structure, presence of a community manager, proportion of contributors to maintainers, etc. We review our typology of enforcement of the CoC [15] through the lens of an online content moderation framework [12], based on content moderation in environments comparable to OSS, for example, Reddit and Wikipedia.

The CoC was used proactively to enforce community guidelines and reactively to moderate unwelcome behaviors. Individual projects decide their moderation schema, derived from several reflective values (e.g., leadership team transparency, driving moderation philosophies, etc.). Common moderated behaviors included the complaints of disgruntled users and offensives due to a language barrier. Upon being “called out” by a moderator, we observed both examples of defiance and apology from the offender (shown in Figure 17-2).



Figure 17-2. Post-moderation, the infringer apologized and corrected their behavior

There are many moderation styles in OSS, for example, human vs. automated moderation. Humans understand nuance and complex situations, while “automated systems offer the kind of moderation required by the massive scale of today’s online community” [12]. There are many OSS bots for detecting non-inclusive language (e.g., `in-solidarity-bot`⁷ and `probot`⁸) in GitHub projects and forums where contributors connect synchronously. A trade-off exists between efficiency and quality of moderation; maintainers and moderators can react quickly to infringements with less careful responses or can spend more time crafting their responses, leaving open the possibility of harm spreading.

OSS projects can be perceived as more transparent than other working settings, but does that apply to moderation? Our analysis suggests CoC conversations exist in private spaces without broader community involvement [15]. There was little public conversation around CoC additions, but longer deliberation when moderation was perceived as unfair, likely due to no shared understanding of the CoC and its usage. This connects with existing work on moderation techniques in online communities,

⁷<https://github.com/jpoehnelt/in-solidarity-bot>

⁸<https://github.com/topics/probot>

for example, work examining the statistical association between types of moderation behaviors and future user activity [8, 11]. We aim to expand the research on CoCs by considering them a form of content moderation, discussed in the next section.

Contributor Experiences with CoCs

CoCs are generally considered helpful in attracting newcomers [18, 21], yet there is little to no empirical evidence to support that. In fact, there is evidence that the CoC's presence has no bearing on a newcomer's joining of a new project compared with other factors considered [9, 19].

Adding nuance to the evidence that CoCs are inconsequential in joining and contribution processes, we present results from two ongoing unpublished studies. The first is a study of trans-contributors and the second a study of OSS maintainers. As part of a larger study, we aimed to understand the experiences of trans-, non-binary, and genderqueer (hereafter referred to simply as "trans") people in OSS and the impact of gender identity on OSS career trajectory. We interviewed 21 trans-contributors to gather details and insights on their involvement in OSS, including their initial contributing experiences, their career trajectories, and positive and negative experiences.

When discussing their early joining experiences in OSS, many of our participants performed reconnaissance work before joining a project. They would screen projects for potential toxicity through a variety of methods, including the presence of a CoC. A participant described a general project culture assessment based on the CoC's presence.

If they've got a code of conduct, they're probably trans-friendly. If they don't, they're probably bigots <laugh>. That's a very rough rule of thumb and there's more nuance, but at a first glance, looking to see if they've got a code of conduct is a very good indicator of how trans-friendly the community will be.

While the presence of the CoC was a positive indicator, our participants also stressed its enforcement. Enforcement of a project's CoC signals to vulnerable contributors whether they are truly welcomed and supported by the broader community [10]. Our participants hoped that communities would enforce community guidelines, rather than simply allowing and ignoring instances of bigotry.

If somebody says something transphobic, that's a warning sign to me, and I need to pay attention to how it's dealt with. Is the community, the project leadership, ...are they laughing with that person or are they scolding and educating that person and correcting the situation? How does that community leadership respond? That response tells me whether or not it's safe for me to be involved in that community.

We also interviewed 21 OSS maintainers to understand how they regard newcomers in their projects. Maintainers were aware of the CoC's message and its role in community management and statements. A participant discussed the CoC's connection to community statements, for example, working to be "actively anti-racist" in their messaging and encoding an actively welcoming mindset among community members.

I'm involved in conversations about code of conduct and, more recently, Black Lives Matter, making sure we actively use anti-racist language in our community and its statements ... The community gets the message reaffirmed that this is an actively inclusive space ... We expect you to make people feel welcome, and we use that explicit language because we really think that it needs the least amount of room for ambiguity.

Another participant spoke passionately about proactively creating an inclusive and welcoming environment for any contributors. They cited the CoC as a document for enforcement against "trolls" and negative behavior. As a maintainer, they expressed a sincere desire to ensure a welcoming environment for everyone and hoped that the CoC would be a meaningful and supporting document for contributors.

Having a visible code of conduct, we're making a commitment to anybody who might want to be involved that you're welcome and we're not going to let anyone else be horrible to you and continue to participate ... I meant it sincerely when I added it. And I hope it means something to someone who's reading it.

Dismissal of CoCs emerged in our interviews, including a participant discussing the potentially performative nature of the CoC. They likened it to greenwashing,⁹ which conveys a false impression about how a company's products are environmentally sound. To this participant, the rules of a CoC were redundant as negative interactions are "of course" discouraged; thus, a CoC makes projects only appear inclusive.

⁹ www.investopedia.com/terms/g/greenwashing.asp

I'm going to be very honest, but for me, this looks like greenwashing, to say, "Oh, you see we're very kind and nice people because that's our standards and we don't accept trolling, insulting, and derogatory comments." I think that's basic. You don't need to say it.

Admittedly, the signal of inclusivity and community provided by CoCs can quickly devolve into a façade if projects do not enforce them. The strength of CoCs derives from community backing of values and project leaders enforcing them. Withholding accountability or allowing negative behavior undermines the authority of the CoC.

Further Recommendations

CoCs are a place to express and define shared community values in OSS projects, an important governance document for moderation, and an indicator of project inclusivity and safety, especially for vulnerable contributors. In this section, we build on these understandings in the form of recommendations for practice in open projects.

For projects creating and adopting a CoC, community consensus is recommended to avoid later confusion and miscommunication. Communities should collaborate in CoC creation and define their values, expected behavior, and consequences of infringements. Creating a custom CoC is a nontrivial task, requiring the input of many members – therein lies its strength. A custom CoC needs collaboration, facilitating shared social learning and values. Projects can also adopt a CoC template and tweak its contents; however, the broader community is encouraged to be involved. If CoC conversations are done privately, common in large projects with hierarchical structures [3], leaders should make an effort to replicate that discussion publicly. A clear community consensus on unacceptable behaviors and their enforcement promotes mutual social learning [6] and can reduce later tension and divisions in the event of a serious infringement. Open collaboration platforms such as GitHub can help by considering ways in which their system designs can better facilitate community discussions at a broader level.

Project leaders should clearly designate a contact method for enforcement within the CoC, either using a moderator's contact info (e.g., the maintainer's email) or creating an email for handling requests (e.g., moderation-team@project.com). As for moderation, our advice is to be firm and fair. Moderators must have a consistent reaction toward the infringer, no matter who they are. To help build shared social learning, we encourage moderators to NOT delete negative interactions and their subsequent moderation.

Moderators should consider adding these (anonymized) examples to their CoCs to maintain a public record of past infringements, providing an opportunity for newcomers to accurately assess the community [5]. We understand, however, that the choice to not delete comments is difficult, especially when members are subject to continuous trolling, harassment, or spamming. In such cases, comments should be deleted, especially if they are targeting a specific individual or group.

We've seen CoCs viewed as a signal of perceived project inclusivity and their enforcement as indicators of the project leadership's commitment to that message. If a project is not committed to enforcement, the CoC's presence may lull contributors into a false sense of safety, especially harmful for vulnerable contributors. Projects should not include a CoC unless they are prepared to enforce it against their "best" contributor. Future research should also include more diverse perspectives (e.g., non-white, cis-male) when considering participant experiences in OSS.

Open Research Questions

Many questions surrounding CoCs remain unanswered. How does the lack of a contact method impact the effectiveness of a CoC? Many negative interactions are also deleted. It is unclear how widespread the deletion of transgressions is among OSS projects nor its long-term implications. Further research can explore the impact of hidden or removed negative interactions on a community's sense of self.

As projects grow and opinions evolve, project leaders can revisit the concerns of the community and reflect these in the CoC. Yet there are still many questions on how best this can be achieved. Should all members participate in CoC discussions or just a representative sample – what would a representative sample look like for a project? How can project leaders best facilitate these discussions in an open and respectful way? When is deliberation on CoC content considered sufficient? What are indicators of a consensus on the contents of the CoC? These are questions to be explored in future research.

There are many different moderation styles and philosophies driving enforcement of the CoC. Should projects be nurturing and educate infringers or be swift in their punishment? Kindness and respect toward others can encourage gracefully accepting constructive feedback and accountability for one's actions, but it may leave the community open to trolling or attacks. Conversely, harsh punishments can effectively discourage infringements from happening again, but run the risk of facilitating a "toxic" and exclusive environment. Studies to help understand the trade-off between

contribution quality and level of activity or the trade-off between the quality of moderation and its efficiency are needed. How can other analyses of moderation values, philosophies, and actions be applied to the decentralized projects of OSS?

We hope that future research can assist communities addressing these questions.

Bibliography

- [1] Aurora, V., & Gardiner, M. (2018). *How to Respond to Code of Conduct Reports. A practical step-by-step guide to handling code of conduct issues*. Frame Shift Consulting.
- [2] Larabel, M. (2018). One Of LLVM's Top Contributors Quits Development Over CoC, Outreach Program. www.fastcompany.com/90250846/codes-of-ethics-probably-dont-work
- [3] Eghbal, N. (2020). *Working in Public: The Making and Maintenance of Open Source Software*. Stripe Press. <https://doi.org/10.1145/3359224>
- [4] Allen, C., & Mehler, D. M. A. (2019). Open science challenges, benefits and tips in early career and beyond. *PLOS Biology*, 17(5), e3000246. <https://doi.org/10.1371/journal.pbio.3000246>
- [5] Bullard, J., & Howison, J. (2015). Learning from Elitist Jerks: Creating high-quality knowledge resources from ongoing conversations. *Journal of the Association for Information Science and Technology*, 66(11), 2267–2276. <https://doi.org/10.1002/asi.23327>
- [6] Chandrasekharan, E., Samory, M., Jhaver, S., Charvat, H., Bruckman, A., Lampe, C., Eisenstein, J., & Gilbert, E. (2018). The Internet's Hidden Rules: An Empirical Study of Reddit Norm Violations at Micro, Meso, and Macro Scales. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW), 1–25. <https://doi.org/10.1145/3274301>
- [7] Dunbar-Hester, C. (2019). *Hacking Diversity: The Politics of Inclusion in Open Technology Cultures*. Princeton University Press. <https://press.princeton.edu/books/hardcover/9780691182070/hacking-diversity>

- [8] Egelman, C. D., Murphy-Hill, E., Kammer, E., Hodges, M. M., Green, C., Jaspan, C., & Lin, J. (2020). Predicting Developers' Negative Feelings about Code Review. *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 174–185.
- [9] Fronchetti, F., Wiese, I., Pinto, G., & Steinmacher, I. (2019). What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity. In F. Bordeleau, A. Sillitti, P. Meirelles, & V. Lenarduzzi (eds.), *Open Source Systems* (Vol. 556, pp. 91–103). Springer International Publishing. https://doi.org/10.1007/978-3-030-20883-7_9
- [10] Haimson, O. L., Gorrell, D., Starks, D. L., & Weinger, Z. (2020). Designing Trans Technology: Defining Challenges and Envisioning Community-Centered Solutions. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13. <https://doi.org/10.1145/3313831.3376669>
- [11] Jhaver, S., Birman, I., Gilbert, E., & Bruckman, A. (2019). Human-Machine Collaboration for Content Regulation: The Case of Reddit Automoderator. *ACM Transactions on Computer-Human Interaction*, 26(5), 1–35. <https://doi.org/10.1145/3338243>
- [12] Jiang, J. A., Nie, P., Brubaker, J. R., & Fiesler, C. (2022). A Trade-off-centered Framework of Content Moderation. <https://doi.org/10.1145/3534929>
- [13] Kuechler, V., Gilbertson, C., & Jensen, C. (2012). Gender Differences in Early Free and Open Source Software Joining Process. In I. Hammouda, B. Lundell, T. Mikkonen, & W. Scacchi (eds.), *Open Source Systems: Long-Term Sustainability* (Vol. 378, pp. 78–93). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33442-9_6
- [14] Lee, A., & Carver, J. C. (2019). FLOSS Participants' Perceptions About Gender and Inclusiveness: A Survey. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 677–687. <https://doi.org/10.1109/ICSE.2019.00077>

- [15] Li, R., Pandurangan, P., Frluckaj, H., & Dabbish, L. (2020). Code of Conduct Conversations in Open Source Software Projects on GitHub. *CSCW*. <https://dl.acm.org/doi/abs/10.1145/3449093>
- [16] Mendez, C., Padala, H. S., Steine-Hanson, Z., Hilderbrand, C., Horvath, A., Hill, C., Simpson, L., Patil, N., Sarma, A., & Burnett, M. (2018). Open source barriers to entry, revisited: A sociotechnical perspective. *Proceedings of the 40th International Conference on Software Engineering*, 1004–1015. <https://doi.org/10.1145/3180155.3180241>
- [17] Nafus, D. (2012). “Patches don’t have gender”: What is not open in open source software. *New Media & Society*, 14(4), 669–683. <https://doi.org/10.1177/1461444811422887>
- [18] Prana, G. A. A., Ford, D., Rastogi, A., Lo, D., Purandare, R., & Nagappan, N. (2021). Including Everyone, Everywhere: Understanding Opportunities and Challenges of Geographic Gender-Inclusion in OSS. *IEEE Transactions on Software Engineering*, 1–1. <https://doi.org/10.1109/TSE.2021.3092813>
- [19] Qiu, H. S., Li, Y. L., Padala, S., Sarma, A., & Vasilescu, B. (2019). The Signals that Potential Contributors Look for When Choosing Open-source Projects. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW), 1–29. <https://doi.org/10.1145/3359224>
- [20] Riehle, D. (2015). How Open Source Is Changing the Software Developer’s Career. *Computer*, 48, 51–57. <https://doi.org/10.1109/MC.2015.132>
- [21] Sholler, D., Steinmacher, I., Ford, D., Averick, M., Hoye, M., & Wilson, G. (2019). Ten simple rules for helping newcomers become contributors to open projects. *PLOS Computational Biology*, 15(9), e1007296. <https://doi.org/10.1371/journal.pcbi.1007296>
- [22] Singh, V., & Brandon, W. (2019). Open Source Software Community Inclusion Initiatives to Support Women Participation. In F. Bordeleau, A. Sillitti, P. Meirelles, & V. Lenarduzzi (eds.), *Open Source Systems* (Vol. 556, pp. 68–79). Springer International Publishing. https://doi.org/10.1007/978-3-030-20883-7_7

- [23] Terrell, J., Kofink, A., Middleton, J., Rainear, C., Murphy-Hill, E., Parnin, C., & Stallings, J. (2017). Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*, 3, e111. <https://doi.org/10.7717/peerj-cs.111>
- [24] Tourani, P., Adams, B., & Serebrenik, A. (2017). Code of conduct in open source projects. *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 24–33. <https://doi.org/10.1109/SANER.2017.7884606>
- [25] Trinkenreich, B. (2020). *Hidden Figures: Roles and Pathways of Successful OSS Contributors | Proceedings of the ACM on Human-Computer Interaction*. <https://dl.acm.org/doi/abs/10.1145/3415251>



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 18

Did Gerrit's Respectful Code Review Reminders Reduce Comment Toxicity?

Emerson Murphy-Hill, Google, USA.*

Jill Dicker, Google, USA.

Delphine Carlson, Google, Germany.

Marian Harbach, Google, Germany.

Ambar Murillo, Google, Germany.

Tao Zhou, Google, Switzerland.

Code reviews are an important part of the software development process for many reasons [3], such as that well-reviewed code increases software quality [8]. But code reviews can sometimes contain toxic, disrespectful, or otherwise contentious discussions that can be stressful on contributors [9, 14]. Such negative interpersonal interactions and rejection disproportionately affect developers from historically marginalized groups. For instance, open source developers who are women have their code reviews rejected more often when they are outsiders to a project and their gender is visible [16]; open source developers who are perceptibly non-White have their reviews rejected more often [11]; and industry developers who are young, White, and male face less pushback than their colleagues [10]. In this chapter, we describe a simple intervention we deployed for two years in the Gerrit code review system designed to reduce negative experiences, as well

as an evaluation of whether the intervention reduced code review comment toxicity. While the intervention did not appear to reduce toxicity in the communities we studied, we recommend future researchers and practitioners study better ways to reliably measure toxicity and more sophisticated interventions.

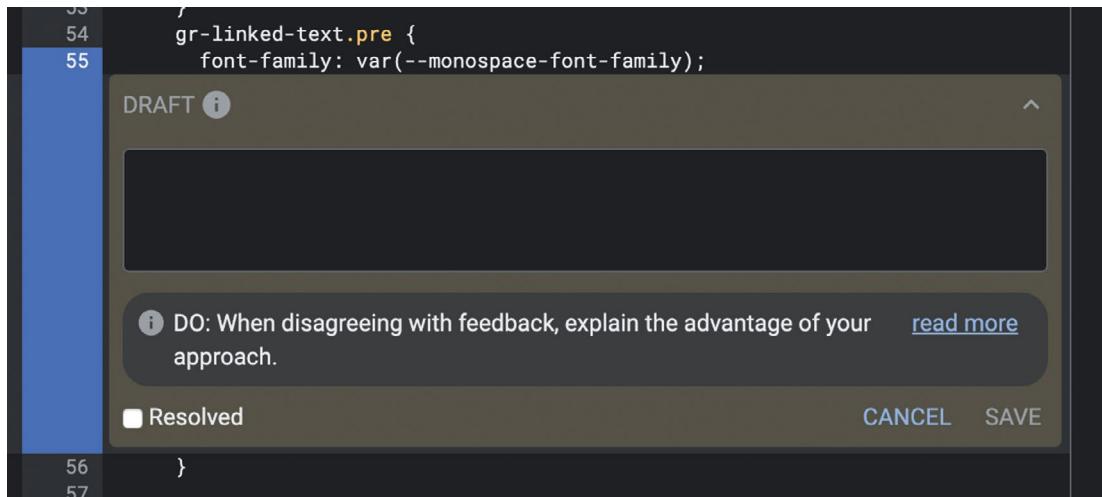


Figure 18-1. An example of a respectful code review reminder in the Gerrit UI

The Design of Respectful Review Reminders

In an effort to reduce the likelihood that negative interpersonal interactions occur, in February 2020, we introduced¹ *respectful code review reminders* in the Gerrit code review system. Our assumption was not that reviewers are intentionally malicious toward code authors, but instead that they have good intentions and simply need to be reminded occasionally to communicate with care.

We designed the reminders to show a text message when a reviewer opens a comment box to provide feedback on a piece of code. Figure 18-1 shows an example. Our design approach for the reminders was based on a blog post about practical advice to engineers for writing respectful code review comments,² and the blog post itself was based in part on our qualitative research on pushback in code review [5]. Using the blog post as a starting point, we (a group of designers, engineers, and researchers) created a

¹<https://gerrit-review.googlesource.com/c/gerrit/+/254957>

²<https://testing.googleblog.com/2019/11/code-health-respectful-reviews-useful.html>

mockup and decided how often we thought reminders should be sent. We designed the reminders with the following intent:

- To improve the likelihood the text was actually read by the developer, we tried to keep it short, with a link with more information. The text of each reminder was one of the following:
 - DO: Assume competence.
 - DO: Provide rationale or context.
 - DO: Consider how comments may be interpreted.
 - DON'T: Criticize the person.
 - DON'T: Use harsh language.
 - DO: Provide specific and actionable feedback.
 - DO: Clearly mark nitpicks and optional comments.
 - DO: Clarify code or reply to the reviewers' comment.
 - DO: When disagreeing with feedback, explain the advantage of your approach.
- So that reviewers would be less likely to become desensitized, whenever a reviewer opened the user interface to leave a comment on the code, the software would randomly decide whether or not to show a message with a probability of 0.3. When a message was shown, the system randomly selected one from the preceding available messages.
- To increase the likelihood that the message was seen, we placed the text directly below where the reviewer would write a comment.
- To reduce the likelihood that the messages would become annoying, reminders initially appeared for each user at most every three days. Some users complained that the feedback was becoming annoying, such as that "I do not think it's respectful of my time to have to dismiss this."³ We thus increased the delay to 14 days.

³<https://bugs.chromium.org/p/gerrit/issues/detail?id=11441#c17>

- To simplify our implementation, messages were shown before the reviewer typed text into the comment box and remained there until the comment was saved. Neither the code being reviewed nor the content of reviewers' comments affected when messages were shown.

In February 2022, the feature was removed from Gerrit, as the analysis described in the following showed a lack of evidence for effectiveness.

Evaluation Approach

We wanted to evaluate whether the reminders had an observable effect on discussions in Gerrit. While there are many potential effects that such reminders might have – such as improvements in sentiment [1, 12], constructiveness [6], or pushback [5] – we made the decision to measure toxicity using Perspective, a public API that returns the output of machine learning models pre-trained on a variety of data, including comments from Wikipedia and the *New York Times*.⁴ We chose to measure this construct with this API because

- Toxicity is a widely used construct in a variety of online contexts that has been used in prior work on analyzing code reviews [13], with mature and scalable analysis tools.
- The Perspective API has relatively good performance. When we applied several natural language processing techniques in prior work, we found that the Perspective API's toxicity scores were the best predictor of human-labeled toxicity in open code review comments [13]. Sarker and colleagues also found that the Perspective API produced the highest F1 score and inter-rater agreement when predicting human-rated toxicity in open source code reviews, including for the Android and Chromium Gerrit projects [15].
- Applying the API is a simple approach, commensurate with the simplicity of the implementation of respectful review reminders.

⁴<https://developers.perspectiveapi.com/s/>

Our research question can be stated as: *Did Gerrit's respectful code review reminders reduce comment toxicity?*

To evaluate the research question, we applied the Perspective API to score toxicity in Gerrit comments. After a project privacy review within Google, designed to ensure that we were following best practices to ensure the privacy of Gerrit's users, we were granted bulk access to the comments in two large projects that use Gerrit for their code review: Chromium⁵ as well as both the internal and open source⁶ repositories of Android. After analyzing comments on these projects with the Perspective API, we examined how the scores changed from before the reminders were deployed in Gerrit to after they were deployed. Since we had no way of knowing if a reminder was shown for each particular comment from the user, we instead assumed that if the deployment of the reminders had an effect on toxicity, that effect would be widely observable rather than only locally observable for certain comments.

After running the Perspective API, to understand and improve the API's labeling accuracy, we randomly sampled comments for manual inspection. The API returns a toxicity score between 0 and 1, representing "how likely it is that a reader would perceive the comment"⁷ as "rude, disrespectful, or unreasonable [and] likely to make people leave a discussion."⁸ However, because the Perspective API model is periodically updated, the precise toxicity scores reported here can change.⁹ Using toxicity scores as strata, one author first inspected 100 comments with scores greater than 0.9, then between 0.8 and 0.9, and so on. The goal was to identify any recurring patterns that represented false positives, so that we could make adjustments and then rerun the toxicity API to better reflect true positives. Such adjustments are typically done in other contexts when natural language processing classifiers are applied to a software engineering context (e.g., [14]).

⁵<https://chromium-review.googlesource.com/>

⁶<https://android-review.googlesource.com/>

⁷<https://developers.perspectiveapi.com/s/about-the-api-score>

⁸<https://developers.perspectiveapi.com/s/about-the-api-attributes-and-languages>

⁹<https://support.perspectiveapi.com/s/about-the-api-faqs>

In comments categorized as having toxicity scores greater than 0.5, we noticed and addressed the following frequent false positives:

- The word *nit* often was rated as highly toxic, yet using the word is considered best practice in code review.¹⁰ We replaced this token with *suggestion* in any comment that contained it, before scoring the comment with the Perspective API.
- The word *remove* often was rated as highly toxic, but appeared not to be in context. We replaced it with *change* before scoring.
- The string `ASSERT` often was rated as highly toxic, but it instead referred to a snippet of code in context. We replaced assert strings with `abc` before scoring.
- The string `CHECK` often was rated as highly toxic, but it instead referred to a snippet of code in context. We replaced it with `ABC` before scoring.

After performing these replacements, we ran the toxicity API and used the resulting toxicity scores in our analysis.

Using the 12 months before the deployment of the reminders as a baseline, we then examined the period after deployment using a sharp regression discontinuity design. Our primary hypothesis was that the reminders had an immediate effect, with a secondary hypothesis that the reminders had a longer-term effect. We also examined whether the effect of the reminders, if any, disappeared after the feature was removed. Given these hypotheses, we dummy-coded the time period into either

- Before (the 12 months before the deployment)
- Short-term (the one week after the deployment)
- Medium-term (between one week and one month after deployment, about three weeks in duration)
- Longer-term (between one month after deployment and the time it was removed, about two years)
- Short-term removal (the one week after removal)

¹⁰<https://google.github.io/eng-practices/review/reviewer/standard.html>

- Medium-term removal (between one week after removal and about two months after removal, about nine weeks in duration)

Given uncertainties in the exact timing of the feature being available to users, we excluded analysis of toxicity scores for two separate one-week periods, one surrounding the estimated rollout date and one surrounding the estimated removal date.

Inspecting percentiles of toxicity scores over time, we observed a noticeable jump in toxicity on July 3, 2020, during the longer-term period. Around this time, we found a sharp increase in reviewers mentioning the acronym LGTM. Because the Perspective API marked it as moderately toxic, we replaced it with the neutral-scoring *looks good to me* in the same way we did with tokens like ASSERT, as mentioned previously. Even after replacement, we noticed an increase in average comment toxicity. After speaking with two people familiar with contemporary community events and inspecting common word frequencies before and after July 3, we were unable to ascertain the cause of the average toxicity change. Nonetheless, we decided to model the shift by dividing the longer-term period into longer-term (A) and (B). Longer-term (A) refers to the period from one month after deployment to July 3, 2020 (about 15 weeks in duration). Longer-term (B) refers to the period from July 3, 2020, to about two years after deployment (about 84 weeks).



Figure 18-2. Toxicity scores during the various study periods

The regression predicted the overall toxicity score for each comment, since this appeared to us to be the simplest way to model toxicity. We included a variety of control variables in our regression. First, we included a random effect for identity of the person who made the comment; the objective here is to control for the typical toxicity that each person exhibits. We also included a variety of fixed effects: the log of the number of revisions in a changelist, using log here to normalize tail skewed distributions; the log of the number of reviewers; the log of the number of inserted lines of code; the log of the number of deleted lines of code; the log of the number of files changed; and whether the change is a reversion of a prior change.

Results

The results of running our regression model are shown in Figure 18-2. The figure shows the toxicity scores that the model estimates, which can be read as mean toxicity scores when accounting for the controls. The whiskers on each bar indicate 95% confidence intervals.

From the figure, we observe the following:

- Given that toxicity scores can range between 0 and 1, toxicity score estimates across all periods remained low – less than 0.1. An example of a common string with a toxicity score of 0.1 is *See comment below*.
- In the short-term, medium-term, and longer-term (A) after the feature was added to Gerrit, toxicity scores increased slightly and statistically significantly.
- In the longer-term (B) period, toxicity scores increased even further, likely due to the July 3, 2020, change.
- No decrease in toxicity was apparent in the removal periods.

Discussion

The preceding evidence suggests that the respectful code review reminders did not reduce the overall level of toxicity in the three Gerrit projects studied. This supports the decision to remove the feature from Gerrit. However, since the problem of negative interpersonal interactions is an observed problem – especially for marginalized developers – other interventions should be considered and evaluated. Such interventions might include

- Context-sensitive suggestions, which activate when comment authors are drafting text that may be interpreted negatively and the tool could provide actionable examples to improve respectfulness of the comment. A domain-specific text analysis tool would be required, however.

- **Feedback on feedback:** Readers could provide their own feedback on comments, perhaps as lightweight as an emoji.¹¹
- **Feedback dashboard:** Users could have a private dashboard that allows them to compare statistics about their comments – for example, overall positive or negative sentiment – and compare those statistics to the larger community. Each user involved in a code review could rate the level of respectfulness of the discussion, after it is merged. This would be reflected in the user dashboard to self-assess and act upon their level of respectfulness.

The evidence here also suggests that the amount of toxicity actually increased after the feature was introduced. If the feature caused the increase, a plausible explanation might be that it induced *reactance* [4] in developers, where people react in opposition to messaging when they view it as a restriction on their choices and behavior. In fact, the regression discontinuity method used here assumes that *defiers* [2], such as the reactant developers here, do not exist in the dataset.

Limitations

There are several limitations to our research. First, based on our manual inspection of comments, it's clear that a significant proportion of comments scored as highly toxic were, in reality, not toxic. While we tried to manually address this by replacing common keywords (e.g., nit) before scoring, other more subtle patterns were not easy to mitigate. For instance, we noticed that self-directed negativity (e.g., I made a stupid mistake) was scored as toxic, as were negative comments directed at the code. On a surface level, these can be interpreted as false positives, but that's not necessarily the case; toxicity directed at the code can also be interpreted as toxicity that reflects on the code's author. General inaccuracy in applying the Perspective API to code review is somewhat expected; it was trained on toxic comments on the open Internet, rather than on code review or software engineering data specifically. A toxicity model trained on code review data specifically would likely yield more accurate results.

Second, the overall effect sizes are very small. While toxicity has shifted statistically significantly, the overall shift may not be practically significant.

¹¹ www.geekwire.com/2016/github-adds-reactions-keep-comments-track/

Third, other changes may confound our results. As we already pointed out, we observed some unexplained change in July 2020, which appears to have had an effect on toxicity scores. As another example, longer-term and later toxicity estimates may be influenced by pandemic-related effects, if any. Other unknown changes may have occurred at various points in time, making it difficult to isolate the causal effect of the politeness feature. Rather than a regression discontinuity design, an A/B testing design could have enabled more confident causal inferences [7], but we did not have access to an A/B testing framework in Gerrit at the time. Moreover, an A/B test in this context may suffer from cross-contamination effects, since toxic comments may beget more toxic comments and vice versa.

Finally, we modeled toxicity by predicting the toxicity of each comment, but other modeling strategies may yield different results. For example, comments could be broken up into sentences; then the toxicity of each sentence could be predicted. As another example, comments could be aggregated by reviewer, and then the average toxicity of the reviewer could be predicted.

Conclusion

The analysis presented in this chapter provides evidence that regularly reminding developers to be respectful in their code review does not measurably reduce comment toxicity. While these results suggest that our specific approach did not have an observable effect, even minor changes – such as decreasing message frequency or using different message text – may yield different results. More sophisticated approaches may also be successful, such as targeted messages to people whose comments may be perceived negatively by their peers. Regardless of the approach taken in the future, researchers should evaluate and report on the effects of such interventions – whether or not an effect is observed. With developers from marginalized groups facing more negative experiences during code review, both in open source [11, 16] and in industry [10], we encourage toolsmiths to design interventions to try to increase equity in code review experiences and outcomes.

Acknowledgments

Thanks to Adam Brown, Alison Chang, Ciera Jaspan, Claire Taylor, Liz Kammer, Rayven Plaza, the Gerrit team, Google's Core Developer team, and our anonymous reviewers for their feedback and support.

Bibliography

- [1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. SentiCR: a customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 106–111, IEEE, 2017.
- [2] Joshua D. Angrist, Guido W. Imbens, and Donald B. Rubin. Identification of causal effects using instrumental variables. *Journal of the American Statistical Association*, 91(434):444– 455, 1996.
- [3] Alberto Bacchelli and Christian Bird. Expectations, outcomes, and challenges of modern code review. In *2013 35th International Conference on Software Engineering (ICSE)*, 712–721, IEEE, 2013.
- [4] Sedona Chinn and P. Sol Hart. Climate change consensus messages cause reactance. *Environmental Communication*, 1–9, 2021.
- [5] Carolyn D. Egelman, Emerson Murphy-Hill, Elizabeth Kammer, Margaret Morrow Hodges, Collin Green, Ciera Jaspan, and James Lin. Predicting developers' negative feelings about code review. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 174–185, IEEE, 2020.
- [6] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Garden, Emerson Murphy-Hill, and Kelly Blincoe. Destructive criticism in software code review impacts inclusion. In *Computer Supported Cooperative Work*, 2022.
- [7] Ron Kohavi, Diane Tang, and Ya Xu. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, 2020.

- [8] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E. Hassan. An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering*, 21(5):2146–2189, 2016.
- [9] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. Did you miss my comment or what? Understanding toxicity in open source discussions. In *In 44th International Conference on Software Engineering (ICSE22)*, 2022.
- [10] Emerson Murphy-Hill, Ciera Jaspan, Carolyn Egelman, and Lan Cheng. The pushback effects of race, ethnicity, gender, and age in code review. *Communications of the ACM*, 65(3):52–57, 2022.
- [11] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. On the relationship between the developers perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering*, 48(8):2955–2968, 2021.
- [12] Rajshakhar Paul, Amiangshu Bosu, and Kazi Zakia Sultana. Expressions of sentiments during code reviews: Male vs. female. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 26–37, IEEE, 2019.
- [13] Huilian Sophie Qiu, Bogdan Vasilescu, Christian Kästner, Carolyn Egelman, Ciera Jaspan, and Emerson Murphy-Hill. Detecting interpersonal conflict in issues and code review: Cross pollinating open-and closed-source approaches. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 41–55, IEEE, 2022.
- [14] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 57–60, 2020.

- [15] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. A benchmark study of the co-temporary toxicity detectors on software engineering interactions. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, 218–227, IEEE, 2020.
- [16] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*, 3:e111, 2017.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 19

Experiences Implementing and Deploying Anonymous Code Review

Jill Dicker, Google, USA.*

Emerson Murphy-Hill, Google, USA.

Anita Sarma, Oregon State University, USA.

Developers from historically marginalized groups often face more rejection and pushback than their peers. To address this problem, at Google we've been experimenting with anonymous author code review (AACR) as a way to combat the biases that all people have. In this chapter, we describe our experience designing, implementing, deploying, and using anonymous author code review over the last few years. We describe the who, how, when, and where to anonymize in a modern code review system. Our experience suggests that the design space for anonymous code review systems is wide and that implementations are not trivial.

Introduction

Prior work [4, 5, 7] has shown that code authors from historically marginalized groups face more rejection and more pushback on their changelists (CLs), also referred to as a *pull request*, than their majority peers, both in open source software and in industry:

- On GitHub, Terrell and colleagues showed that code review acceptance rates differ between men and women, depending on whether reviewers' can infer their gender from their GitHub profiles, that is, based on their username or profile photo. When the gender of

a non-owner of a project is apparent, women have lower pull request acceptance rates than men [7]. When the gender is harder to infer, that trend is reversed – women actually have *higher* acceptance rates.

- Based on race and ethnicity inferred from GitHub users' names, Nadri and colleagues found that perceptibly White developers tended to have higher pull request acceptance rates than non-White developers [5].
- In an industrial setting, we showed that men and White developers have lower odds of receiving pushback from their code reviewers [4]. In that work we also showed that older developers faced more pushback than younger ones.

Such research builds on the observation that it's common for software developers to look at identity signals in a code author's profile [1], such as their name or profile photo. In doing so, human biases likely come into play, resulting in the disparities in rejection and pushback observed in the research literature.

One way that prior work [4, 5] has suggested that such disparities can be mitigated is through anonymous code review, where information about a code change's author is hidden from a reviewer, so that the reviewer may focus on the content of the changelist without being biased by the identity of the code author.

We have built two such anonymous author code review (AACR) features for code review inside of Google. In particular, the first author of this paper, a software engineer at Google, built a Chrome browser extension that provided AACR for those that installed it. The second author of this paper, a research scientist at Google, then evaluated the engineering impacts of this extension, described in a published experiment [4]. Since then, the first author implemented a new version of AACR directly in a code review tool (called Critique), which has been in production for more than two years. The third author of this paper has been studying usage of the Critique AACR implementation as a visiting scientist at Google.

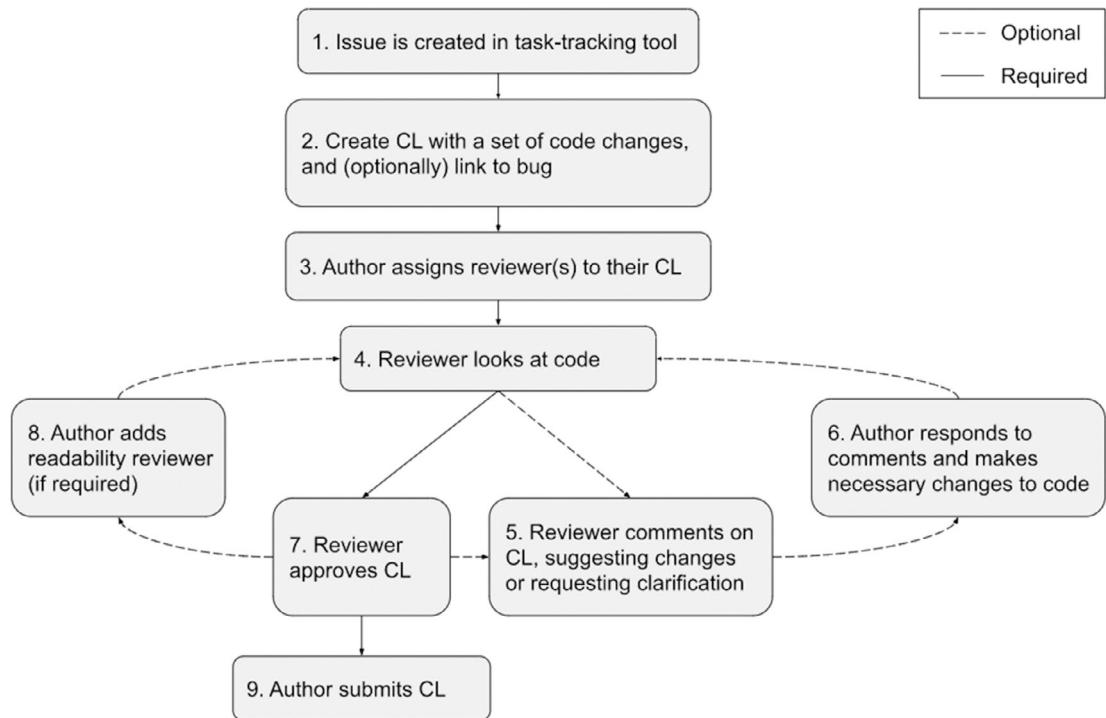


Figure 19-1. Flowchart showing an overview of the code review process at Google [6]

While the idea of AACR seems straightforward – just remove the author’s name during a code review (Figure 19-1) – in this chapter we discuss some practical considerations building and deploying this tool designed to increase equity in code review outcomes. We hope that our experiences will be helpful to toolsmiths and developers in other organizations that wish to implement anonymous code review. We structure the remainder of the chapter as follows: first, deciding who to anonymize; second, deciding how to deploy anonymization; third, deciding when to anonymize; and fourth, deciding where to anonymize.

Who to Anonymize

Anonymous code reviews can take different forms. One option is to anonymize the author identity to mitigate any biases that may emerge based on social cues deciphered from the author profile or identity. The other option is double anonymous. We believe the first option is a better option, since code reviewers act as gatekeepers and already

occupy the power position in this relationship. Further, there is a possibility that reviewer anonymity can lead to more destructive criticism, which is already a concern in open source [2].

How to Deploy

There are different levels at which AACR can be deployed, ranging from the individual to the entire organization. Here we discuss several approaches.

Authors: An AACR option can be authors choose particular changelists or have all their changes be anonymized. We did not choose this option because reviewers may interpret this as authors not trusting them to evaluate their code objectively. Additionally, if people from marginalized communities are the ones who face pushback and turn this option on, this by itself would signal their identity, thereby perpetuating biases.

Reviewers: Another AACR option is to allow reviewers to opt in to reviewing all incoming changelists with the author identity hidden. This has the benefit of avoiding confusion when the author's identity is hidden in a changelist, since the reviewer is aware they opted into this feature. We allowed reviewers to break the glass to reveal the author's identity, in cases where more context is needed. This ensures that AACR does not block a review from moving forward. One drawback with this approach is that reviewers who are aware of unconscious biases in code reviews – and try to combat them – may be more likely to opt in, yet those who do not opt in would arguably benefit more.

Leadership: AACR could also be deployed in a top-down manner, at the behest of leadership. However, our prior experiment suggests that developers tend to prefer to retain control of whether or not they use anonymous code review [3]. Furthermore, our experience has been that there are two “chicken and egg” problems with a top-down approach. First, missing desirable features – such as indications of what time zone an anonymized author is in – tend to be more of a blocker for organizations than for self-selecting individuals. Yet significant feature investments in AACR are less justifiable without an existing, significant user base. Second, organizations tend to want evidence that AACR will solve inclusion problems, but a large user base is necessary to detect statistically significant effects.

Review types: This feature can be enabled for specific types of reviews, rather than being dependent on who the author or reviewer of the changelist is. One type of review that seems appropriate for anonymizing this way are large-scale changelists where an author is making a change across the entire codebase and sends out small reviews to each team whose code was impacted. Examples of large-scale changelists include changing an API method name or upgrading a specific build dependency. Reviewing this type of changelist is much less complex than changelists that are modifying behavior or features, so the identity of the author should not be relevant. Google *readability reviews* are another example of a type of review that is appropriate for always anonymizing the author. These reviews are performed after a changelist has already been approved by a teammate, as shown in step 8 of Figure 19-1, and the review is limited to ensuring the code conforms to language-specific best practices.

When to Anonymize

There are several ways one can implement AACR so that changelists are anonymized and deanonymized at various times.

During review: A common concern raised with anonymizing the author of changelists is that developers will be less familiar with what work their colleagues are doing. To mitigate this issue, we implemented AACR such that as soon as the code is merged into the codebase, at step 9 of Figure 19-1, the identity of the author is no longer anonymized. We had considered going a step further and revealing the identity of the author as soon as the reviewer gave their mark of approval for the change to be merged (step 7 of Figure 19-1), but we found it is a relatively common practice to give approval before all the review comments are resolved. This is especially common when reviews are done across time zones and reviewers don't want to slow down the author.

The screenshot shows a code review interface for a changelist (CL 350658808). At the top, there's a green circular icon with a checkmark and the word "Critique". A search bar says "Search CLs". To the right are icons for search, info, and more. Below the header, the changelist details are shown: "CL 350658808 by" followed by a yellowed-out animal icon (representing an anonymous author), "Unresolved", and a "Reply" button. The changelist summary is "Pilot analysis for the team agreement study on code quality perceptions." Below this, the changelist metadata includes: Reviewers (✓emersonm), CC (ciera, developer-research-eng), Bugs (empty), Workspace (code_quality_team, Open), Created (Thu, Jan 7, 2021, 4:14 PM GMT-8), Last snapshot (Fri, Jan 8, 2021, 11:54 AM GMT-8), and Releases (Rapid Candidates, CL Status). On the right, there are buttons for "Score" (with a question mark icon), "LGTM", "Approval" (both with "[emersonm]"), and "Comments" (Unresolved - 4 comments). At the bottom are "Snapshot CL" and "Modify CL" buttons.

Figure 19-2. Implementation of AACR in the Critique Code Review Tool at Google

Human authors: After releasing this feature, we realized that we had anonymized changelists sent by robot authors. Large-scale refactorings are sometimes done by sending out automatically generated small changes, with the author set as a non-human account. It turned out that looking at the author was one of the primary ways reviewers determined that they were performing a review of a large-scale change. Before we implemented a fix to never anonymize non-human accounts, reviewers who had anonymization on would sometimes even try to communicate with the robot author in the code review tool, unaware it wasn't a real person.

Where to Anonymize

The basic idea behind anonymous author code review is to hide the author's identity from reviewers in the code review tool. Figure 19-2 shows a picture of how we replaced the author's username at the top of a code review with an anonymous animal.

The author of a change request is not only available via the code review tool but also in a suite of tools related to it. Here are some places where the author of the change request is discoverable outside of the code review tool:

- In a task-tracking tool, which is often linked from the changelist
- In a tool that displays the results of running the predefined set of continuous integration tests for each proposed code change, which is always linked from the changelist

- In an extension that notifies developers of any new tasks, code reviews, etc. that are assigned to them
- In emails sent by the code review tool when there are updates to the review

Additionally, within the code review tool, there were places the author's name appeared that we weren't expecting:

- In other changelists in the same chain of changelists
- In the workspace (or branch) name
- As part of the directory structure of the code being modified
- In the code itself

Some of the items mentioned here have a clear best solution. For example, the name of the workspace (or branch) is not relevant to reviewing the content of the code, so can either be omitted entirely or hidden if it contains the author's name. However, many of these items have legitimate reasons to keep the author's name visible. In the following we outline the most contentious of the issues.

Chains of CLs

A relatively common practice within Google is to split code changes into small reviewable pieces and send them for review as a *chain*, or *stack*, of changelists. Each changelist in the stack links to the others in case the reviewer wants to see more context about the full change, as seen in Figure 19-3. If a reviewer of a changelist in such a stack has the author identity hidden, then the author's identity should also be hidden when viewing any other changelists in the stack, to avoid deanonymizing the author when the reviewer is simply trying to get more context for their review. However, if any changelists in the stack are already merged into the codebase, the author's identity will be revealed because anonymization is only applied during review as discussed previously.

Needs attention →							3 CLs < >
Changelist	Author	Status	Last Action	Reviewers	Size	Description	
476420947	[REDACTED]	Pending	11:35 AM by [REDACTED]	[REDACTED]	XS	Enable feature in production.	▼
476420246	[REDACTED]	Pending	11:34 AM by [REDACTED]	[REDACTED]	M	Read user settings using new util method.	▼
476418881	[REDACTED]	Pending	11:34 AM by [REDACTED]	[REDACTED]	XS	Adding a util method to read data.	▼

Figure 19-3. How the code review tool displays a chain of CLs, all written by the same author

Email Notifications

Anonymizing the content: Anytime there is an update to a changelist (e.g., the code is updated by the author or new comments are made in the code review tool), an email summarizing the review activity is sent to the author of the change and all reviewers. We modified the content of the emails to omit the identity of the author where it wasn't necessary and used the generic term "CL author" when it was necessary to refer to the author of the CL. See Figure 19-4 for an example of such an email.

Anonymizing the recipient list: One unintended consequence of allowing individual reviewers to opt in was that a single code review could have a mix of reviewers, some using the anonymization feature and some not. When any summary email is sent out, it is sent as a single email to all reviewers and the author. Since the author is one of the recipients, this would effectively deanonymize the author for any reviewers of the CL. One proposed solution to this problem was to send a separate email to each reviewer and a separate email to the author. However, having a single email thread that includes all reviewers and the author is a feature that many reviewers rely on for communication about the changelist. For example, some users check their email on a mobile device while not at the office and reply to the email as a way to alert everyone else that their review will be delayed. In the two years since this feature was released, there is still no way for users who are reviewing anonymously to view summary emails without risking deanonymizing the author. Our recommendation for users who turn this feature on is that they filter out the email notifications from their inbox, which is not an ideal experience for users, especially those who rely on the email notifications in their workflow.

Identifying author-anonymous reviewers: When a user is first assigned to review a changelist, an email is sent out to all existing reviewers (if any) and the author notifying them that a reviewer has been added. We made the decision to explicitly alert the author

when a reviewer is reviewing their code anonymously for a couple of reasons. First, so the author would be aware that talking to their reviewer outside the code review tool (in person or via chat) would deanonymize the review. Second, so the author wouldn't be confused if the reviewer made a comment that would be considered strange if the reviewer had known the identity of the author. For example, a reviewer who doesn't know the identity of the author might say something like "Will you be able to release this to production next week?", and if the author is a colleague who is going on leave the next day, it might be taken differently than intended. For these reasons, anytime someone is added who will be reviewing a changelist with author identity hidden, a sentence is added to the bottom of the email saying "[user] is reviewing this CL with author identity hidden via Anonymous Code Review[link to documentation].". One unexpected result of adding this text was that it served as an advertisement for AACR. This allowed for feature adoption to grow organically.

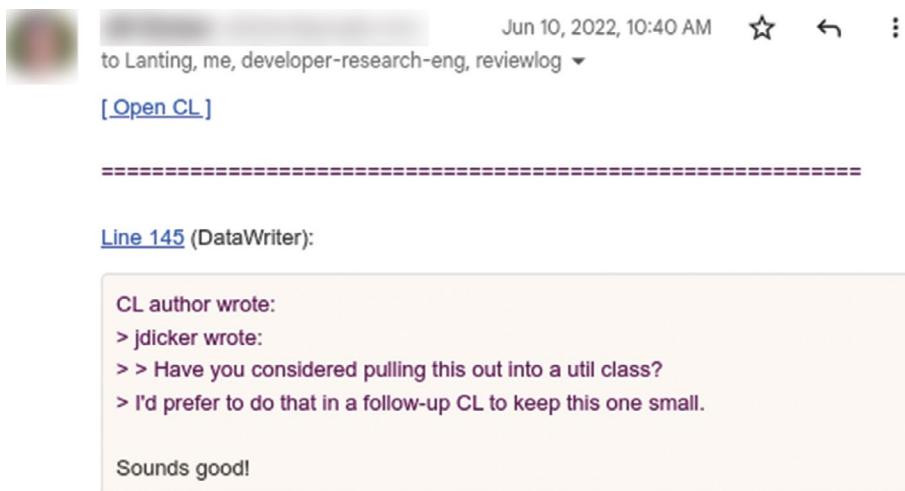


Figure 19-4. A summary email referring to the author as “CL author” to avoid identifying them. Here, Jill Dicker (jdicker) is reviewing the changelist with the author’s identity hidden. For reasons discussed in the main text, the author of the changelist (Lanting) is visible in the email recipient list.

Within the Code

Occasionally the identity of the author appears in the code itself. One common reason this happens is the author leaves a “TODO” in the code that includes their username and a description of what is left to do in a future change. Some less common ways

the author's identity is visible in the code are if the author is adding code to their own personal directory that includes their username or when authors use their own name when writing unit tests. In weighing the benefits of preserving anonymity at the cost of no longer showing the true code that will be submitted into the codebase, we decided the risk and confusion of changing the content of the code was not worth it.

Linked Bug Reports

In many code review systems, it's common to link a change being reviewed with a bug or bugs that are related (Figure 19-1, Step 1). This is useful for a code reviewer to understand more about the code change, especially understanding the rationale for why the change was necessary in the first place. Typically, the author of the change is the same person who is assigned to the linked bug report. This presents a challenge for AACR, since reviewers who look at the linked bug may see the identity of the assignee and reasonably infer that this person is the change's author.

The solution is to anonymize the assignee of the linked bug report. In our implementations, we chose not to do this, mostly due to technical complexity; the code review and the bug systems are independent codebases, so looking up which code reviews are anonymized from the bug tracker is nontrivial. A robust implementation, however, would likely be (a) only anonymize assignees when the bug viewer is also reviewing a linked code review with AACR and (b) perform consistent anonymizations across the code review tool and bug tracker, that is, use the same anonymous animal for each.

Discussion

Designing, implementing, and using anonymous code review has helped us think more deeply about bias in software development tools. A broader question that AACR raises is, what information is relevant and what information is irrelevant when performing software development tasks, and when should irrelevant information be hidden?

From a principled design perspective, by excluding irrelevant information – in this case, a code author's identity – we aim to reduce the influence of bias. From that basic principle, we can examine where else irrelevant information should be removed. One idea here is issue trackers, where participants – issue reporters, triagers, and implementors – work collaboratively to decide if, when, and how a bug should be

fixed or a feature should be implemented. Analogous to code review, it's plausible that people from marginalized identities may face more negative outcomes; for instance, we might predict that issues filed by men would be more likely to be fixed than issues filed by others. For such tools and tasks, anonymization may also be a promising path to reducing the impact of bias and increasing diversity and inclusion.

At the same time, removing identity from software engineering systems is a blunt tool for a more specific problem. That is, identity *per se* isn't the thing that activates people's biases, but rather it's a proxy for demographic identities, such as historically marginalized genders, races, ages, and so on. Because it's difficult to mask only a person's demographic identities, in anonymous code review, we instead mask the entirety of their identity. But this comes with a hidden disadvantage, which is that identity carries useful signals that the anonymization masks. For instance, reviewers who use AACR say they do need to know authors' time zones, so that they can decide when to do a review [3]. But even revealing time zone is potentially fraught. For instance, because Asian authors typically get more pushback than White authors [4], knowing that an author is in Indian Standard Time suggests the author is more likely to be Indian. What we've concluded is that deciding what information to reveal about a developer is more complicated than it appears at first glance.

Another facet of identity that might be useful to reveal during anonymous code review is to indicate whether the author is on the same team as the reviewer. Preference data reveals that most anonymous code reviewers think that this facet would be useful or essential to know, but a nontrivial proportion think that knowing it could be harmful [3]. To us, the lesson here is that user preferences should be gathered and considered, but that other information needs to be taken into account when deciding how to implement anonymization. Indeed, we might ask whether anonymous code reviewers are even in a good position to know whether revealing some facet of identity will be harmful or not. Rather, directly measuring harm through empirical research may be a more fruitful path.

However, our experience has been that even direct measurement isn't a panacea for decision making. For example, while our previous data on code review shows that folks from marginalized groups tend to face disproportionate pushback, it also shows that more junior code authors tend to face more pushback than more senior ones. Based on this result, should we conclude that junior authors are discriminated against? On one hand, it seems plausible that people see a junior developer and assume they're not as competent as a more senior one, independent of the developer's actual performance. On the other hand, senior developers usually do become more competent as they

gain experience, so arguably the data does not suggest the presence of undue bias. So whether a code review system should display an author's seniority remains an open question, even in the presence of data. The lesson that we have drawn from this is that some design decisions should be data-driven and others need not be. A good researcher, designer, or leader should know which is which.

Conclusion

Code review is a process where biases can taint reviewers' judgements, and anonymous author code review is one technique designed to reduce the impact of such biases. Our experience implementing and deploying anonymous author code review has demonstrated that implementing it in a modern code review ecosystem is not as trivial as one might imagine. We hope that the lessons we've shared in this chapter are helpful for others who are considering implementing it in their tools and organizations.

Acknowledgments

Thanks to Ben Holtz, Katie Stolee, Lanting He, the Critique team, Google's Core Developer team, and anonymous reviewers for their feedback and support.

Bibliography

- [1] Denae Ford, Mahnaz Behroozi, Alexander Serebrenik, and Chris Parnin. Beyond the code itself: how programmers really look at pull requests. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 51–60, IEEE, 2019.
- [2] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Garden, Emerson Murphy-Hill, and Kelly Blincoe. Destructive criticism in software code review impacts inclusion. In *Computer Supported Cooperative Work*, 2022. To appear.

- [3] Emerson Murphy-Hill, Jillian Dicker, Margaret Morrow Hodges, Carolyn D. Egelman, Ciera Jaspan, Lan Cheng, Elizabeth Kammer, Ben Holtz, Matthew A. Jorde, Andrea Knight Dolan, et al. Engineering impacts of anonymous author code review: A field experiment. *IEEE Transactions on Software Engineering*, 48(7):2495–2509, 2021.
- [4] Emerson Murphy-Hill, Ciera Jaspan, Carolyn Egelman, and Lan Cheng. The pushback effects of race, ethnicity, gender, and age in code review. *Communications of the ACM*, 65(3):52–57, 2022.
- [5] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. On the relationship between the developers perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering*, 48(8):2955–2968, 2021.
- [6] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. Modern code review: a case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 181–190, 2018.
- [7] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*, 3:e111, 2017.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 20

Mentorship of Women in OSS Projects: A Cross-Disciplinary, Integrative Review¹

Mariska Jacobs, Eindhoven University of Technology, The Netherlands.

Reed Milewicz, Sandia National Laboratories, USA.*

Alexander Serebrenik, Eindhoven University of Technology, The Netherlands.

Mentorship is a relationship in which a more experienced or more knowledgeable person (a mentor) helps guide a less experienced or less knowledgeable person (a mentee); here we draw upon mentor role theory, which holds that mentorship often acts as a vehicle for both career development and psychosocial support, with the end goal of cultivating the mentee's whole self [14]. While mentorship can benefit everyone, studies have shown that positive mentorship experiences are especially significant for members of underrepresented groups. A 2019 report by the National Academies of Sciences, Engineering, and Medicine (NASEM) found that mentorship can be a powerful tool for cultivating professionals from underrepresented groups in STEMM fields [7]; through a close working alliance with a mentor, women and minority mentees can acquire not just the skills they need to succeed but also an affirmation of belonging and professional identity that is so crucial to retention.

In this chapter, we examine the practice of mentorship as a strategy for the retention of women in OSS contexts. Scholarship in recent years has drawn attention

¹The first and second authors contributed equally to this chapter.

to the importance of diversity, equity, and inclusion in growing and sustaining open source software (OSS) communities [SE6]. Along gender lines, women developers are underrepresented in OSS projects and have a higher disengagement rate than men [17]. Studies suggest that a mix of factors such as gender bias [SE15] and a lack of confidence [SE30] affect rates of early disengagement; as Singh and Bongiovanni put it, women in OSS must engage in “vexatious labor” to prove their worth and maintain their place in the community [18]. One interpretation of the data is that women contributors to OSS seek a “sense of belonging” that is held at risk by barriers to inclusion [13, 19, 22].

While mentorship has been shown to be an effective mechanism for onboarding support in OSS development [3, 9, 11, 21], the connection between mentorship practices and diversity and inclusion (D&I) has been understudied in software engineering. Numerous studies outside of software engineering, however, have linked mentorship with diversity and inclusion [5, 6, 7, 8, 10, 24]. Here we see an opportunity to leverage insights from the broader literature on mentorship to help frame the challenges faced by women contributors to OSS and how inclusive mentorship practices may help address those challenges.

To that end, we present findings from a cross-disciplinary review of mentorship among women professionals in OSS contexts. Following the guidelines by Petersen et al. [16], we investigated the challenges women protégées face in OSS and possible strategies that may help overcome those challenges. To further enrich our understanding of these barriers and strategies, we conducted a second mapping study on the challenges faced by women in mentorship in the broader literature outside software engineering. This approach enables us (1) to situate OSS mentorship in the broader context, (2) to compare and contrast the experiences of women professionals across disciplines, and (3) to surface additional strategies for ensuring the retention of women in OSS through mentorship.

Background

There is a growing body of evidence pointing toward mentorship as a key mechanism for onboarding and retention in open source communities, and researchers in recent years have sought to understand the practice of mentorship in OSS – especially as it concerns newcomers. Fagerholm et al. [9] examined how mentoring and project characteristics influence the effectiveness and efficiency of onboarding. Balali et al. [2, 3] and Steinmacher et al. [SE26] have investigated mentorship in OSS, the challenges the mentors and

mentees face, and the strategies they use to navigate those challenges. While formal mentorship programs are less common outside of large open source communities, findings from Feng et al. suggest that informal mentorship relationships between junior and senior contributors are widespread in OSS projects [SE11].

With regard to women in OSS, Balali et al. have called attention to gender-specific challenges faced by women receiving mentorship, including confidence issues, differences in communication styles, and underestimation of their skills by their peers [3]. Work by Ford et al. arrived at similar findings in a study of the Stack Overflow community [12]; the authors found that women are more likely than men to doubt that they have the level of expertise needed to contribute and to feel overwhelmed when competing with a large number of other contributors and recommended mentorship as a strategy for helping address them. Meanwhile, a recent literature survey by Trinkenreich et al. on women's participation in OSS recommends training mentors to better guide women newcomers [23]. Singh et al. highlight a need for more same-gender mentoring between women in OSS and that communities should reward and recognize women who help onboard and mentor others [18].

We note that the potential for mentorship to create a more inclusive climate for women professionals is not unique to OSS. Indeed, while mentorship in software engineering has received increasing attention in recent years, this scholarship represents only a small fraction of the broader mentorship literature (cf., a cross-discipline analysis of mentorship by Lefebvre et al. [15]), including literature on mentorship and diversity and inclusion (D&I); we see opportunities to leverage that wealth of knowledge to inform mentorship practice in our domain. As noted by Wagner et al., for topics that span different disciplines, “the discovery of different methodologies, operationalizations, constructs, or relationships from other disciplines may inspire further research and ultimately broaden and enrich the understanding” of a given topic [25]. For mentorship in particular, Allen and Eby recommend taking a cross-disciplinary view as this allows “researchers and practitioners to obtain a richer and more inclusive perspective of the primary themes of mentoring research and practice” [1].

Methodology

The goal of our work is to identify challenges mentors and women mentees face in OSS contexts and possible strategies that may help overcome those challenges. To do this, we performed a cross-disciplinary twin systematic mapping study. That is, using

established guidelines [16], we conducted two mapping studies (see Figure 20-1): the first targeting OSS mentorship literature and the second targeting comparable literature on mentorship of women in other disciplines – the latter is an offshoot of a broader, ongoing systematic literature review on the meaning of mentorship across disciplines. Using these results, we synthesized a systematic map (Figure 20-2) to provide an overview of the best available evidence on this subject. The following research questions guided the study:

RQ1 What are the challenges that women mentees face in an OSS project? How do these challenges compare to those faced by women mentees in other disciplines?

RQ2 What strategies can help overcome the challenges faced by mentors and women mentees in OSS projects? Are there additional strategies from the broader mentorship literature that could be applicable to OSS contexts?

We opted to use Google Scholar because it provides a single point of access to multiple data sources both inside and outside of software engineering and its support for Boolean expressions. After several rounds of iterative refinement, we selected the following search strings

Search	Query String
(S1) Mentorship of Women in OSS Contexts	mentoring AND (onboarding OR challenges OR barriers) AND (female OR women OR gender) AND (OSS OR "open source software")
(S2) Mentorship in Other Disciplines	mentorship AND (mentorship OR mentoring OR mentor*) AND (characteristics OR qualities OR outcomes OR barriers OR facilitators)

together with the following inclusion (**IC**) and exclusion criteria (**EC**):

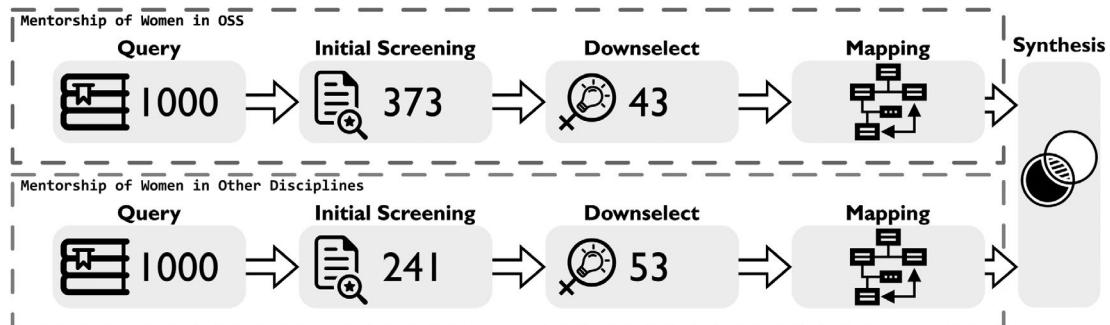


Figure 20-1. An illustration of the methodology used to collect and assess evidence from scholarly literature on the mentorship of women

IC1 The study presents an approach to mentoring women either in an OSS project or similar organizational contexts.

IC2 The study describes challenges or barriers of women in an OSS project or similar organizational contexts.

IC3 The study evaluates mentoring women or newcomers in an OSS project or similar organizational contexts.

IC4 The abstract of a study mentions both women and mentorship, and from the abstract it is clear that the paper can be used to answer one of our research questions.

EC1 The study is not peer-reviewed.

EC2 The study is not written in English.

We executed our search on OSS mentorship on December 14, 2020, and the search on mentorship more broadly on April 10, 2022, each returning the top 1,000 search results mined from Google Scholar. The papers were evaluated following the Google Scholar ranking against the inclusion and exclusion criteria. For each corpus, the authors used open coding techniques to extract concepts and topics surrounding barriers to and strategies for the mentorship of women and constructed a mapping to organize what is known. Of note, when analyzing studies on mentorship outside of software engineering, we coded both claims directly made by studies and claims alluded to in sources cited; for example, [GL22] examines the mentorship of women in policing and cites other relevant sources on the experiences of women in that profession. This allows us to capture the state of practice across different disciplines and their respective bodies of knowledge.

Results

In total, 43 papers met the inclusion criteria for mentorship of women in OSS (S1), and 53 papers met the inclusion criteria for mentorship of women in other professional, non-SE contexts (S2). The papers on OSS mentorship discuss both formal [SE19] and informal mentoring [SE16], mentoring in Apache projects [SE3] and via summer programs such as Google Summer of Code [SE25], mentoring programs designed for women and minority groups such as Outreachy and Rails Girls Summer of Code [SE25], as well as mentoring programs targeting all newcomers. Most of the papers have been published between 2015 and 2020, in venues targeting software engineering (e.g., ICSE, FSE) and collaborative work (e.g., CSCW). The papers in our cross-disciplinary corpus, meanwhile, cover the mentorship of women in diverse professional contexts, including higher education [GL6], primary and secondary school teaching [GL18], executive

boardrooms [GL4], health and fitness [GL15], medicine [GL24], and policing [GL22]. These papers were published between 2007 and 2021 and were spread evenly over a wide range of venues.

The results of the mapping study on these corpora are summarized in Figure 20-2. With respect to the challenges that women mentees face in seeking mentorship, we grouped these into personal, relational, and organizational barriers. With respect to strategies that can help, there are two broad categories: (1) adopting mentorship practices that make mentorship more accessible and effective (e.g., encouraging mentees having multiple mentors) or that may particularly benefit women mentees (e.g., same-gender mentorship) and (2) policies that can be implemented at a project or organizational level to build effective routines for inclusive mentoring (e.g., training and monitoring mentorship progress). A third category (Other Strategies) was added to capture innovative ideas that did not neatly fit into the first two groups. Here, we weave together insights from the OSS literature with comparable evidence on the mentorship of women in other disciplinary contexts in order to provide a richer perspective on the challenges women face and concrete methods for overcoming those challenges.

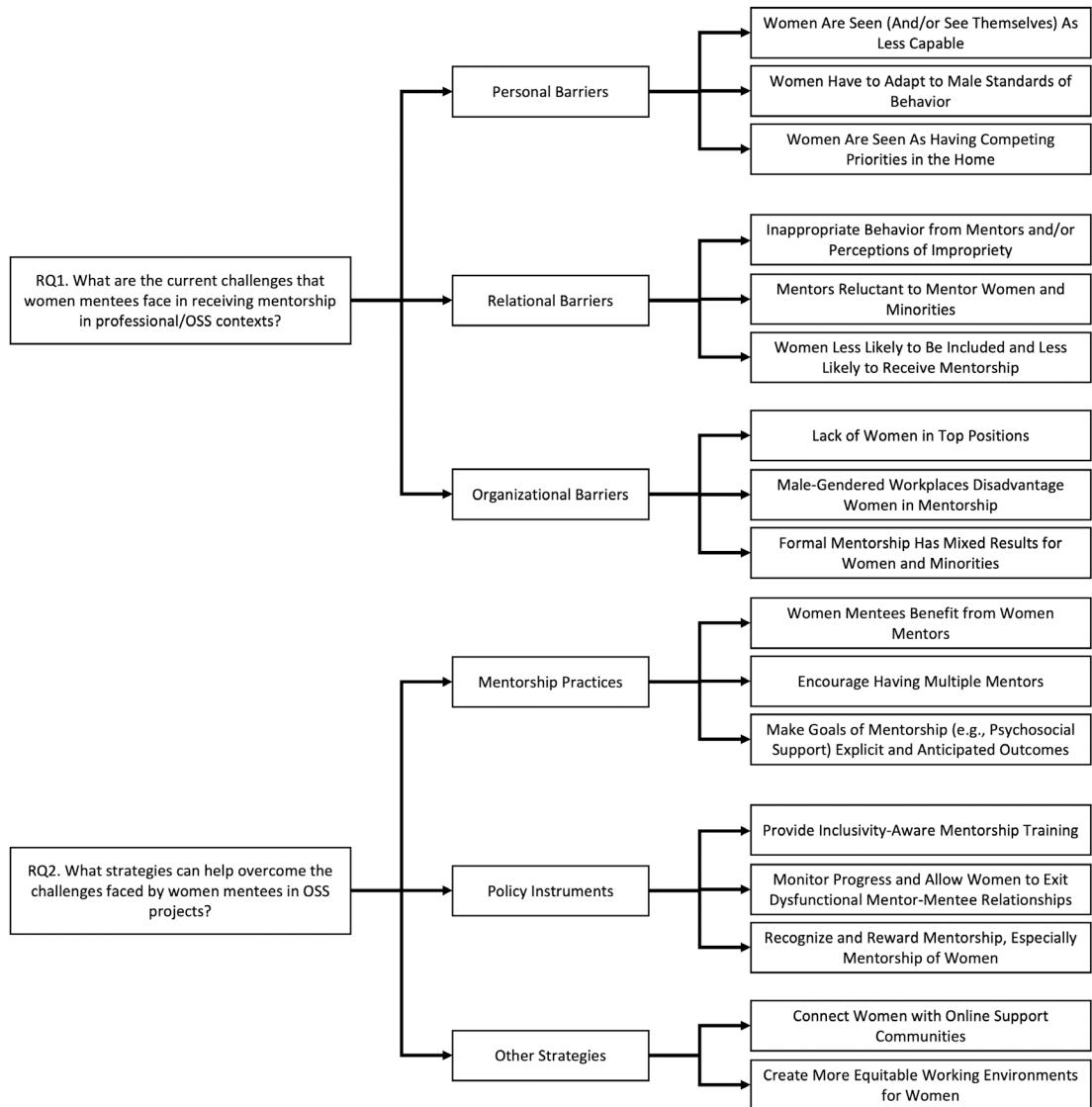


Figure 20-2. Our systematic map of challenges faced by women mentees and strategies to overcome those challenges based on both our cross-disciplinary and OSS-specific corpora. Barriers and strategies that were attested in the OSS literature are labeled.

RQ1. What Are the Challenges That Women Mentees Face in Receiving Mentorship in Professional/OSS Contexts?

(Personal) Women are seen and/or see themselves as less capable: A recurring theme among the studies in our cross-disciplinary corpus is that women are frequently both seen and may see themselves as less capable than men. In male-dominated industries, women are often viewed as outsiders whose competencies are – by default – in question [GL13, GL21]. This is especially true for women who belong to multiple minority groups, such as women of color in higher education, who frequently find themselves “having their authority challenged while having their qualifications and intelligence questioned” [GL5]. As a result, women professionals may be reluctant to seek mentorship for fear of appearing incompetent or in need of help [GL7].

In OSS: Bias against women and negative assumptions about their ability to contribute are a prominent factor driving the underrepresentation of women in OSS [SE15]. Terrell et al. [SE28] showed that the acceptance rate of pull requests of women is higher than that of men when women hide their gender, but lower when they reveal their gender. Women contributors may internalize these negative perceptions; Balali et al. found that women’s sense of self-efficacy is lower than men’s and that women feel less comfortable and less accepted by male counterparts [SE3]. It is worth noting, however, that Bosu and Sultana found that gender bias is not always prominent in OSS projects [SE6]. Studying gender diversity and inclusion in OSS projects through analysis of code review repositories, their results show that three out of the ten projects indicate technical biases against women developers, including lower code acceptance rates as well as delayed feedback during code reviews for women.

(Personal) Women are perceived as competing priorities in the home: Women in the workforce frequently face countervailing demands in the home and at work that can affect how they perceive mentorship. Ghosh and Haynes observe that in many cultures “women are taught to think of their career aspirations to be secondary and to give family priority over work” [GL13]. As an example of this, Leck et al. note that “while male protégés may perceive outcomes of mentoring as valuable (e.g., getting a promotion), women may not because promotions are typically accompanied by increased responsibility which may impinge on other activities women value, such as child care and family time” [GL7]. Mismatches in priorities and expectations around mentorship on both sides can lead to less successful and productive mentor-mentee relationships.

In OSS: We did not find evidence for gendered expectations around work life and home life in OSS contexts as in our cross-disciplinary review. We hypothesize that this may be due to several factors, including the degree and duration of involvement in OSS projects (i.e., women's careers in the workforce play out over their entire life course vs. involvements in OSS projects, which may be more episodic) and/or demographic differences (i.e., with respect to age and family status). There is some evidence, however, to suggest that women may express different motivations to join OSS projects compared with men. Balali et al. found that women contributors were more likely to report joining an OSS project out of excitement rather than a sense of obligation [SE3]. Barcomb studied retaining voluntary episodic contributors in OSS software communities and thereby took gender as one of the moderating effects [SE5]. Although not supported with significance due to a small number of women in the sample size, results suggest large differences between men and women; women had higher path coefficients for contributor benefit motives, social norms, psychological sense of community, and community commitment and a lower path coefficient for satisfaction than men.

(Personal) Women have to adapt to male standards of behavior: As a result of stereotypical gender roles – both those imposed and internalized – women professionals who want to get ahead of their careers often must change how they behave. Jones cites numerous sources that show how women frequently “mask their femininity or behave in masculine ways to deflect unwelcome attention, blend into the system and to achieve success” [GL22]. Case in point, women in many cultures are expected to be more passive and compliant than men, which may make women hesitant to speak up and ask for mentorship [GL2]. Likewise, Fowler notes that women professionals are frequently expected to say “yes” to work responsibilities because of the negative ramifications of saying “no” [GL3]; the author recommends that women mentees may benefit from career advice on how to navigate those conversations. Due to differences in communication styles between men and women, studies suggest that women are more likely to adjust their communication patterns to avoid misinterpretation and miscommunication, including in mentor-mentee contexts [GL1].

In OSS: Studies in OSS contexts have also identified differences in communication styles between men and women and expectations of conduct placed upon women as potential barriers. Imtiaz et al. comment on the “tightrope effect,” where women can only express a limited range of behaviors deemed socially acceptable [SE15]; being too polite or impolite has bigger negative consequences for women than for men, as men are often forgiven for their behavior and sometimes even rewarded for being brash.

Gallus and Bhatia found significant gender differences in people's conversational styles when studying the role of gender in Wikipedia contributions [SE14]; the authors show, however, that differences in communication styles diminish when women reach positions of authority, illustrating how women must change their behaviors as they "climb the ladder." Meanwhile, Balali et al.'s study [SE3] shares reports of women on the differences in communication style and these causing men to come off as creepy.

(Relational) Inappropriate behavior from mentors and/or perceptions of impropriety: Because mentorship traditionally involves a more senior person guiding a less experienced junior, the power dynamics can make mentees vulnerable to mistreatment or exploitation. For example, mentees may experience bullying or incivility from their mentors and not be in a position to push back [GL11]. These issues can be especially pronounced for women mentees. In fields where men outnumber women, mentor-mentee relationships are frequently cross-gender, which can raise suspicions of sexual or romantic impropriety [GL7]; concerns about how the relationship may be perceived may lead to mentors or mentees limiting their one-on-one interactions [GL13]. Of importance to our work, Leck et al. cite findings from multiple sources that online mentoring (or e-mentoring) may reduce the impact of differences of age, gender, and race in mentor-mentee relationships [GL16].

In OSS: In OSS contexts, studies have observed how harassment and sexually charged environments can drive away women participants. Nafus highlights how sexualized online environments include "willful inattention to offensive talk," with one interviewee describing how male developers casually joked about women being raped [SE20]. Lin and den Besten [SE17] demonstrate that, in the open source development, languages or presuppositions or anecdotes cited for strategic communication can reflect certain types of cultures and they dominate the stage/field/platform. Humor is often used to create common ground and gain the attention and interest of people who might have the key to solving a problem. Unfortunately, what looks appealing to some groups will offend others. In a male-dominated environment, male humor dominates the field and tends to exclude consideration of women's needs, values, and preferences. Meanwhile, a 2010 study by Powell et al. found that 50% of women contributors to OSS reported having witnessed gender-based harassment within the OSS community and that 50% experienced harassment themselves – 62% saying that harassment was a deterrent to participation in OSS [SE22]. To the best of our knowledge, the extent to which such project cultures affect the mentorship of women in OSS contexts has not been specifically studied, but the findings in our cross-disciplinary corpus would suggest

that women would be less likely to seek mentorship from men in such environments. In the interviews conducted by Balali et al., participants noted how women mentees would have conversations with women mentors that they probably would not have had with men mentors [SE3]; this can be a problem because the success of mentoring decreases when mentees feel like they cannot share all their concerns with their mentors.

(Relational) Mentors reluctant to mentor women: Several papers in our cross-disciplinary corpus reported mentors being reluctant to support women mentees due to in-group/out-group biases and/or concerns about underperforming mentees harming the social status of their mentors. McDonald and Westphal found that men in corporate executive roles provided relatively more mentorship to men who were first-time directors compared with women and suggested that this may be due to in-group/out-group bias as women may be seen as outsiders or less relatable [GL4] (see also [GL12]). Women mentors, meanwhile, may be judged by how well their mentees perform, leading them to be more selective in whom they mentor; in a study of sports mentorship, Harden et al. observe “female mentors in leadership position are concerned about their reputation if a protégé makes a mistake. Essentially, the female mentor believes this has a negative impact on her ability as a leader” [GL14].

In OSS: In the context of OSS, recent work by Feng et al. found that 93.81% of implicit mentoring interactions in OSS projects were between same-gender pairs and that women were more significantly more likely to participate in cross-gender mentoring than men [SE11]. The authors suggest that unconscious bias and the tendency for people to preferentially interact with others like themselves may explain why men mentors tend to mentor other men.

(Relational) Women less likely to be included and less likely to receive mentorship: In our cross-disciplinary corpus, we identified a recurring theme of women being less likely to be included in workplace activities, limiting their opportunities to find and establish relationships with potential mentors. A review of mentorship in higher education by Fowler finds that “women reported feeling less included in discussions about research, teaching, and promotion and three times less likely to receive career help from colleagues than men [...] Such exclusion, whether intentional or not, is a major barrier to achievement” [GL3] (see also [GL11]). In policing, Jones argues that findings such as these suggest that “compared to their male counterparts, women may have fewer formal and informal opportunities for developing mentoring relationships” [GL22]. According to Kalpazidou and Faber, having fewer opportunities to network and learn from others may be a driving factor in women professionals exiting their careers [GL10].

In OSS: For OSS projects, it is crucial to adequately integrate newcomers into projects, but resources for mentorship can be limited. Balali et al. have observed that a lack of time for mentoring is a significant barrier for mentees to receive mentorship in general [SE3]. Meanwhile, Fiesler et al. note there is a perception that mentorship responsibilities can distract mentors from doing core work on projects [SE12]. Women contributors may be particularly sensitive to insufficient support during onboarding.

(Organizational) Male-gendered environments disadvantage women in mentorship: For the purposes of our work, we follow the definition of male-gendered industries given by Ramaswami et al. [GL21]: industries where either (1) men make up 75% or more of the workforce or (2) the profession is male-stereotyped (e.g., aggressive, competitive, engineering-intensive, etc.). Studies suggest that the culture, norms, and practices of male-gendered professions tend to put “outsiders” (women and minorities) at a disadvantage. In medicine, McNamara et al. argue, “Mentoring challenges faced by women in health care may arise from a clash between social roles and a male-dominated hierarchical medical culture” [GL2]. Similar findings are reported by Kalpazidou and Faber in higher education, where women researchers frequently report a lack of culture fit in academia and a greater sense of isolation relative to men [GL10].

In OSS: The OSS community fits with the definition of a male-gendered working environment. Not only do women make up a small minority of OSS contributors, the disparities are especially pronounced among top contributors; Wang et al. found that only 3% of the top 5,000 developers on GitHub are women [SE30]. Using our cross-disciplinary corpus as a guide, this hints at a possible culture mismatch for women in OSS and that women participants would become attuned to these issues. Indeed, Prana et al. found that women place greater importance on the social aspects of participating in OSS projects relative to men [SE23]; according to their study, women value selecting a project that has friends and colleagues on it more so than men (64% of women vs. 25% of men), and more women find having a shared gender identity with fellow contributors to be important compared with men (37% of women vs. 1% of men). That is, women are more likely to seek social signals that indicate that a project will make room for them and others like themselves. Gender-related biases in male-gendered environments can also manifest in subtle ways through work practices and tools. For example, Mendez et al. [SE18] found that tools and infrastructure in OSS projects are implicated with causing gender biases. Their study showed that in 88% of the barriers, self-efficacy was identified; that is, for women newcomers with a lower sense of self-efficacy, tool issues can further erode their confidence. A follow-up study by Padala et al. [SE21] found that even when

tool biases affect both men and women, women can be disproportionately affected: for example, a lack of support for cognitive diversity in OSS tools disadvantaged women significantly more than men.

(Organizational) Lack of women in top positions: A common problem in male-dominated industries is the concept of the “old boys’ network,” where men in power preferentially hand down opportunities to other men, to the exclusion of women. In the context of primary and secondary education, Peters illustrates how “seasoned professionals (typically White males) have sought to assist protégés who are ‘younger versions of themselves’ [...] As women have entered school leadership, they have experienced limited access to productive mentoring relationships, further limiting their access to school leadership positions” [GL18]. The lack of women at the top can become a self-fulfilling prophecy, as men mentors in leadership may associate gender with performance potential and the women mentors are unavailable to help lift other women up [GL6, GL7, GL13, GL21]. Meanwhile, women who do succeed and become leaders often find themselves overburdened by mentorship relative to their male peers. For example, Corneille et al. note how senior women of color in higher education may feel “an obligation to serve to honor the legacy of those who provided mentorship throughout previous generations” [GL8]; similarly in sports Bower describes how “the demand for female mentors creates a problem due to the shortage of women in the upper levels of the organization. When there is a shortage of women at upper levels of management, women in these positions are overburdened with women needing mentors” [GL17] (see also [GL23]).

In OSS: In a study of over 700 OSS projects, Canedo et al. found that while 5.35% of contributors were women, women were relatively underrepresented in top positions on projects, making up only 2.30% of the total number of core developers [SE8] (cf., Wang et al.[SE30]). Of note, the authors reported finding no differences in the work activities of core developers along gender lines. According to Steinmacher et al., this extends mentorship for developers more generally: men and women are equally likely to perform mentorship in OSS projects [SE26]. So while men and women senior developers may perform mentorship at similar rates, there are relatively few senior women available to offer mentorship in OSS projects. It is unclear, however, whether this actually translates into greater workloads for women; Feng et al. found that women provided more mentorship than men, but the effect size was small [SE11].

(Organizational) Formal mentorship can fail to provide the support that women need:

Even when organizations do recognize the value of mentorship and set up formal structures to encourage it, this does not guarantee that women will get the support that they need. In a study of mentorship of women in a medical residency program, McNamara et al. found that women's relationships with assigned mentors tended to be transient and impersonal and that arbitrarily matching mentors to mentees did not go far enough in encouraging beneficial mentoring relationships [GL2]; in that study, the best mentor-mentee relationships grew organically from informal relationships. Researchers caution, however, that informal mentorship is not a panacea, as it tends to reproduce existing organizational and cultural barriers [GL10]. This also tracks with sources cited earlier showing how women tend to have less access to mentorship than men – having only informal mentorship channels may mean women receive no mentorship at all.

In OSS: Some OSS projects, particularly high-profile ones, have specialized programs with mentors' selection criteria to identify mentors who are a good fit for the students [SE25], but formal mentorship practices are not widespread across projects. Literature suggests that OSS mentors are not usually formally trained and either voluntarily elect to mentor out of personal interest or are asked to do so by the community [SE4]. This could be a newcomer asking for help from a senior developer [SE10] or core members commenting on issues of newcomers, which is comparable to mentoring [SE24]. Relationships are then established between a more experienced developer and a newcomer after having had a couple of interchanges of email, leading up to exchanging personal messages on other platforms [SE9]. Kariri and Rodríguez [SE16] studied mentoring in the Stack Overflow community. Mentoring here also takes place informally; people help other people out by answering questions people post without being obliged to do so.

RQ2. What Strategies Can Help Overcome the Challenges Faced by Women Mentees in OSS Projects?

(Mentorship Practices) Women mentees benefit from women mentors: Women benefit from having mentors of all genders. That being said, there are particular benefits in matching women mentees with same-sex mentors. Leck et al. cite numerous other works that show how “compared to cross-gender relationships, single-gender dyads alleviate the difficulty of mirroring ‘male behaviours,’ increase interpersonal comfort, and provide more psychosocial and career development support” [GL7, GL16]; we note

that Høigaard and Mathiesen ([GL9]) give a contrary result and did not find a significant difference between same-gender and cross-gender mentorships in their study in terms of psychosocial support or role modeling ($n=36$), which they attribute to all mentors in their study receiving training on mentorship and communication. Women lifting up other women may help break the cycle of women being underrepresented in OSS project leadership. Bosu and Sultana note that promoting and mentoring women to leadership positions as an effective solution to foster gender diversity in OSS [SE6].

(Mentorship Practices) Encourage having multiple mentors: Hybrid models of mentorship such as mentor networks, peer mentoring, and group mentoring can lessen the problem of not having enough women mentors to match to women mentees [GL13]. Moreover, Smith-Jentsch et al. remark that an advantage of online mentoring (as is the case for OSS) is that it is easier for mentees to enter relationships with a diverse network of mentors [GL1]. This diversity of mentors can span both personal backgrounds and skillsets; reporting on the needs of mentorship of black women in academia, Evans and Cokley explain how “a person of a different culture or sex may provide excellent research mentorship, but may not be able to sufficiently attend to African American women’s research interest or unique needs. For example, a primary mentor may be chosen for his or her research skills whereas a secondary mentor may be chosen to foster ethnic minority or sex/gender research and career interests” [GL5]. Likewise, for women contributors to OSS, having access to different mentors may help provide a stronger support network while also reducing the burden placed on individual mentors [SE12].

(Mentorship Practices) Make goals of mentorship (e.g., psychosocial support) explicit and anticipated outcomes: In general, having clear goals in mentorship helps ensure that the relationship is productive and impactful. For example, Trainer et al. found that task definition is an important design consideration in mentorship schemes [SE29]. Case in point, mentees working on user-facing, interdependent software develop both technical skills and social ties. According to the authors, it is important to structure mentees’ tasks so that can yield important learning and interpersonal benefits for the mentees; identifying what goals mentees have up front is key to matching them with tasks that advance those goals. Mentorship, however, goes beyond just skill development and career support but also encompasses psychosocial support, and this may be especially important for retention of women mentees [GL3, GL7]. Findings by Smith-Jentsch et al. indicate that, all things being equal, psychosocial support tends to be less frequent in e-mentoring contexts [GL1]; for this reason, Leck et al. argue that mentors in e-mentoring contexts should be explicitly reminded that adequate psychosocial support is a necessary function of mentorship [GL16]. OSS projects that want to encourage

mentorship of women should be up-front about the purpose and intended outcomes of that mentorship and should think holistically about both career and psychosocial support aspects.

(Policy Instruments) Provide inclusivity-aware mentorship training: Mentorship programs should be geared toward inclusion through intentional training and guidance to raise awareness of the needs of women and minorities [GL3, GL5, GL19, GL20]. Studies indicate that having inclusive environments is significantly more important to women than men [GL7, GL15]. With the understanding that women may be less likely to seek mentorship (e.g., for fear of being seen as less capable), McNamara et al. argue that “[women mentees] should be educated about the importance of mentorship, given various strategies to initiate contact with potential mentors, and encouraged to actively seek and maintain their mentoring relationships” [GL2]. Likewise, as we mentioned earlier, gender-related differences in communication can create barriers for newcomers to OSS. Intentionally discussing these issues and challenging gendered humor [SE17] may promote a more inclusive environment. Brainstorming can also support minority team members’ engagement and satisfaction [SE2, SE13] by providing structure for all team members to voice ideas and by encouraging integration of every contribution.

(Policy Instruments) Monitor progress and allow women to exit dysfunctional mentor-mentee relationships: To minimize negative outcomes, mentoring programs should monitor the progress of mentees and enable them to exit from a mentor-mentee relationship that is not a good fit for them, and this may be especially important for online mentorship [GL1, GL7, GL16]; doing so also has the added benefit of creating opportunities to gather data and improve upon the mentorship program. Silva et al. [SE25] show that OSS projects can adopt mentoring coordination actions, such as monitoring mentors’ activities as a strategy to help the mentors with the challenges they face and reduce the odds of failure. While this can benefit all mentees during onboarding, it is significant for women contributors who may otherwise feel uncomfortable raising their concerns over a poor mentor-mentee relationship.

(Policy Instruments) Recognize and reward mentorship, especially mentorship of women: Mentorship requires a substantial investment of time and effort and should be rewarded accordingly [GL10]. Corneille et al. recommend highlighting mentorship success stories on institutional websites and that funding agencies should create incentives to encourage mentorship of women and minorities [GL8]. Other authors have suggested making space for mentorship by decreasing mentors’ workloads elsewhere, signaling that building mentor-mentee relationships is just as important as any other

activity [GL5, GL23]. That is, mentorship needs to be treated as a first-class contribution to projects, and mentorship of women and minorities should be especially prioritized.

(Other Strategies) Connect women with online support communities: Joining an online community can be a means to overcome the challenges faced by female software developers in an OSS project; for example, FLOSScoach is a community that helps increase the self-efficacy of newcomers in an OSS project [SE27]. Another study by Abanoz shows that social support mechanisms can provide the necessary support for individuals with the activities they carry out and they create the necessary environment for individuals' motivations [SE1]. In this way, they enable individuals to perceive themselves in a better situation and perform actions in line with possible selves, improving the low self-efficacy of women software developers that is mentioned as a challenge in OSS projects.

(Other Strategies) Create more equitable working environments for women: Related to gender bias in tools, the GenderMag method proposes a solution [SE7]; GenderMag, short for Gender Inclusiveness Magnifier, is a method for identifying gender inclusiveness barriers in software and generating ideas to fix those barriers. Padala et al. [SE21] used this technique with OSS teams to improve newcomer experiences in OSS projects.

Discussion

The goal of this study was to investigate the mentorship of women in OSS projects to support the onboarding process to prevent early disengagement of women in those projects. We observe that women in OSS face a variety of personal, relational, and organizational barriers to receiving mentorship. By situating the SE literature on this subject within a broader, cross-disciplinary context, we see these experiences are comparable to women professionals in other contexts.

By establishing that mentorship and women's experiences of mentorship are comparable, this positions us to leverage theoretical insights and potential solutions from the broader literature. There are numerous approaches backed by evidence that can help ensure that women in OSS receive the mentorship they need, including adopting better mentorship practices, using policy instruments to promote a culture of mentorship excellence, and other strategies that increase women's sense of belonging in the OSS community. This both reinforces the recommendations of SE researchers

(e.g., matching women mentees with women mentors) and also highlights strategies that have been relatively understudied in our field (e.g., mentorship training programs and reward mechanisms for mentorship).

As noted by Stoeger et al., a key issue in the design of mentorship programs is a lack of awareness of the state-of-the-art within mentoring research [20]; we believe that acquainting SE practitioners with insights from the broader literature may be helpful. As for SE researchers, we believe that the wealth of scholarly resources on mentorship in other domains presents opportunities for replication studies and other forms of cross-disciplinary dialogue.

As with any mapping study, our work focuses on providing a broad overview of the subject matter rather than an exhaustive and in-depth treatment of the literature; there are many more studies available to us, including more recent works that were not in scope at the time we carried out our search on the SE literature. Moreover, as this was a cross-disciplinary review, we faced limitations due to a lack of institutional access to certain journals and proceedings outside of our domain. We do believe, however, that we achieved sufficient coverage of the relevant literature to be able to tell an accurate story of women's experiences in mentorship. We also note that neither mentorship nor the experiences of women professionals are entirely the same across disciplines. The literature we drew upon comes to us from multiple, distinct traditions of mentorship research and practice. We do argue, however, that the findings we present from our two corpora are complementary even if they are not perfectly comparable.

Conclusion

Mentorship is a widespread professional practice across numerous domains for developing talent and fostering enduring communities of practice and one that deserves special attention in meeting the personal and professional needs of members of underrepresented groups. In this chapter, we presented initial findings from a systematic mapping study on mentoring women in OSS projects. Using sources both from within SE research on OSS and mentorship literature in other fields, we identified challenges experienced by women mentees as well as strategies that can be used to overcome these challenges. We hope that our work helps enrich the discussion on how mentorship can play an instrumental role in promoting diversity, equity, and inclusion in software development practice.

Acknowledgments

Sandia National Laboratories is a multimission laboratory managed and operated by the National Technology and Engineering Solutions of Sandia, LLC (NT-ESS), a wholly owned subsidiary of Honeywell International, Inc., for the US Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Images are used by permission. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of NT-ESS, the US Department of Energy, or the US Government. SAND2023-06679B.

This paper uses icons from the Noun Project by artists Adrien Coquet, Binpodo, Eucalyp, Numero Uno, and Stephen Plaster.

Table 20-1. Selection of sources from literature review on mentorship of women in the general (non-SE) literature

Label	Reference
GL1	KA Smith-Jentsch, SA Scielzo, and CS Yarbrough. A comparison of face-to-face and electronic peer-mentoring: Interactions with mentor gender. Elsevier, 2008
GL2	MC McNamara, MA McNeil, and J Chang. A pilot study exploring gender differences in residents' strategies for establishing mentoring relationships. In <i>Medical Education Online</i> . Taylor & Francis, 2008
GL3	JL Fowler. Academics at work: mentoring in research, teaching, and service. In <i>International Journal for Academic Development</i> . Taylor & Francis, 2017
GL4	ML McDonald and JD Westphal. Access denied: Low mentoring of women and minority first-time directors and its negative effects on appointments to additional boards. In <i>Academy of Management Journal</i> . journals.aom.org , 2013
GL5	GL Evans and KO Cokley. African American women and the academy: Using career mentoring to increase research productivity. psycnet.apa.org , 2008
GL6	A Allen and BR Butler. African American women faculty: Towards a model of coethnic mentorship in the academe. In <i>Journal of Progressive Policy & Practice</i> . caarpweb.org , 2014

(continued)

Table 20-1. (continued)

Label	Reference
GL7	J Leck, B Orser, and A Riding. An examination of gender influences in career mentoring. Wiley Online Library, 2009
GL8	M Corneille, A Lee, S Allen, and J Cannady. Barriers to the advancement of women of color faculty in stem: The need for promoting equity using an intersectional framework. emerald.com , 2019
GL9	R Høigaard and P Mathisen. Benefits of formal mentoring for female leaders. radar.brookes.ac.uk , 2009
GL10	E Kalpazidou Schmidt and ST Faber. Benefits of peer mentoring to mentors, female mentees and higher education institutions. Taylor & Francis, 2016
GL11	M Cross, S Lee, H Bridgman, DK Thapa, and M Cleary. Benefits, barriers and enablers of mentoring female health academics: an integrative review. journals.plos.org , 2019
GL12	L Searby, J Ballenger, and J Tripses. Climbing the ladder, holding the ladder: The mentoring experiences of higher education female leaders. awl-ojs-tamu.tdl.org , 2015
GL13	R Ghosh and RK Haynes. Cross gender mentoring in the era of globalization: Implications for mentoring the organizational women of India. In <i>Online Submission</i> . ERIC, 2008
GL14	SL Harden, RA Clark, and WB Johnson. Cross-gender mentorship in clinical psychology doctoral programs: an exploratory survey study. Taylor & Francis, 2009
GL15	GG Bower. Developing effective mentoring relationships with women in the health and fitness industry: Suggestions from the perspective of the protégé. journals.humankinetics.com , 2008
GL16	JD Leck, C Elliott, and B Rockwell. E-mentoring women: Lessons learned from a pilot program. clutejournals.com , 2012
GL17	GG Bower. Effective mentoring relationships with women in sport: Results of a meta-ethnography. In <i>Advancing Women in Leadership Journal</i> . awl-ojs-tamu.tdl.org , 2009

(continued)

Table 20-1. (continued)

Label	Reference
GL18	A Peters. Elements of successful mentoring of a female school leader. In <i>Leadership and Policy in Schools</i> . Taylor & Francis, 2010
GL19	J Park, J Park, A Williams, and AL Morse. Exploring the roles of mentoring relationship on female student-athletes' career development. csri-jia.org , 2017
GL20	E Petridou. E-mentoring women entrepreneurs: discussing participants' reactions. In <i>Gender in Management: An International Journal</i> . emerald.com , 2009
GL21	A Ramaswami, GF Dreher, and R Bretz. Gender, mentoring, and career success: The importance of organizational context. Wiley Online Library, 2010
GL22	J Jones. How can mentoring support women in a male-dominated workplace? A case study of the UK police force. In <i>Palgrave Communications</i> . nature.com , 2017
GL23	SJ Potter, E Abrams, and L Townson. Mentoring undergraduate researchers: Faculty mentors perceptions of the challenges and benefits of the research relationship. clutejournals.com , 2009
GL24	EA Faucett, HC McCrary, and T Milinic. The role of same-sex mentorship and organizational support in encouraging women to pursue surgery. Elsevier, 2017

Table 20-2. Selection of sources from literature review on mentorship of women in OSS from the SE literature

Label	Reference
SE1	Enes Abanoz. Code: A tool to repair gender gap in digital age. In <i>Teaching and Learning Practices That Promote Sustainable Development and Active Citizenship</i> , pages 254–275. IGI Global, 2020
SE2	Cat Allman. The human factor in open source. <i>Open Source Business Resource</i> , 2009
SE3	Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurelio Gerosa. Newcomers' barriers ...is that all? An analysis of mentors' and newcomers' barriers in OSS projects. <i>CSCW</i> , 27:679–714, 2018
SE4	Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco A. Gerosa, Igor Steinmacher, and Anita Sarma. Recommending tasks to newcomers in OSS projects: How do mentors handle it? In <i>Int Symp Open Collaboration</i> , pages 1–14, 2020
SE5	Ann Barcomb. <i>Retaining and Managing Episodic Contributors in Free/Libre/Open Source Software Communities</i> . PhD thesis, University of Limerick, 2019
SE6	Amiangshu Bosu and Kazi Zakia Sultana. Diversity and inclusion in open source software (OSS) projects: Where do we stand? In <i>2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)</i> , pages 1–11. IEEE, 2019
SE7	Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. Finding gender-inclusiveness software issues with GenderMag: A field investigation. In <i>Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems</i> , pages 2586–2598, 2016
SE8	Edna Dias Canedo, Rodrigo Bonifácio, Márcio Vinicius Okimoto, Alexander Serebrenik, Gustavo Pinto, and Eduardo Monteiro. Work practices and perceptions from women core developers in OSS communities. In <i>Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)</i> , pages 1–11, 2020
SE9	Kevin Carillo, Sid Huff, and Brenda Chawner. What makes a good contributor? Understanding contributor behavior within large free/open source software projects – a socialization perspective. <i>J Strategic Information Systems</i> , 26(4):322–359, 2017
SE10	Kattiana Constantino, Shurui Zhou, Mauricio Souza, Eduardo Figueiredo, and Christian Kästner. Understanding collaborative software development: An interview study. In <i>ICGSE</i> , pages 55–65, 2020

(continued)

Table 20-2. (continued)

Label	Reference
SE11	Zixuan Feng, Amreeta Chatterjee, Anita Sarma, and Iftekhar Ahmed. Implicit mentoring: The unacknowledged developer efforts in open source. <i>CoRR</i> , abs/2202.11300, 2022
SE12	Casey Fiesler, Shannon Morrison, R. Benjamin Shapiro, and Amy S. Bruckman. Growing their own: Legitimate peripheral participation for computational learning in an online fandom community. In <i>CSCW</i> , pages 1375–1386, 2017
SE13	Anna Filippova, Erik H. Trainer, and James D. Herbsleb. From diversity by numbers to diversity as process: supporting inclusiveness in software development teams with brainstorming. In <i>ICSE</i> , pages 152–163, 2017
SE14	Jana Gallus and Sudeep Bhatia. Gender, power and emotions in the collaborative production of knowledge: A large-scale analysis of Wikipedia editor conversations. <i>Organizational Behavior and Human Decision Processes</i> , 160:115–130, 2020
SE15	Nasif Imtiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina Bai, and Emerson Murphy-Hill. Investigating the effects of gender bias on GitHub. In <i>ICSE</i> , pages 700–711. IEEE, 2019
SE16	Elham Kariri and Carlos Rodríguez. E-mentoring activities in online programming communities: An empirical study on Stack Overflow. In <i>Service Research and Innovation</i> , pages 123–138. Springer, 2018
SE17	Yu-Wei Lin and Matthijs den Besten. Gendered work culture in free/libre open source software development. <i>Gender, Work & Organization</i> , 26(7):1017–1031, 2019
SE18	Christopher Mendez, Hema Susmita Padala, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Nupoor Patil, Anita Sarma, and Margaret Burnett. Open source barriers to entry, revisited: A sociotechnical perspective. In <i>ICSE</i> , pages 1004–1015, 2018
SE19	Ralph Morelli, Allen Tucker, Norman Danner, Trishan R. De Lanerolle, Heidi JC Ellis, Ozgur Izmirli, Danny Krizanc, and Gary Parker. Revitalizing computing education through free and open source software for humanity. <i>CACM</i> , 52(8):67–75, 2009
SE20	Dawn Nafus. “Patches don’t have gender”: What is not open in open source software. <i>New Media & Society</i> , 14(4):669–683, 2012

(continued)

Table 20-2. (continued)**Label Reference**

- SE21 Susmita Hema Padala, Christopher John Mendez, Luiz Felipe Dias, Igor Steinmacher, Zoe Steine Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Dale Simpson, Margaret Burnett, Marco Aurelio Gerosa, and Anita Sarma. How gender-biased tools shape newcomer experiences in OSS projects. *TSE*, 2020
- SE22 Whitney E. Powell, D. Scott Hunsinger, and B. Dawn Medlin. Gender differences within the open source community: An exploratory study. *Journal of Information Technology*, 21(4):29–37, 2010
- SE23 Gede Artha Azriadi Prana, Denae Ford, Ayushi Rastogi, David Lo, Rahul Purandare, and Nachiappan Nagappan. Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in OSS. Technical Report 2010.00822, arXiv, 2020
- SE24 Leonard Przybilla, Maximilian Rahn, Manuel Wiesche, and Helmut Krcmar. The more, the merrier? The effect of size of core team subgroups on success of open source projects. In *Internationale Tagung Wirtschaftsinformatik*, 2019
- SE25 Jefferson Silva, Igor Wiese, Daniel M. German, Christoph Treude, Marco Aurélio Gerosa, and Igor Steinmacher. A theory of the engagement in open source projects via summer of code programs. In *ESEC/FSE*, pages 421–431, 2020
- SE26 Igor Steinmacher, Sogol Balali, Bianca Trinkenreich, Mariam Guizani, Daniel Izquierdo-Cortazar, Griselda G. Cuevas Zambrano, Marco Aurélio Gerosa, and Anita Sarma. Being a mentor in open source projects. *J. Internet Serv. Appl.*, 12(1):7, 2021
- SE27 Igor Steinmacher, Igor Wiese, Tayana Uchoa Conte, and Marco Aurelio Gerosa. Increasing the self-efficacy of newcomers to open source software projects. In *Brazilian Symposium on Software Engineering*, pages 160–169, 2015
- SE28 Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson R. Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: pull request acceptance of women versus men. *PeerJ Comput. Sci.*, 3:e111, 2017
- SE29 Erik H Trainer, Arun Kalyanasundaram, and James D. Herbsleb. E-mentoring for software engineering: a socio-technical perspective. In *ICSE-SEET*, pages 107–116. IEEE, 2017
- SE30 Zhendong Wang, Yi Wang, and David Redmiles. Competence-confidence gap: A threat to female developers' contribution on GitHub. In *ICSE SEIS*, pages 81–90. IEEE, 2018

Bibliography

- [1] Tammy D. Allen, Lillian T. Eby, Mark L. Poteet, Elizabeth Lentz, and Lizzette Lima. Career benefits associated with mentoring for protégés: A meta-analysis. *Journal of Applied Psychology*, 89(1):127, 2004.
- [2] Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco Aurélio Gerosa, Igor Steinmacher, and Anita Sarma. Recommending tasks to newcomers in OSS projects: How do mentors handle it? In Gregorio Robles, Klaas-Jan Stol, and Xiaofeng Wang (editors), *OpenSym 2020: 16th International Symposium on Open Collaboration, Virtual Conference, Spain, August 26–27, 2020*, pages 7:1–7:14. ACM, 2020.
- [3] Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurélio Gerosa. Newcomers' barriers ...is that all? An analysis of mentors' and newcomers' barriers in OSS projects. *Comput. Support. Cooperative Work.*, 27(3–6):679–714, 2018.
- [4] Bettina M. Beech, Jorge Calles-Escandon, Kristen G. Hairston, Ms. Sarah E. Langdon, Brenda A. Latham-Sadler, and Ronny A. Bell. Mentoring programs for underrepresented minority faculty in academic medical centers: a systematic review of the literature. *Academic Medicine: Journal of the Association of American Medical Colleges*, 88(4), 2013.
- [5] Eliana Bonifacino, Eloho O. Ufomata, Amy H. Farkas, Rose Turner, and Jennifer A. Corbelli. Mentorship of underrepresented physicians and trainees in academic medicine: A systematic review. *Journal of General Internal Medicine*, 1–12, 2021.
- [6] Ann J. Brown, Damon M. Seils, and Paula M. Thompson. Access and diversity in academic mentoring. *JAMA*, 298(7):739–739, 2007.
- [7] Angela Byars-Winston and Maria Lund Dahlberg (editors). *The Science of Effective Mentorship in STEMM*. National Academies Press, 2019.

- [8] David Clutterbuck and Belle Rose Ragins. *Mentoring and Diversity: An International Perspective*. Routledge, 2002.
- [9] Fabian Fagerholm, Alejandro S. Guinea, Jürgen Münch, and Jay Borenstein. The role of mentoring and project characteristics for onboarding in open source software projects. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2014.
- [10] Amy H. Farkas, Eliana Bonifacino, Rose Turner, Sarah A. Tilstra, and Jennifer A. Corbelli. Mentorship of women in academic medicine: a systematic review. *Journal of General Internal Medicine*, 34(7):1322–1329, 2019.
- [11] Denae Ford, Alisse Harkins, and Chris Parnin. Someone like me: How does peer parity influence participation of women on Stack Overflow? In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 239–243. IEEE, 2017.
- [12] Denae Ford, Justin Smith, Philip J. Guo, and Chris Parnin. Paradise unplugged: Identifying barriers for female participation on stack overflow. In *FSE*, pages 846–857, 2016.
- [13] Hana Frluckaj, Laura Dabbish, David Gray Widder, Huilian Sophie Qiu, and James Herbsleb. Gender and participation in open source software development. 2022.
- [14] Katherine E. Kram. Mentoring at work. *Scott Foresman*, 1985.
- [15] Jordan S. Lefebvre, Gordon A. Bloom, and Todd M. Loughead. A citation network analysis of career mentoring across disciplines: A roadmap for mentoring research in sport. *Psychology of Sport and Exercise*, 49:101676, 2020.
- [16] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, pages 1–10, 2008.

- [17] Huiyan Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. Going farther together: The impact of social capital on sustained participation in open source. In *ICSE*, pages 688–699. IEEE, 2019.
- [18] Vandana Singh and Brice Bongiovanni. Motivated and capable but no space for error. *The International Journal of Information, Diversity, & Inclusion*, 5(3):98–126, 2021.
- [19] Vandana Singh, Brice Bongiovanni, and William Brandon. Codes of conduct in open source software – for warm and fuzzy feelings or equality in community? *Software Quality Journal*, 30(2):581–620, 2022.
- [20] Heidrun Stoeger, Daniel Patrick Balestrini, and Albert Ziegler. Key issues in professionalizing mentoring practices. *Annals of the New York Academy of Sciences*, 1483(1):5–18, 2021.
- [21] Erik H. Trainer, Arun Kalyanasundaram, and James D. Herbsleb. E-mentoring for software engineering: A socio-technical perspective. In *39th IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training Track, ICSE-SEET 2017, Buenos Aires, Argentina, May 20–28, 2017*, pages 107–116. IEEE Computer Society, 2017.
- [22] Bianca Trinkenreich. Please don't go – a comprehensive approach to increase women's participation in open source software. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 293–298. IEEE, 2021.
- [23] Bianca Trinkenreich, Igor Wiese, Anita Sarma, Marco Gerosa, and Igor Steinmacher. Women's participation in open source software: a survey of the literature. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(4):1–37, 2022.

- [24] Gloriana Trujillo, Pauline G. Aguinaldo, Chelsie Anderson, Julian Bustamante, Diego R. Gelsinger, Maria J. Pastor, Jeanette Wright, Leticia Márquez-Magaña, and Blake Riggs. Near-peer stem mentoring offers unexpected benefits for mentors from traditionally underrepresented backgrounds. *Perspectives on Undergraduate Research and Mentoring: PURM*, 4(1), 2015.
- [25] Nicole Wagner, Khaled Hassanein, and Milena Head. Computer use by older adults: A multi-disciplinary review. *Computers in Human Behavior*, 26(5):870–882, 2010.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 21

Bringing Diversity in Software Engineering Education from the Middle East and Africa

Mohammad Samarah, University of Maryland Baltimore County, USA.*

A.K. Ocansey, Nekotech Center of Excellence, Ghana.

Esaa Mohammad Sabti Samarah, Florida State University, USA.

Olufunmilola Babalola, Youth of Hope—Nekotech Center Ghana.

Introduction

Software and software engineering touch every human in every corner of the globe, yet many parts of the world are vastly underrepresented or completely absent from participating in software engineering research, development of software applications, and the creation of software-intensive products. Various nations across the African continent and the Middle East are examples. In this chapter, we use the nations of Jordan and Ghana as exemplars to review current efforts within the region. Ghana and Jordan were selected as exemplars for several reasons. First, both Ghana in West Africa and Jordan in the heart of the Middle East are both economically stable countries that border nations of varying economic and political unrest, making them attractive to persons in the region seeking educational and professional opportunities. Second, like the geographic regions where they are found, Ghana and Jordan contain large concentrations

of young people who are undereducated and underemployed. Currently, young people within these two nations have limited opportunity to contribute to the development of emerging markets and are therefore poised to make an immediate impact if given the opportunity. Lastly, the economies of both nations have been steadily increasing for the past ten years with annual rates of growth at or near global averages, suggesting economic stability conducive to software engineering education and development.

The goal of this review is to answer three fundamental questions, namely, how big is the gap in software engineering practice in the Middle East and Africa, what has been the impact of this gap on EDI globally, and are Ghana and Jordan contributing at sufficient levels to software engineering practice? To answer these questions, a systematized review of each nation's population, GDP, human capital, and global software footprint was conducted. Using guidance from the lead authors, Dr. Samarah and Dr. Ocansey, themselves natives of Jordan and Ghana, respectively, academic literature, global population repositories, and industry resources were reviewed to gauge the extent to which each of these two nations is contributing to software engineering practice, education, research, and policy. Google Scholar was used to conduct an initial search of the academic literature using the terms "Global Software Engineering," "Software Engineering Education in Africa," "Software Engineering Education in the Middle East," and "[EDI] Global Software Engineering Education." Authors used reference harvesting techniques to identify relevant literature from articles identified in the initial search. Repositories and industry resources cited throughout the chapter were used to provide estimates of current population and economic trends in the two exemplar countries and their respective regions. Electronic databases including Academic Search Complete (EBSCO), Embase, and CENTRAL were also used to access full manuscripts of identified articles. Search criteria were scoping and represent an exploratory analysis of the literature available on this topic. We caution against generalizing findings presented in this chapter as representative of the entire region, but rather challenge the reader to apply the information presented as two case studies to be referenced as prototypes in future equity, diversity, and inclusion initiatives. We argue that the populations of Jordan and Ghana, like their respective regions, are ideally positioned for innovation. Using a systematized review of the literature, we identify four themes relevant to current efforts to encourage diversity in software engineering from the Middle East and Africa, namely, increasing access to education broadly, identification of current gaps in engineering practice, improving software engineering education and related research efforts, and improvements in policy to encourage software engineering innovation in underrepresented regions. Emanating from these four themes identified in the extant

literature, we propose a plan for positive change to bring diversity from Africa and the Middle East to software engineering innovation and the future of software based on four pillars: (1) Access and Social Justice, (2) Software Engineering Practice, (3) Software Engineering Education and Research, and (4) Software Engineering Policy.

Pillar 1: Access and Social Justice

There is a crisis of education in the Middle East and Africa. This region includes scores of linguistic, ethnic, and cultural subgroups that far outnumber the territorial borders superimposed on the map during the last two centuries. The population within the Middle East and Northwest Africa (MENA) is younger than the global average including an estimated 108 million people between the ages of 15 and 24. Millions of these children and young adults have limited access to primary, secondary, and post-secondary forms of education [1].

Armed conflict in the region has made a fragile situation even worse, forcing the displacement of millions of people and limiting the availability of critical resources. The United Nations (UN) High Commissioner for Refugees estimates the global population of displaced persons and refugees at 84 million individuals, with 51% of that number being children and youth 18 years old and younger [2]. Estimates on the number of displaced persons within the region vary between 15 and 16 million people, a number that is expected to grow in the proceeding decade. Like the global population of refugees, displaced persons within these countries are more likely to be under the age of 18, be women/girls, and have limited access to ongoing education [2].

Nearly half of the population in MENA are under the age of 24, with one in five individuals between the ages 10 and 24 years old. The human potential contained in this concentration of youth is enormous; however, to tap into this potential, equal access to educational opportunities is required. It is widely known that disparities in education contribute to social and health inequity. In the Middle East and Northwest Africa, these disparities are abundantly clear when examining the average return in annual income for each additional year of schooling. Researchers examining wage income as a function of schooling and experience found that MENA countries fall below the global average by as much as half [3]. Egypt has the lowest rate of return in the region at 5%, which is half of the global 10% average. These observed disparities are exacerbated by gender, with women in the region receiving significantly less return for identical levels of education. What's more, women and girls in the region have less access to education, further limiting their ability to maximize their potential [4].

Education has physical health consequences as well, with evidence suggesting that the length of time a person remains engaged in formal education predicts health and longevity [5]. Education is such a strong predictor of health that a dose-response relationship between increased education and better health outcomes has been observed and replicated over time [6–8]. Education among women and girls carries specific benefits, including improved infant and child mortality, fertility, more sustainable population growth, and improved child nutrition [9]. These findings have important implications for the scores of youths and young adults with limited access to education in MENA with a particular emphasis on women and girls in the region who are further disadvantaged than their peers.

Global Software Engineering

Global software engineering (GSE) is now standard practice [10–13]. GSE refers to the software development carried out by decentralized teams of experts located around the globe working concurrently to develop commercially viable software for companies, governments, and related groups. Scores of institutions have developed GSE-specific training programs to meet swelling demand in this domain [11]. John White, former CEO of the Association of Computing Machinery (ACM), wrote in their 2006 report on the globalization and offshoring of software that

... Computing and information technology has experienced a dramatic shift in the past five years to a truly global industry, the forces that have driven and shaped this change are still at play and will continue.

—John White, CEO, Association of Computing Machinery

The intervening decade and a half since this report was published have provided ample evidence to support this prediction. GSE is the dominant form of software development for multinational companies looking to compete on a global scale. The demand for GSE-capable software engineering talent is predicated on the availability of quality GSE training programs that produce capable engineers. GSE is distinct from other subdomains within software engineering education in that the integration of global methods is essential to effective practice. Effective GSE training and practice requires teamwork across institutions with varying cultural work norms and different skill sets. To do this effectively, GSE educators and producers must collaborate with diverse talent from underrepresented regions. Emerging cohorts of young people like those in MENA countries are ideally positioned to inform the continued expansion of GSE.

Access Is Social Justice: Global Software Engineering

Social justice is focused on the preservation of the dignity and worth of human beings including their basic human rights. Inclusive to human rights is respect for an individual's well-being through the preservation of basic needs including water, food, shelter, health, and education [14]. These rights overlap with one another with each component sharing collinearity. Access to education and economic opportunity is therefore essential to any social justice effort aimed at the promotion and preservation of human rights. Equal access to education is a powerful antidote to the effects of displacement, poverty, and social disparities [15].

Although all forms of education have the potential to address these long-standing disparities, GSE education should be considered as an essential component of any plan to expand educational and economic opportunities to under-resourced populations. The expansion of GSE-related employment is expected to continue over the next decade, which will have important economic implications for the significant number of people aging into the workforce in MENA. In the absence of opportunity, these individuals will likely fall prey to intergenerational cycles of poverty and poor health outcomes observed in the region over the last half century [16]. In addition, the expansion of GSE to labor markets left largely untapped in MENA necessitates the inclusion of workers from within those regions to inform products and ensure accurate representation. This, of course, must specifically include women who are globally underrepresented in software education and development [17].

Strategies to Expand Access

Unlocking the potential of young people, specifically women and girls, in MENA is an important social justice strategy that can be leveraged for the preservation of human rights in the region. Promoting equal access to education must be a key part of this strategy. Promising trends have been observed in the region with some estimates indicating that the education gap between men and women in the region is shrinking [18]. Despite these improvements many challenges remain including securing employment following education [3, 19]. Among areas within MENA where access to education has been expanded, the quality of the education provided is often low, which impacts employment opportunity [20]. Failure to secure sufficient economic opportunity following investments in education can undermine progress made in the region.

Effective interventions aimed at addressing education disparities in the region should be multifaceted. First, efforts to expand educational opportunities should be targeted toward women and girls who are often left out of education and labor force participation due to cultural mores and societal structure [21]. Evidence suggests that all groups benefit from interventions targeted at the most disadvantaged groups. Second, recruitment efforts should focus on identifying motivated students who are qualified or nearly qualified to participate in GSE-specific training. For students who are nearly qualified, bridge programs should be offered to ensure adequate preparation prior to enrollment and retention. Third, GSE training programs must track with global standards for education to ensure marketability of students following graduation.

Efforts to improve diversity, equity, and inclusion in software engineering education in MENA countries must elevate beyond rhetoric and include systematic planning for the inclusion of young people broadly and young women and girls specifically. Institutions of higher education, non-governmental organizations, and governing bodies would benefit from expanding GSE training programs within MENA countries as an effective social justice campaign to promote economic opportunity and social health. The human potential locked within the scores of young people aging into the labor market in MENA represents a major challenge for globalization. Institutions that work toward unlocking this potential through systematic investments in education and training will undoubtedly benefit.

Pillar 2: Software Engineering Practice

In this section, we look at software engineering practice in Ghana and Jordan as two exemplar case studies from Northwest Africa and the Middle East, respectively. The objective of this review is to understand the gap in software engineering practices in the Middle East and Northwest Africa and to assess whether Ghana and Jordan are contributing to software engineering practices at sufficient levels. To answer these questions, we conducted a systematized review of each nation's population, GDP, human capital, and global software footprint.

Demographics

Africa and the Middle East are vast and varied regions. To gain an insight into software engineering practices, it is helpful to first look at current demographics of the region. In a 1987 study on software expansion in the developing world, Robert Schware concluded

that some countries may acquire a competitive advantage, while others may lack behind, suggesting that governments and policy makers need to implement strategies to encourage country-wide software development [22]. Thirty-five years later, we argue that the region as a whole continues to have potential to act as a main player in this field. To provide a comparative view of the region, we take Ghana and Jordan as two representative countries. We then examine seven key indicators informed by World Bank Open Data and compare them to world standings. These indicators are population, population growth, GDP, GDP growth, government expenditure on education, school enrollment, and the World Bank Human Capital Index (HCI) [23].

Population and GDP

In 2021, Ghana had a population of 31.73 million people with a population growth of 2.1%, while Jordan's population during that same year was 10.27 million with a 0.6% annual population growth compared with a world population growth of 0.9%. In addition, 30% or more of the population in both countries are ages 0–14, with Ghana at 37% and Jordan at 32% compared with 25% for the world.

The GDP growth of Ghana and Jordan was 5.4% (77.59B USD) and 2.2% (45.75B USD) in 2021, compared with a global average GDP growth of 5.9%. More important, however, are trends in educational spending across both countries. The expenditure on primary, secondary, and tertiary education as a percentage of overall education spending in Ghana is 22%, 37%, and 18% for primary, secondary, and tertiary education, respectively. In Jordan, the government spends 42%, 34%, and 23% of their overall education expenditure on primary, secondary, and tertiary education. Although varied, these data suggest that the largest portions of expenditures across both nations are within primary and secondary education. It stands to reason then that this investment in early education may drive increased enrollment in tertiary education. Looking at school enrollment for tertiary education as a percentage of gross education enrollment, 19% of students in Ghana matriculated to tertiary education, while 34% of students in Jordan matriculated to tertiary education. Globally, 40% of students as a proportion of gross education enrollment matriculate to tertiary education. This comparison underscores the need for more work in the region to provide primary and secondary education students with meaningful pathways to tertiary education.

Human Capital

A meaningful indicator is the Human Capital Index (HCI). Key findings using this index suggest that the likelihood a child born in Ghana and Jordan grows to full productivity with complete education and full health is 45% and 55%, respectively, which are lower than probabilities calculated in middle-income countries. The HCI for girls is slightly higher than that for boys in both countries with estimates suggesting a 0.46 and a 0.58 probability for girls in Ghana and Jordan, respectively. In addition, the HCI ratio of the richest to poorest 20% in Ghana is 1.16 and in Jordan is 1.23. The global average of the richest to poorest is 1.35 with a global range of 1.12–1.68.

Global Footprint

If we examine country presence on the public Internet, we find that the Google search engine shows Ghana with 74,000 publicly accessible websites and Jordan having over 4 million. The Microsoft Bing search engine returns about 10,000 websites for Ghana and over 2 million websites for Jordan. If we look at scholarly output with a measurement of access via a country domain name, Google Scholar shows 13 for Ghana and about 33,000 for Jordan. The low numbers for Ghana may be attributed to using website domain names that do not have the country two-letter code. This conspicuous absence of country-specific domain names is telling in that it suggests an incentive to mask locality in order to gain access to online space. Looking at big tech companies' presence in Ghana and Jordan, we see that Google opened its Africa Artificial Intelligence (AI) center in 2018 in Accra, while GE had an office there since 2014. Twitter has recently announced plans to open its first Africa office in Ghana as well [24]. In Jordan, the King Hussein Business Park houses more than 75 national and international companies such as Microsoft, Cisco, HP, and others including more than 100 startups and the Oasis500 high-tech incubator [25]. Oasis500 was launched in 2011 [26] and has received 14,000 startup applications with a recent evaluation of \$200M. Importantly, 26% of Oasis500 founders are women [27]. The Country Commercial Guide forwarded by the International Trade Administration of the US Department of Commerce identified software as a leading sub-sector in Jordan. The report gives guidelines for opportunities in IT infrastructure projects, Arabization, gaming, financial services, and e-government [28]; however, it did not offer any guidelines for seeking opportunities in the software sub-sector. In comparison, the same guide for Ghana does not offer much guidance on software sub-sector opportunities [29].

Given the region's population scale, expected population growth, and high concentration of youth, both countries offer substantial untapped human capital. Although there is a rising scene of startups in both Ghana and Jordan, the common theme remains one of development and untapped potential. Like development initiatives in Jordan and Ghana, startups in developing nations should focus their energy on serving the local market by first creating innovations that have a direct local impact. Once developed, initiatives within these nations can expand their reach regionally and then globally.

Pillar 3: Software Engineering Education and Research

In this section, we apply a similar case study model to examine software engineering education and research in the Middle East and Africa. The goal of this exercise is to answer another fundamental question: what is the gap in software engineering education and research in the region, and what has been its impact on EDI globally? Are Ghana, Jordan, and neighboring countries making meaningful contributions to software engineering education and research, and how does this impact EDI? Does the region provide software engineering education that includes both a global and a local perspective? Is the region producing innovations, new products, and new services, or is it simply consuming, maintaining, and servicing products built by other developed countries?

It is telling that in 1991, the United Nations University in Macau created the International Institute for Software Technology (UNU-IIIST) with an agreement encompassing the UN and the governments of the nations of Macau, Portugal, and China. In 2003, a colloquium was published to celebrate the tenth anniversary of IIIST with a renewed mission to bring software technology capabilities to developing countries [30]. In 2015, a new director was charged with rebuilding this institute [31] as the United Nations University Institute on Computing and Society (UNU-CS). As this change shows, the impact of software was clearly understood even in the early 1990s, and it became more apparent in the past ten years. Although the UN mission to bring software capabilities to developing nations is noble, we argue that this effort needs to be led by the nations themselves to bring lasting change and full participation, equity, diversity, and inclusion. We examined several studies that looked at different aspects

of software engineering research or software engineering education in the context of Ghana and Jordan. In the following section, we describe some of the studies in detail and suggest improvements to software engineering research and education in the region.

Software Education in African Nations

Cyriaano and Osman argue that software engineering education in Africa not only needs to consider a global perspective but also include understanding of the student and local environments. They argue that often the curriculum being taught was developed in the context of industrialized societies with examples that are foreign to the African students [32]; thus, the effectiveness of the coursework is not achieved, and students are often left confused and lost. For example, one study looked at how teaching an Artificial Intelligence (AI) undergraduate computer science course represented challenges for students in Namibia and how the course plan and approach had to be adapted to the local context [33]. In another study, Korpela and colleagues set out to answer higher-level questions related to software engineering's ability to improve people's lives and economic conditions within Nigeria. Specifically, Korpela and colleagues investigated the way in which information systems development has improved life and human conditions in Nigeria using the results of over 15 years of European-African research. They concluded that this development can have a direct impact on Nigeria and can be generalized to other African and developing countries as well [34].

Software Engineering Education in the Middle East

Studies investigating the impact of software engineering education in the Middle East ranged from research looking at general software engineering education, specific case studies for STEM programs at the tertiary education level, and software engineering coursework to specific topics in software engineering and its applicability to the industry. One study by Al-Zaghoul and colleagues looked at software engineering education in Jordan and how to improve the curriculum beyond global standards from IEEE, ACM, and the Software Engineering Body of Knowledge (SWEBOK). They examined the changes proposed by their local educational accreditation commission, the Higher Education and Accreditation Commission [35]. One can conclude from their study that not only a global view is necessary but also a local context.

Another study by Radaideh et al. benchmarked the software engineering education curriculum at the undergraduate level at one STEM university, Jordan University of Science and Technology (JUST). Their focus was to examine how compliant the curriculum is with the IEEE Software Engineering Body of Knowledge, in particular, knowledge areas in software requirements, software design, software testing, software construction, and software maintenance. They found that the JUST curriculum is compliant with the first three areas and partially compliant with the last two [36].

Looking at specific software engineering courses, Hanna et al. compared courses taught at Jordanian universities with those in other countries including the United States and the United Kingdom. They also examined which courses are software engineering specific and which ones are not. They found that more than 60% of courses taught in such programs are non-software engineering courses and that the coursework needs to be augmented to prepare students to join the market force [37]. Yet, AbuLail and Shkoukani show that software reverse engineering is critical to software engineering education [38], suggesting that cross-training is required and necessary for software comprehension and for bringing to society software engineers who can make improvements to existing systems with full understanding of the system operational and local context.

It is clear from the studies originating from developing countries that there are multiple aspects that need to be addressed in order to achieve higher participation and inclusion of this region in software engineering research and education. Specifically, these components are (a) collaboration and awareness of software engineering research and education among regional countries, (b) software engineering education in primary and secondary schools, (c) changes to curriculum to include both a global and a local perspective, (d) supporting software engineering education research, and (e) greater participation and collaboration between academia and industry in both theoretical and applied research. We are starting to see evidence of the region taking note that they need to work together to play a meaningful role in this space. For example, collaboration among regional countries, the Africa and Middle East Conference on Software Engineering, and the Software Engineering in Africa conference have merged into a new conference, the Federated Africa and Middle East Conference on Software Engineering (FAMECSE). These initiatives, we hope, are welcome signals of a more robust future of software engineering education in the region.

Pillar 4: Software Engineering Policy

Software, software engineering, and the countless products and tools produced therefrom impact every community in the world. Yet, huge swaths of the global population are functionally absent from software engineering education, research, and development. EDI in software engineering is not simply a philanthropic initiative; it is also a sound development strategy for policy makers and visionaries seeking to shape future markets. The gap between the wealthiest and least-resourced countries in the world is projected to continue growing in 2023 [39]. These disparities are particularly alarming when looking at projected outcomes for young women and girls [40]. Regions with limited access to software engineering education, research, and development continue to fall behind the rest of the world in innovation. This phenomenon, of course, is not for lack of available talent. Regions like the continent of Africa and the Middle East are frequently overlooked for investments in education and research, despite the overwhelming presence of human capital in the region [2]. Improving EDI within software engineering education, research, and development within this region must be included in future plans for the global development of software engineering.

Onshoring Global Software Engineering

Evidence suggests that the infusion of knowledge diversity within software education and development can produce vast amounts of creativity and innovation [41]. This process, however, is not without challenges. Failure to mindfully infuse diversity within GSE contexts can result in an erosion of innovation [42]. This corrosive effect on innovation often occurs at the intersection of diverse forms of technical knowledge and shared forms of common knowledge [43]. Globalization within software development has historically been based on the recruitment and retention of diverse forms of technical knowledge and labor that could be leveraged as an “offshore” resource. Offshoring global software engineering is still common today [44]; however, the practice is becoming increasingly antiquated in the context of emerging evidence characterizing the benefits and best practices of knowledge management (KM) [45, 46]. In contrast to offshoring practices that are largely unidirectional, effective KM involves an exchange of information that is based on the social characteristics of teams, shared values, organizational structure, and free availability of information [45]. Policy makers and leaders interested in the benefits of EDI within software engineering should take heed of this emerging evidence and transform their knowledge sharing practices accordingly.

To affect software engineering futures in their own country, leaders and policy makers should be invested in a correctional process of onshoring software engineering talent. For decades now GSE developers have exploited global software engineering talent in India, Central and South America, and Eastern Europe to reduce the costs of company processes [44]. Offshoring, however, is a short-term strategy often leveraged by developers in wealthy nations for cost-saving benefits and not much else. Onshoring, in the context of EDI in software engineering, is instead focused on the development of an ecosystem of domestic innovation such that emerging markets can *contribute* to the development of software innovation. Categorically distinct from offshore labor provided to foreign-based multinational companies, onshoring software engineering within Middle Eastern and African nations will require investments in education, research, and development.

Using EDI to Change the Global Software Engineering Landscape

Investments in EDI within the context of software engineering development and innovation have the potential to change the software engineering landscape globally. Leaders and change makers who fail to recognize the enormous potential of EDI within GSE do so at their own peril. Evidence suggests that software companies that prioritize the recruitment and retention of a culturally, ethnically, and gender-diverse workforce outperform companies that do not. Specifically, companies rated in the top quartile for ethnic and cultural diversity outperform those in the bottom three quartiles by as much as 36%, a figure that has been steadily increasing over the last five years [47]. This marked increase in profitability becomes even more clear when focusing on gender-diverse organizations who, on average, are 48% more likely to outperform their non-diverse competitors [47]. EDI in software engineering increases innovation and creativity [48], increases access to new markets [49], and improves a team's overall adaptability [50].

To effectively create change for a more diverse future in software engineering, effort must be expended in underrepresented regions to recruit, train, and retain the software engineering talent of tomorrow. Africa and the Middle East contain hegemonic potential with respect to GSE. The availability of talent and untapped markets within these regions are ripe for training and innovation. To benefit from the diversity these regions have to offer, critical investments need to be made across four distinct domains, namely, education, awareness, research, and development. First, high-quality software

engineering education and more broadly STEM education need to be available from grade school to college level. In the absence of available educational needs, incentives need to be forwarded to encourage those who seek an education abroad to return to their country of origin to strengthen domestic innovation and development. Second, public and governmental authorities need to contribute to public awareness about the positive benefits of software education and development. Third, software engineering research should be led and conducted by diverse teams within the region working to solve global and local challenges. Fourth, the development of software applications and software-intensive products for consumers within the region must be designed by diverse teams from within the region to ensure success.

Bibliography

- [1] The World Bank. Population ages 0–14 – Middle East & North Africa | Data based on age/sex distributions of United Nations Population Division's World Population Prospects. [Internet]. 2019 [cited August 4, 2022]. Available from <https://data.worldbank.org/indicator/SP.POP.0014.T0.ZS?locations=ZQ>
- [2] UNHCR. Office of the United Nations High Commissioner for Refugees – Refugee Data Finder. [Internet]. International Organizations, 2022. Available from www.unhcr.org/refugee-statistics/
- [3] Rizk R. Returns to education in MENA countries: a continuing story of under-achievement. International Journal of Education Economics and Development. 2019;10(4):427–448.
- [4] World Economic Forum. Gender equality is 170 years away. We cannot wait that long. [Internet]. World Economic Forum, 2017 [cited March 31, 2023]. Available from www.weforum.org/agenda/2017/01/gender-equality-is-170-years-away-we-cannot-wait-that-long/
- [5] Johnston RB. Poor education predicts poor health-a challenge unmet by American medicine. NAM Perspectives. 2019.

- [6] Cohen AK, Syme SL. Education: a missed opportunity for public health intervention. *American Journal of Public Health*. 2013;103(6):997–1001.
- [7] Zajacova A, Lawrence EM. The relationship between education and health: reducing disparities through a contextual approach. *Annual Review of Public Health*. 2018;39:273–289.
- [8] Zimmerman E, Woolf SH. Understanding the relationship between education and health. *NAM Perspectives*. 2014.
- [9] Dwyer DH, Bince J. *A home divided: Women and income in the Third World*. 1988.
- [10] Aspray W, Mayadas F, Vardi MY. Globalization and offshoring of software. In: *The Innovation Imperative*. Edward Elgar Publishing, 2009.
- [11] Fortaleza LL, Conte T, Marczak S, Prikladnicki R. Towards a GSE international teaching network: Mapping Global Software Engineering courses. In: *2012 Second International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD)*. IEEE, 2012. pp. 1–5.
- [12] Noll J, Beecham S, Richardson I. Global software development and collaboration: barriers and solutions. *ACM Inroads*. 2011;1(3):66–78.
- [13] Raza B, MacDonell SG, Clear T. Research in global software engineering: a systematic snapshot. In: *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer, 2013. pp. 126–140.
- [14] Nadkarni VV, Sinha R. Transforming social work education in India: integrating human rights. *Journal of Human Rights and Social Work*. 2016;1(1):9–18.
- [15] Awan MS, Malik N, Sarwar H, Waqas M. Impact of education on poverty reduction. 2011.
- [16] Khouri RG. Poverty, inequality and the structural threat to the Arab region. *Shifting Global Politics and the Middle East*. 2019;28.

- [17] Peixoto A, González CSG, Strachan R, Plaza P, de los Angeles Martinez M, Blazquez M, et al. Diversity and inclusion in engineering education: Looking through the gender question. In: 2018 IEEE Global Engineering Education Conference (EDUCON). IEEE, 2018. pp. 2071–2075.
- [18] Roudi-Fahimi F, Moghadam VM. Empowering women, developing society: Female education in the Middle East and North Africa. Al-Raida Journal. 2006;4–11.
- [19] Chapman DW, Miric SL. Education quality in the Middle East. International Review of Education. 2009;55(4):311–344.
- [20] Drine I. Education and entrepreneurship to address youth unemployment in MENA Region. In: Expert Group Meeting on Strategies for Eradicating Poverty to Achieve Sustainable Development for All United Nations, New York. 2017.
- [21] Moghadam VM. The women's movement in the Middle East and North Africa: responding to restructuring and fundamentalism. Women's Studies Quarterly. 1998;26(3/4):57–67.
- [22] Schware R. Software industry development in the Third World: Policy guidelines, institutional options, and constraints. World Development. 1987;15(10–11):1249–1267.
- [23] The World Bank. World Bank Open Data. [Internet]. 2022 [cited September 28, 2022]. Available from <https://data.worldbank.org/>
- [24] Asiedu K. Tech Giants Made a Home in Ghana. Now They're Quiet on Its Anti-LGBTQ+ Bill. [Internet]. Pulitzer Center, 2022 [cited September 28, 2022]. Available from <https://pulitzercenter.org/stories/tech-giants-made-home-ghana-now-theyre-quiet-its-anti-lgbtq-bill>
- [25] King Hussein Business Park. The Pulse of Business. [Internet]. 2022 [cited September 28, 2022]. Available from <https://businesspark-jo.com/>

- [26] The Next Society. Invest in Jordan's startup founders. [Internet]. 2022 [cited September 28, 2022]. Available from www.thenextsociety.co/oasis500
- [27] Oasis500. The first of its kind in MENA. [Internet]. Oasis500, 2022 [cited September 28, 2022]. Available from <https://oasis500.com/>
- [28] International Trade Administration. Jordan - Information and Communication Technology. [Internet]. 2021 [cited September 28, 2022]. Available from www.trade.gov/country-commercial-guides/jordan-information-and-communication-technology
- [29] International Trade Administration. Ghana Country Commercial Guide. [Internet]. 2021 [cited September 28, 2022]. Available from www.trade.gov/ghana-country-commercial-guide
- [30] Chaochen Z. UNU and UNU/IIST. In: Aichernig BK, Maibaum T (editors). Formal Methods at the Crossroads from Panacea to Foundational Support: 10th Anniversary Colloquium of UNU/IIST, the International Institute for Software Technology of the United Nations University, Lisbon, Portugal, March 18–20, 2002, Revised Papers. [Internet]. Berlin, Heidelberg: Springer, 2003 [cited September 28, 2022]. pp. 26–33. (Lecture Notes in Computer Science). Available from https://doi.org/10.1007/978-3-540-40007-3_2
- [31] United Nations University. Prof. Michael Best Appointed Director of UNU Institute on Computing and Society, Macao, China – United Nations University. [Internet]. 2015 [cited September 28, 2022]. Available from <https://unu.edu/media-relations/releases/prof-michael-best-appointed-director-of-unu-institute-on-computing-and-society-macao-china.html>
- [32] Cyriaano A, Osman R. Fitting a generic computer science curriculum into context: The African case. In: ICERI2011 Proceedings. IATED, 2011. pp. 5705–5712.

- [33] Shipepe A, Uwu-Khaeb L, Kolog EA, Apiola M, Mufeti K, Sutinen E. Towards the Fourth Industrial Revolution in Namibia: An Undergraduate AI Course Africanized. In: 2021 IEEE Frontiers in Education Conference (FIE). IEEE, 2021. pp. 1–8.
- [34] Korpela M, Mursu A, Soriyan HA, Harpe R de la, Macome E. Information systems practice for development in Africa: Results from INDEHELA. In: Social Inclusion: Societal and Organizational Implications for Information Systems. Springer, 2006. pp. 15–35.
- [35] Al-Zaghoul F, Hudaib A, Ahed M. Software engineering education in Jordan. In: 2014 6th International Conference on Computer Science and Information Technology (CSIT). IEEE, 2014. pp. 127–132.
- [36] Radaideh MA. Benchmarking the Software Engineering Undergraduate Program Curriculum at Jordan University of Science and Technology with the IEEE Software Engineering Body of Knowledge (Software Engineering Knowledge Areas# 1–5). In: Advances in Software Engineering, Education, and e-Learning. Springer, 2021. pp. 747–768.
- [37] Hanna S, Jaber H, Jaber FA, Al Shalaby T, Almasalmeh A. Enhancing the software engineering curriculums: A case study of the Jordanian Universities. In: 2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T). IEEE, 2014. pp. 84–93.
- [38] AbuLail R, Shkoukani M. The Reality of Software Reverse Engineering Education in the Jordanian Universities and How to improve it. International Journal on Computer Science and Engineering. 2013;5(3):174.
- [39] The World Bank. Global Wealth Has Grown, But at the Expense of Future Prosperity. [Internet]. World Bank, 2021 [cited September 19, 2022]. Available from www.worldbank.org/en/news/press-release/2021/10/27/global-wealth-has-grown-but-at-the-expense-of-future-prosperity-world-bank

- [40] The World Bank. Unrealized Potential: The High Cost of Gender Inequality in Earnings. [Internet]. World Bank, 2018 [cited September 19, 2022]. Available from www.worldbank.org/en/topic/gender/publication/unrealized-potential-the-high-cost-of-gender-inequality-in-earnings
- [41] Orlikowski WJ. Knowing in practice: Enacting a collective capability in distributed organizing. *Organization Science*. 2002;13(3):249–73.
- [42] Desouza KC, Awazu Y. Knowledge management at SMEs: five peculiarities. *Journal of Knowledge Management*. 2006.
- [43] Clear T, Beecham S, Barr J, Daniels M, McDermott R, Oudshoorn M, et al. Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review. *Proceedings of the 2015 ITiCSE on Working Group Reports*. 2015;1–39.
- [44] Woodard MS, Sherman KE. Toward a more complete understanding of offshoring: Bringing employees into the conversation. *The International Journal of Human Resource Management*. 2015;26(16):2019–38.
- [45] Choo CW, de Alvarenga Neto RCD. Beyond the ba: managing enabling contexts in knowledge organizations. *Journal of Knowledge Management*. 2010.
- [46] Santos V, Goldman A, De Souza CR. Fostering effective inter-team knowledge sharing in agile software development. *Empirical Software Engineering*. 2015;20(4):1006–51.
- [47] Dixon-Fyle S, Dolan K, Hunt DV, Prince S. How diversity, equity, and inclusion (DE&I) matter | McKinsey. [Internet]. McKinsey & Company, 2020 [cited September 19, 2022]. Available from www.mckinsey.com/featured-insights/diversity-and-inclusion/diversity-wins-how-inclusion-matters
- [48] Holger D. The Business Case for More Diversity – The Wall Street Journal. [Internet]. 2019 [cited September 19, 2022]. Available from www.wsj.com/articles/the-business-case-for-more-diversity-11572091200

- [49] Hewlett SA, Marshall M, Sherbin L. How Diversity Can Drive Innovation. Harvard Business Review. [Internet]. December 1, 2013 [cited September 19, 2022]. Available from <https://hbr.org/2013/12/how-diversity-can-drive-innovation>
- [50] Sakpal M. CIOs should embrace DEI to build successful teams. [Internet]. Gartner, 2019 [cited September 19, 2022]. Available from www.gartner.com/smarterwithgartner/diversity-and-inclusion-build-high-performance-teams



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 22

Open Sourcing Diversity, Equity, and Inclusion

Demetris Cheatham, GitHub, USA.*

Introduction

The timeline of software engineering can be neatly split into before and after open source. From Linux, Python, and Blender all the way to a literal trip to Mars, open source has revolutionized our relationship to technology and, by extension, fundamentally transformed the world we live in today. These accomplishments are owed to a simple set of core principles sometimes referred to as the open source way: transparency, collaboration, inclusion, community, and early, frequent releases. Yoked together, these practices have freed software developers from redundant and siloed work, curtailing stagnation while radically multiplying opportunities for innovation and growth. And, in privileging hard skill sets over formal credentialism, open source has lowered barriers to entering the tech field and made professional advancement more attainable.

While open source has proven itself to be effective, global, and progressive, it's still vulnerable to structural systems of bias and oppression, including racism, ableism, sexism, ageism, and queerphobia. Case in point: Despite the well-established fact that greater diversity on teams produces higher-quality, more innovative products, the tech industry is still overwhelmingly led by white, university-educated, cis-gender men. To put it bluntly, if we know that more inclusive, ethically operated businesses and projects make for higher-performing, happier teams and more robust technologies, then the lack of diversity in tech doesn't just perpetuate inequality – it's actively holding us back.

I've spent my entire career working to advance diversity, equity, and inclusion (DEI) in one capacity or another across the public, private, and nonprofit sectors, from Wall Street to Washington, DC, and beyond. As a technology analyst at Goldman Sachs – my first job out of college – I learned what it feels like to participate in an inclusion program for people from underrepresented backgrounds. I then returned to school for my JD and MBA, eventually becoming the first woman and youngest person to serve as Executive Director of the National Bar Association. There, I encountered the reach and limitations of social sector organizations in effecting change. When I moved on, it was to a chief-of-staff role with an elected official in Washington, DC, for whom I oversaw efforts on racial equity and economic inclusion and where I learned how government policy fits into anti-bias work. More recently, I held the role of Global Diversity and Inclusion Lead at Red Hat, where I was faced with the challenges and benefits of doing DEI work at a private corporation. And, in my continuing work as a lecturer at North Carolina State University, I have a first-hand view of the structural biases that students, as well as educators, have to grapple with on a daily basis.

Today, I'm the Chief of Staff to the CEO, formerly the Senior Director of Diversity, Inclusion and Belonging Strategy, at GitHub, where I find myself with a rare opportunity to synthesize these distinct but interwoven experiences. What I've found across the board is no shortage of good intentions. I've seen how much research is conducted, how many reports are published, and how many statements and policies are generated in the service of achieving DEI goals. And I've seen how, more often than not, DEI work is done behind closed doors, such that the resources out there tend to remain siloed, duplicative, and limited by the scope or reach of the institutions that create them.

In other words, if open source has a “diversity problem,” then DEI has an “open source problem.” I've set out to explore exactly this dilemma and investigate a critical question: **How can we open source DEI?**

To answer this question, I've led GitHub in the launch of an open source DEI program called All In. Our primary mission is to use open source principles to foster open source environments that are welcoming, hospitable, and nurturing to all talent. To extend my earlier metaphor, this includes documenting our efforts and findings in an open source environment so that businesses, community leaders, and other interested parties can both draw from and contribute to this critical work.

GitHub isn't the only hosting platform for open source projects, but at the time of writing, it's certainly the largest: GitHub reaches over 100 million developers and counting, across every state in the United States and virtually every country and region in

the world. I say this not to toot our own horn but to illustrate that we are a resource that's shared by private companies, government agencies, universities, project managers, students, and developers. Being that common denominator makes us uniquely well-positioned to harness those aforementioned principles of open source (transparency, collaboration, inclusion, community, and early, frequent releases) to unite specialized stakeholders in addressing the glaring disparities in representation and belonging within this field.

Over the course of this chapter, I'll investigate the ways in which an open source model can supplant slow, disjointed DEI efforts with nimble, inquisitive, community-driven solutions. I'll walk you through the steps we've taken thus far, from gathering quantitative and qualitative data to establishing and growing All In. Finally, I'll share how you, too, can join this movement.

The Journey

Even the best intended efforts can go awry when leaders make assumptions, rush to conclusions, or allow implicit or explicit bias to guide their decision-making. I've repeatedly found that the best antidote to these pitfalls is dialogue, even – or especially – when it's not pretty. Step 1 of open sourcing DEI was no different: we had to start by listening.

At GitHub, we took a two-pronged approach to information gathering. It was important for us to collect both quantitative and qualitative data, not only as a jumping-off point but to allow us to set benchmarks and track changes over time.

First, we partnered with the Linux Foundation to create the 2021 Open Source Diversity, Equity, and Inclusion Survey. In order to gather a data set that spoke as accurately as possible to the state of DEI in open source at a particular moment in time, we knew it was critical to examine lived, intersectional realities and ask questions that invited as many participants and perspectives as possible. Because ample research points to the ways in which bias can infiltrate seemingly neutral survey questions, we decided to make crafting this survey something of an open source DEI project unto itself. After each revision, we asked ourselves, "Who hasn't seen this? What voices aren't represented here? Who else do we need to talk to?" Ultimately, over 200 people with different identities reviewed and contributed to the questions. The survey, launched in July 2021 and garnered responses from over 7,000 members of the international open source community.

Second, we set out to gain a better understanding of the strengths, limitations, and challenges faced by open source community leaders. These project managers and maintainers are key to driving change, but we already knew anecdotally that many were struggling to realize their DEI goals. In that spirit, we embarked on a listening tour, inviting maintainers from around the world to speak to us about their experiences. We ran these conversations virtually and at industry conferences and also circulated an online form for those who couldn't participate in an individual interview or focus group. In this initial four-month push, we spoke to over 300 maintainers who ran projects of varying sizes, demographic makeups, and missions.

I'd like to share some of the most meaningful and immediately actionable conclusions we've reached in analyzing both sets of data:

1. Open source needs to be more inclusive.

One thing that was consistent, no matter who we talked to or the size of their community, is that there are still significant barriers to access and belonging in open source for folks from underrepresented or historically excluded backgrounds. Though we were initially quite pleased to learn that 82% of our respondents agreed with the statement, "I feel welcome in open source," we've had to balance that feedback with the fact that over 80% of our survey respondents identify as male, 74% identify as heterosexual, and 71% are between the ages of 25 and 54: a relatively homogenous group.

Diversity, equity, and belonging work needs to focus on the 18%, or the one in almost five of our survey respondents, for whom inclusion is the exception, not the norm. We simply cannot expect to recruit and retain a broader, more diverse pool of developers until more is being done for the 25% of people with disabilities, the 26% of women, the 29% of persons of color in North America, and the 38% of non-binary and third-gender contributors who do not feel welcome in open source.

2. DEI intervention should begin at the community level.

Statements and policies about diversity and inclusion are important, but our survey respondents made it clear that a sense of belonging is generated first and foremost out of day-to-day interactions between community members.

Negative interactions take a multitude of forms, any of which can lead to someone leaving a project or even giving up on the open source community for good. We found that women, non-binary, LGBTQ+, and people with disabilities were twice as likely as other respondents to have experienced threats of violence and that profanity, racist jokes, sexual imagery, and rudeness all affect someone's sense of belonging. But what our survey emphasized is that silence and passivity can be just as toxic as explicitly hostile interactions and bigoted language.

Some people reported feeling that without certain technical skills and knowledge, they aren't welcome to participate. This is likely complicated by the fact that many said the path to leadership and personal growth on a project can be opaque, leaving aspiring developers in limbo. Other respondents shared that not being white or male or highly educated or wealthy makes them feel that their voices aren't heard and their contributions aren't valued. Still others said that when they do try to participate, their contributions are rejected or ignored. A full 80% of respondents who don't feel welcome said that feeling ignored or not receiving a response to their contributions happens occasionally, and almost 40% said that it happens regularly.

3. **Maintainers need help.**

If day-to-day interactions between team members are what create or dispel a sense of belonging, then it's the maintainer or project manager who sets the tone and framework of those interactions.

Now, on the bright side, virtually all of the maintainers we spoke to said they understand the importance of fostering an inclusive culture and working with a diverse team and that they have access to plenty of training resources and boilerplate codes of conduct. But maintainers shared that the pressure to onboard people quickly to get a project off the ground can be all-consuming, leaving them stretched too thin to focus on the community's culture or invite the participation of contributors outside their relatively homogenous personal networks.

All agreed that it's more efficient and humane for inclusion efforts to happen at the outset of a project – and that this is exactly the moment when they're least likely to have adequate time to actually focus on it. With the appropriate resources to identify and de-escalate code-of-conduct violations and enforce best practices, maintainers would be better equipped to safeguard their communities.

4. We need to invest better in our ecosystem.

Much of the focus thus far has been on existing community members. But if we truly want to improve diversity, equity, and inclusion in the open source landscape, we need to identify students and potential community members earlier in their journey and supplement their tech education. As we've begun working with researchers, students, maintainers, and community leaders, it's become increasingly clear that current funding strategies are suboptimal for DEI.

This starts with access to both broadband and devices. In 2021, roughly 255% of American households did not have Internet access [Catherine McNally, 2021], and the numbers are much higher in other countries. Even where access is technically available, the cost of that access can be prohibitive.

Similarly, deeper investment in our educational infrastructure is absolutely necessary to increase access and digital sustainability. Currently, the partnerships between companies and universities consist of funding special research projects, offering internship opportunities, or industry-designed certification courses. But for many under-resourced schools, infrastructure limitations make it burdensome, if not impossible, to take advantage of the opportunities sent their way. Directing funding toward infrastructure has the potential to revolutionize the tech landscape.

Finally, we need to expand the umbrella of what counts as “tech” education. One of the great, democratizing elements of open source is that you don't need an undergraduate degree in order

to participate. Recognizing this, increasing numbers of employers are joining GitHub in shifting away from degree requirements (excepting, of course, degree- and certification-dependent roles in accounting, legal, etc.). We have an opportunity to redefine what makes a programmer and ask how we can grow the number of programmers, especially from marginalized communities, in a more equitable, sustainable way. To do so, educational efforts need to incorporate elements like resume-building, personal finance, the nuances of online and asynchronous communication, and interpersonal and leadership skills into our teaching models.

5. **There are opportunities for everyone in the chain.**

My first foray into DEI work was as an undergraduate student at North Carolina A&T State University. I count myself very privileged to have attended a historically black college and university (HBCU) that continues to be well-resourced and recognized for graduating the highest number of black engineers in the country. Hundreds of companies sent recruiters to our campus each year, and as a top-performing computer science student, I had an abundance of job offers before I even graduated. What I saw, though, was that these recruiters were overlooking some of the university's best programmers. These were fellow students and friends who were athletes or active military, who had hours-long commutes or full-time jobs in addition to their full-time coursework, and whose commitments made it hard for them to keep up a high GPA. Unlike me, they didn't have time to be in the computer lab all day every day – but that didn't mean they couldn't program circles around those of us who could.

To combat this issue, I launched a university-sponsored program called Aggie Engineering Ambassadors, which required recruiters to come to my organization in addition to career services. Aggie Engineering Ambassadors partnered with these companies to host get-togethers in informal spaces – roller-skating rinks, bowling alleys, and the like – where students could spend time and connect with recruiters, share their stories, and introduce themselves as multifaceted human beings. I'm proud to say many students came out of those interactions with internships.

The lesson here is that change feeds change, and amazing things can happen when you bring people together. If we're truly going to advance diversity and inclusion in tech – or anywhere – we'll need everyone at the same table. This project isn't just inspired by open source: it is open source. There's room, and a need, for participation from everyone: individuals, communities, companies, foundations, researchers, and beyond. Only then can we ensure we're finding and embracing the best talent, at a time when we need it most.

The Implementation

Collectively, the research-based findings outlined previously led me to create All In, a global ship to learn initiative focused on driving diversity and inclusion as an open source community, for an open source community. All In brings together corporate partners, universities, industry leaders, researchers, and foundations to collectively tackle barriers to access and success within open source for people from underrepresented backgrounds and regions.

When we launched in 2021, it was with a two-track pilot tailored to the demographics we identified as needing the most help: students and maintainers.

All In for Students

In the absence of universally functional school infrastructures, we knew we wanted to find other ways to ensure successful outcomes for our most vulnerable and marginalized students and contributors. At this time, not all computer science programs are created equal, and opportunities for students with great potential can be limited by financial constraints and access. All In for Students offers additional support and education through specialized instruction designed to offset inequalities in the current computer science education system.

Our 12-month pilot of All In for Students included stipends, technology resources, curricula to support technical education and career development, and individualized mentoring, all provided and facilitated by our growing chain of partner institutions. To begin, we turned to our seven founding university partners, all of which are minority-serving institutions, and worked closely with department chairs and professors to identify candidates across diverse lived experiences. Working directly with these

minority-serving universities ensured we'd be able to pragmatically address what individual schools and students actually needed.

We wanted to recruit the same students I remembered from my Aggie Engineering Ambassadors days, the ones whose academic experiences might be complicated by other obligations, be it commuting two hours to school each day because they couldn't afford to live on campus, providing caregiver support (especially due to COVID-19), working full time, serving as active-duty military, or training year-round as a student athlete to maintain their scholarships. I already knew these students were hardworking, dedicated, and motivated – they just needed an opportunity to thrive.

I also knew that we needed to start small: our initial cohort was just 30 students. When you're a well-resourced institution like GitHub, resisting the temptation to scale right out of the gate isn't easy. But massive programs rarely align with the realities of those who are under-resourced and can even exacerbate the problems they're trying to solve. It's those who are under-resourced who bear the burden of this misalignment, which causes burnout and stress. When programs are scaled up too quickly, we lose the ability to see people as individuals, and relationships and trust become that much harder to build.

It's sometimes said that people of color are over-mentored and under-sponsored, meaning that opportunities to receive guidance exist in far greater profusion than professional endorsements. I can tell you from personal experience that, for people of color, personal connections and relationships are absolutely vital to our success in tech. My first internship, the summer after my freshman year of college, was through a diversity program called Project Breakthrough, with IBM. There were only eight of us in that program, and I always say that, without that impact program, you probably wouldn't see me on this trajectory today. Most of the successful people of color I know in tech can also point not only to an impact program but to a specific person who was willing to put their time and reputation into doing personal outreach, writing testimonials, or otherwise advocating for their mentee.

With the significance of personal connection in mind, our students were paired with specialists at every phase of their journey with us. In our effort to expand the umbrella of tech education, students received a range of services, beginning with professional development training. Speakers covered topics ranging from career possibilities in tech to dealing with micro-aggressions to developing and owning their personal stories, each designed to equip our students for workplace success. Students also enrolled in online, self-paced courses designed to introduce them to the fundamentals of open source,

cloud infrastructure technologies, and DevOps. To address the isolation that sometimes comes with asynchronous learning, students had access to a professor for one-on-one instruction and a class forum to connect with other members of their cohort. Finally, students completed a ten-week open source project. Students were paid for their time and given the opportunity to work on teams whose members had established careers in tech while regularly meeting with a mentor to support them through the experience.

Throughout the program, each and every student worked closely with the All In team on a weekly basis. The goal was for these students to walk away from the program with a robust resume, a free laptop, and ultimately an internship with one of our corporate partners. Of the 24 students who completed the program requirements, 100% went on to intern with leading partner companies including GitHub, Red Hat, Microsoft, Fidelity, Intel, and Cisco.

As we look toward the future of All In for students, it's with an eye toward responsible growth that continues to bridge the mentorship-sponsorship divide: something we believe is achievable by growing our team of All In liaisons and facilitators as we continue to expand our corporate and university partners.

All In for Maintainers

Working with students allows us to find and nurture more diverse talent to join our communities, but it's equally critical that the communities they enter are adequately resourced to provide a safe and nurturing environment to members of all backgrounds. In 2021, we also set the foundation for the All In for Maintainers program, aimed at providing training and technical support to community leaders who want to advance diversity and inclusion but lack the resources to do so.

We already encourage maintainers to take small but impactful steps, like issuing an automated message welcoming new contributors to the community, letting them know that their voice is appreciated and their contributions are valued, and explicitly telling them ways they can contribute meaningfully no matter their experience level. To that end, we're working with our partners to fund grants for maintainers from marginalized and/or historically excluded backgrounds and for projects that serve under-resourced communities.

Many of our maintainers have explained that they don't need more information, per se – during our listening tour, many touched on the overwhelming volume of resources, talks, podcasts, and digital checklists. Instead, they need a way to sort through those

resources and implement the most relevant recommendations, to know where to go to address a particular issue when it's actually happening in their community. To provide them with a clear pathway, we're in the process of building an open source hub of DEI resources that's intuitively organized, navigable, and, most importantly, actionable.

To distinguish the maintainers and communities who have committed to doing this DEI work, we've become a founding partner to build a DEI badging program with the CHAOSS Project. We'll be working with participants in this program to provide actionable recommendations, direct technical assistance, and even financial support.

Conclusion

At the outset of this chapter, I argued that diversity, equity, and inclusion efforts would benefit as much from open source principles as open source communities would benefit from greater diversity, equity, and inclusion. We launched All In to facilitate growth on both these fronts, to feed two birds with one seed. In our first year, I'm proud to say that we brought on 17 founding partners and seven partnering educational institutions and reached more than 1,000 community members. This year alone, we've expanded All In to reach 300 students from 79 HBCUs, Hispanic-serving institutions (HSIs), and community colleges, and we're on track to reach over 5,000 community members by the end of 2025. We've done our initial research, and we've set the foundation to bring the revolutionary qualities of open source to diversity, equity, and inclusion efforts in the tech industry. Now, it's time for us to spread our wings.

One statistic that jumped out at me from our survey was the 89% of respondents who feel they can have a positive impact on the world. This tells me that the open source community is suffused with optimism and motivation. I have no doubt that we are collectively capable of achieving our mission. I want to invite you to join us in open sourcing diversity, equity, and inclusion and take part on this journey to make a lasting impact together. Much as open source has revolutionized more than just technology, open sourcing DEI has the potential to revolutionize far more than just the open source community. Whether you're a developer, a CEO, a student, a fellow DEI professional, a researcher, or an activist, there's a place for you at <https://allinopensource.org>. I know I'm All In. GitHub is All In. Are you?



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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.

PART V

Challenges and Initiatives to Designing Inclusive Software Engineering Education Environments

CHAPTER 23

Rethinking Gender Diversity and Inclusion Initiatives for CS and SE in a University Setting

Jocelyn Simmonds, University of Chile, Chile.*

María Cecilia Bastarrica, University of Chile, Chile.

Nancy Hitschfeld, University of Chile, Chile.

Pablo Villar Mascaró, University of Chile, Chile.

Diego S. Wistuba La Torre, University of Chile, Chile.

Jo Muñoz Montenegro, University of Chile, Chile.

Millaray Cárdenas, University of Chile, Chile.

In this chapter, we summarize lessons learned from gender-related initiatives conducted at the Computer Science Department (CSD) of the University of Chile, located in Santiago, Chile.

Introduction

The lack of gender diversity in the software engineering (SE) industry in South America is well documented [4, 7], which is usually attributed to the low number of female students that apply to undergraduate computer science (CS) and SE programs. In fact,

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

out of the Science, Technology, Engineering, and Math (STEM) fields, CS and SE are the least gender-diverse, with representation hovering around 10–15%. Acknowledging this situation, there has been an explosion of programs and activities promoting women in computing (WiC), seeking to create spaces for women to network among themselves and increase their confidence in their own work. We have previously reported about the impact of one such initiative, an admissions program for women at the University of Chile, which has one of the most competitive science and engineering programs in Chile [1, 9]. Thanks to this program, a third of our undergraduate students are now women, up from 19% in 2014, and several local universities have adopted similar programs.

Recognizing that diversity in STEM is important, our campus now has a Diversity and Gender Office, which is in charge of making crosscutting changes to improve diversity on campus regarding gender, sexual orientation, and indigenous peoples, mainly. This office has recently been awarded the United Nations Development Programme's Gender Equality Seal.¹ Obtaining this seal meant defining dimensions, actions, and metrics to help promote gender equality, like creating gender taskforces at a department level. Since our department (CSD) has historically had one of the lowest percentages of female students, the first two authors of this chapter were asked to pilot the creation of one of these taskforces. We invited current and former students, administrative staff, and faculty members to participate in the taskforce, given the broad nature of the issue that we were asked to tackle.

The CSD Gender Taskforce started by gathering information from different sources, trying to understand where we stood in terms of diversity. This led to the discovery of a hidden reality: our students are much more diverse regarding gender and sexual orientation than we had originally imagined. The undergraduate CS student union carried out a survey about gender identity and sexual orientation, with almost 180 responses (approximately 30% of our students). Almost 8% of the respondents identified themselves as non-binary or trans and 33% as non-heterosexual. In other words, although the department's work on improving the participation of women in computing is crucial, it is too narrow and ignores other sources of discrimination, like gender nonconformity and sexual orientation.

Rodriguez-Perez et al. [8] carried out a systematic literature review about perceived diversity in software engineering, identifying five dimensions of diversity that have been studied by the SE community: gender, nationality, age, race, and combinations of the previous four. Gender was by far the most studied dimension, but as we had also done until this study, most of these works assumed gender as a binary construct. The only

¹www.genderequalityseal.org/programme/

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

exceptions are the works published by Ford et al. [2] and Prado et al. [6], which focus on the experiences of transgender developers. Ford et al. [2] studied whether transgender developers really felt more included in the development process under remote working conditions, and Prado et al. [6] developed a set of recommendations for hackathons that want to be more inclusive of transgender people.

In order to rethink the diversity and inclusion initiatives at a department level, we carried out a qualitative study, looking to identify issues that hinder gender diversity. The contributions of this study (and chapter) are the following:

1. Provide evidence that, although focusing on improving female participation is great, there is a broader, more systemic problem of lack of representation in our field, and any diversity and inclusion efforts undertaken by universities must be intersectional.
2. Serve as a local wake-up call. Students that identify as diverse use different strategies to fit in on campus, like creating peer support groups. However, we are not addressing the basic issues that may affect their learning experiences and that can put them at a disadvantage compared with their peers.
3. Provide recommendations for CS and SE departments that are looking to become more diverse and inclusive, based on the experiences and issues that affect students that identify as part of the LGBTIQ+ community, as well as our previous experiences.

Establishing a Departmental Gender Taskforce

Figure 23-1 shows a timeline of the main actions the CSD has been involved in to improve the participation of women in computing in Chile and Latin America. From 2011 to 2016, we focused on bringing together women in computing, with panels and conferences^{2,3} helping set up affirmative action programs on campus,^{4,5} as well as

²<https://chilewic.cl/>

³<https://latinity.info/>

⁴<https://ingenieria.uchile.cl/admision/ingresos-especiales/cupos-equidad-de-genero>

⁵<https://ingenieria.uchile.cl/noticias/103151/fcfm-pondra-en-marcha-programa-de-equidad-de-genero-en-la-academia>

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

establishing outreach activities to empower girls and young women.⁶ From 2017 the focus has been on consolidating these initiatives and helping establish the Diversity and Gender Office⁷ on campus. Work on the Gender Equality Seal began in 2020,⁸ and the CSD Gender Taskforce is a part of this initiative.



Figure 23-1. Timeline of the gender-related initiatives organized by or involving CSD members

The goal of this taskforce is to recommend actions that the CSD can take to become more diverse and inclusive. Participants include current and former students, administrative staff, and faculty members, and meetings are held on a regular basis every trimester. The taskforce was asked to collect data related to gender in the CSD and then propose actions to improve on this baseline diagnostic. We collected gendered data about our undergraduate and graduate programs, teaching teams, etc. The undergraduate student union carried out the survey mentioned in the introduction, finding that 8% of the almost 180 respondents identified as non-cisgender and 33% as non-heterosexual. Since our campus has been a historically male-dominated space in a socially conservative country, we wanted to explore the issues that these students experience on campus. To do this, we organized a focus group, so as to draw upon respondents' attitudes, feelings, beliefs, experiences, and reactions in a way where other methods are not applicable.

⁶<https://ninaspro.cl/>

⁷<https://ingenieria.uchile.cl/sobre-la-fcfm/estructura/direcciones/direccion-de-diversidad-y-genero>

⁸<https://ingenieria.uchile.cl/noticias/184359/fcfm-recibe-el-sello-general-igualdad-del-pnud>

Study Design

All the authors of this chapter participated in the organization of the focus group. The first three authors are faculty members. Also, the second and third authors are recent department chairs and as such are highly familiarized with the departmental academic and administrative processes. They also helped design and implement the campus Gender Equality Admissions and Faculty programs that appear in Figure 23-1. The first author co-founded the WiC conferences that appear in the same timeline. The remaining authors are undergraduate and graduate students that volunteered to help carry out this study.

We defined two main topics that we wanted to explore with the participants, both with a gender perspective: (1) the learning environment on campus and (2) the use of campus spaces. The first topic was further broken down into three subtopics – classroom experience, courses, and internships – and the second topic into two – the use of classrooms and labs and the common areas, like cafeterias and restrooms. We posted a call for participation on the CSD student forum, inviting students that identified as members of the diverse CSD community to sign up. We got 11 responses, but only 6 were available on the date of the focus group.

The focus group was moderated by one of the authors, assisted by another author, a CSD student that identifies as a part of the diverse community, who observed, took notes, and asked clarifying questions. The focus group was carried out in person and lasted two hours, where participants first signed an informed consent, and we then asked them for permission to record (only audio). We began the focus group by asking participants to introduce themselves. They wrote their preferred pronouns on their place cards and told us how long they have been students at the CSD, telling us what they liked about computing and whether they have been teaching assistants (TAs) in CSD courses. We then moved over to the focus group topics, asking them to talk about their experiences as gender- and/or sexually diverse students on campus.

The focus group transcript and observation notes were coded by one of the authors, following general open coding guidelines. The rest of the authors revised the coding process and the later translation (as the focus group was carried out in Spanish). The participants were divided evenly by the number of years spent at the CSD (two participants in their first year, two in their second year, and two in their third year). Two of the participants identify as non-binary, and the rest identify themselves as women. None of them have been TAs in CSD courses.

Findings

We now report the main findings of our focus group, grouped by main topic. Positive experiences and perceptions are labeled with a plus sign and negative ones with a minus sign.

Regarding the Learning Environment

LE1 – Faculty attitudes and behaviors: The participants indicated that they have a good relationship with the CSD professors. Their experience with CSD courses is starkly different from the initial math and physics courses, where they felt that professors want to show off how much they know. CSD professors answer student questions outside the classroom and try to build a trusting environment in the classroom. For example, they ask students for their preferred pronouns and use inclusive language (+). Professors correct themselves if they make mistakes, so the participants perceive an effort to improve (+). They did not feel that teaching assistants (TAs) made the same effort (–). Member checking revealed various possible reasons, ranging from ignorance/disinterest to a lack of empowerment on behalf of the TAs.

LE2 – Peer attitudes in the classroom: CSD has almost doubled in enrollment in a short time, but participants perceive that it is a horizontally interrelated community. The gender-diverse subcommunity is organized, and they expressed strong feelings of friendship and camaraderie, even if they do not know every member of the subcommunity personally (+). However, they have identified a group of cisgender heterosexual (cishet) male students that prefer to avoid social interactions with the CSD community in general, since they seem to be uncomfortable with or not interested in diversity (–). For example, some of these students have made fun of people for using inclusive language (–).

LE3 – General perspectives about CSD courses: Unlike math and physics courses, CSD courses focus more on homework and teamwork (+). Support material from previous semesters is available (+), but not always of the best quality (–). Some professors could make better use of the Learning Management System (LMS) used on campus (–), and professors should avoid making unilateral changes to course evaluations, like adding or removing an assignment (–).

LE4 – Gender-diverse perspective in CSD courses: Assignments and midterms are written in a neutral manner (+), using animals or groups of people when characters are needed for illustration (+). Professors avoid using only traditionally male names in

examples (+). However, courses do not usually include content created by members of any historically marginalized groups, not only gender, nor are their contributions highlighted (-). For example, one participant indicated that they knew more about WiC from volunteering at a coding camp for girls than from the CSD.

LE5 – Perceptions about their own learning abilities: The participants indicated that, as part of the diverse community at CSD, they felt like they cannot fail, so they must make a larger effort than their cishet male peers to show that they are just as good as students (-). Since most of the CSD students are cishet men, the participants also felt the constant need to compare themselves or compete with the cishet men students (-). They feel an implicit pressure in being a woman or part of the diverse community (-).

LE6 – General perspectives about internships: The COVID-19 pandemic delayed the internship process (-). The recent changes to procedures for reporting internships to receive credit are also confusing/unclear (-). They feel like the CSD provides little guidance on how/where to look for internships, how to prepare a CV, etc. (-). They know professors can help them with the internship search process (+), but shyness prevents them from contacting them (-).

LE7 – Gender and sexual diversity at internships: The transition from the CSD to the work environment is difficult (-). They have to move from a diverse community to an environment where they are usually the only person that identifies as gender- and/or sexually diverse (-). If there are more women at the company, they are usually not engineers (-). Some participants mentioned that they perceive that they are treated differently than the rest of the people at the company (-) and that they feel that they have to show they know more than a cishet male intern (-). One participant was more empowered and asked interviewers about their company's diversity policies and statistics (+). Interviewers usually show little understanding of gender identities and sexual orientation (-). Another participant mentioned a case where the technical interview turned into a conversation about the gender transitioning process (-).

Regarding the Use of Campus Spaces

CS1 – General perspectives about common spaces: The participants shared various reasons why they were sometimes uncomfortable in common spaces. They feel intellectually insecure and want to avoid being “wrong” in public, like making coding mistakes (-). Since they feel watched when working on a computer, they prefer to use computers that face the wall (-). They are also afraid or embarrassed to use whiteboards

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

to work out problems (-). They also get unsolicited advice or help when coding or solving problems (-). They find it hard to shut down these unsolicited offers; some of those that offer help are oblivious to the discomfort caused by these interactions, while others assume that their help is needed (e.g., “mansplaining”) (-). Some of the older participants indicated they gained confidence over time (+) and that they now feel that they can ask for help or even offer help themselves (+). The general consensus was that cishet men do not seem to be afraid of making mistakes in public, so they like to flaunt what they know in common spaces like study halls and labs (-). These cishet students are also comfortable explaining course topics to their peers and frequently speak over women and students that identify as diverse (-). Women and other students that have a hard time with a course prefer to study in the library, since it has cubicles where they can work in a more sheltered manner (-). The participants that identify as women reiterated that they felt watched in the computer labs (-).

CS2 – About specific spaces in the CSD: Participants feel like part of their insecurities are ingrained (-), but that the vibe at the CSD helps reduce their insecurities (+). Their experience during the first two common years of math and physics courses was very different, where they overheard some hallway conversations against diversity (-). At the CSD, the participants indicate that it is not clear which spaces can be used by students (-), since in-person classes only restarted in March 2022, after two years of virtual classes. Students use mainly two rooms at the CSD: (1) an open layout computer lab, which has computers along the walls, with work tables in the middle, and (2) a smaller room with lounges, a TV, and video game consoles. Both spaces are sometimes filled with cishet men, and then some people that identify as diverse prefer to not use these spaces (-). However, when they see someone who identifies as diverse using these spaces, they gain confidence and start using these spaces more (+), getting to know more people that identify as diverse in the process (+).

CS3 – Restrooms: Restrooms are a complex issue. When the original campus buildings were built (1911–1922), there were only a handful of women students on campus, so no restrooms were built for women. This situation has improved over the years, and we now have similar amount of men’s and women’s restrooms. However, today we have trans and non-binary students who do not feel safe using these binary restrooms (-). One of the women’s restrooms was first made accessible and has been recently made gender neutral (+). However, as it is in one of the oldest buildings on campus, it usually has problems with plumbing, and it is in a highly transited area, which severely limits its use (-). The non-binary participants indicated that they feel

safer using women's restrooms (+), but that they worry about offending other students when using them (-). One of the non-binary participants was stopped from using a restroom by a security guard, since their gender expression did not "match" the restroom (-). Locker rooms are also binary (-). As a result, non-binary and trans students avoid using the restrooms on campus (-). Students on another campus proposed removing the urinals from the men's restrooms, in order to make them more gender neutral. This proposal was met with ridicule by male cis het students on our campus (-). There is no access to hygiene products for menstruating people on campus (-), while there are free condoms at the campus health center. Finally, the CSD restrooms are also binary (-).

CS4 – Virtual spaces: This topic was not planned and emerged spontaneously at the end of the focus group. On campus, we use a local LMS that has a campus-wide discussion forum, among other communication tools. The participants find that the campus forum is useful (+), but that it can also be a toxic space, where trolling, harassment, and other negative behaviors are enabled and/or tolerated (-). Cis het men dominate conversations, without necessarily making positive contributions (-). Some people use the forum to start controversies (-), and there are no moderation mechanisms in place (-). Users are only identified by name (profile photos are optional), so students with common Spanish names are anonymous in practice (-). As a result, the participants indicate that posting their opinions on this forum makes them anxious (-). They do not want to draw attention to themselves, since it may lead to harassment on their social network accounts (-), which has happened in the past.

Recommendations

Based on these findings, we propose six recommendations for CS/SE departments working on becoming more diverse and inclusive. The goal is to create safe and welcoming learning environments that promote inclusiveness, since software engineers that study in these spaces need to be aware that their potential users and future teammates are far more diverse than the demographics of their current programs. We are in the process of implementing these recommendations, together with those already identified by the CSD Gender Taskforce, like diversifying teaching teams.

R1 – Carry out awareness campaigns within the CSD: Motivated by findings LE1, LE2, and CS2. Concepts like gender identity and sexual orientation are under constant revision, so recurring campaigns about these topics for professors, teaching assistants, and students should be the norm. Spanish is a gendered language that defaults to

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

the masculine gender, so these campaigns should also focus on the use of inclusive language. Also, since the community is constantly changing, the importance of these topics should be communicated clearly by department authorities. The community should also be familiar with protocols defined at a central level, like protocols for handling harassment cases. A study by Garvey et al. [3] found that both faculty members and administrative staff play an important role in enhancing campus climate, for example, by implementing plans to deal with incidents of harassment and/or discrimination. Students that perceived a warmer campus climate rated their academic success as higher.

R2 – Rethink CSD courses with a more inclusive lens: Motivated by findings LE1, LE3, and LE4. Contributions from historically marginalized groups are not usually highlighted in CSD courses, and professors should make an effort to include contributions from these groups in their course materials. Professors should also avoid making value judgments about course evaluations and grade averages, since certain groups of students, such as women and/or those that identify as diverse, can take these comments very personally, affecting their self-confidence and sense of belonging in CS/SE. Similar to R1, this recommendation also focuses on creating a warmer climate, now specifically in the CSD.

R3 – Foster student self-confidence and job readiness: Motivated by findings LE5, LE6, and LE7. All of the participants mentioned feelings of insecurity, anxiety, shyness, as well as a lack of self-confidence in their knowledge of course materials. This affects how they use common spaces, how they participate in group and course discussions, etc. As such, the CSD should develop workshops to help students manage these feelings and improve their self-confidence. We should also focus on job readiness, since the transition from the CSD, where diverse students have a support group, to industry, where they are usually the only person that identifies as diverse, can be tough. This involves helping them identify internship opportunities, put together a CV, etc. This is consistent with existing work, like a study carried out by Wang et al. [10] that found that female developers on GitHub have a competence-confidence gap regarding their participation in new projects. They also found that female developers first build up a reputation as helpers on projects, before attempting to contribute to the more technical parts of a project.

R4 – Revise the use of campus spaces: Motivated by findings CS1, CS2, CS3, and CS4. After two years of online courses, students are just starting to rediscover the campus. As such, we should work on clarifying and promoting the use of common

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

spaces. This includes defining codes of conduct, helping students identify desirable and undesirable personal interactions in these spaces, both physical and virtual. We also recommend reviewing the layout of labs and classrooms, so that there are spaces where students who feel the need to work unobserved can do so. Finally, students that identify as diverse should feel safe going to the restroom on campus, so more neutral restrooms should be made available. Hygiene products for people that menstruate should also be available. The decline of “lab culture” was already an issue before the COVID-19 pandemic, as students started to use their own laptops instead of shared workstations on campus [5]. Students worked from home during the worst of the pandemic, and it is now time to focus on rebuilding supportive and collaborative learning environments, both in person and virtually.

R5 – Work with campus authorities on general issues: Motivated by findings LE1, LE2, CS1, CS3, and CS4. Several issues raised by the focus group participants should be handled by campus authorities, specifically a Gender and Diversity Office, which should tackle these issues with a gender-diverse perspective. We need to raise awareness that binary restrooms and changing rooms have a high impact on non-binary and trans students. We are also concerned about student conversations against diversity and women in common spaces, as well as professor attitudes during the first two years of common courses. These are all issues that must be strategically handled by campus authorities. In line with the recommendations made by Garvey et al. [3], best practices to improve the campus climate should be institutionalized, seeking to build a campus climate that is proactive and social justice oriented.

R6 – Work toward creating inclusive workplaces in industry: Motivated by findings LE6 and LE7. Students raised several concerns relating to internships. Although the CSD cannot directly intervene these spaces, we can raise awareness about measures that companies can take to create a more diverse and inclusive work environment. This requires a stronger commitment than just changing hiring practices; it also requires defining or examining codes of conduct, protocols for handling harassment in the workplace, etc. Also, students cannot be responsible for educating company employees on topics like gender identity and sexual orientation; they are there to improve their own technical and professional skills. The negative experiences about internships that our focus group participants described are similar to some of the challenges that have been found to push women away from the tech industry (see Chapter 4, “Breaking the Glass Floor for Women in Tech”). As such, our recommendation is aligned with the strategies that the authors of that chapter have identified to address these challenges.

Conclusions

Gender- and/or sexually diverse students face a myriad of challenges. In the case of our students, they are also studying engineering at a highly competitive science and engineering program in Latin America. Chile is a socially conservative Latin American country, and Spanish is a gendered language, where the default gender is masculine. This means that challenges that affect them need to be addressed in an intersectional manner, meaning that we have to be aware of the different types of diversity that are present in our community when designing and undertaking new diversity, equity, and inclusion policies on campus. The recommendations made in this chapter are in the process of being adopted within our department. For example, we recently created a gender-neutral restroom. We hope these recommendations may also serve as guidance for other CS and SE departments motivated to address this topic.

Bibliography

- [1] María Cecilia Bastarrica, Nancy Hitschfeld, Maíra Marques Samary, and Jocelyn Simmonds. Affirmative action for attracting women to STEM in Chile. In Erika Ábrahám, Elisabetta Di Nitto, and Raffaela Mirandola (editors), *2018 IEEE/ACM 1st International Workshop on Gender Equality in Software Engineering, GE@ICSE, Gothenburg, Sweden, May 28, 2018*, pages 45–48. ACM, 2018.
- [2] Denae Ford, Reed Milewicz, and Alexander Serebrenik. How remote work can foster a more inclusive environment for transgender developers. In Ivica Crnkovic, Karina Kohl Silveira, and Sara Sprenkle (editors), *Proceedings of the 2nd International Workshop on Gender Equality in Software Engineering, GE@ICSE 2019, Montreal, QC, Canada, May 27, 2019*, pages 9–12. IEEE/ACM, 2019.
- [3] Jason C. Garvey, Dian D. Squire, Brett Stachler, and Susan Rankin. The impact of campus climate on queer-spectrum student academic success. *Journal of LGBT Youth*, 15(2):89–105, 2018.

CHAPTER 23 RETHINKING GENDER DIVERSITY AND INCLUSION INITIATIVES FOR CS AND SE IN A UNIVERSITY SETTING

- [4] Jeongeun Kim and Sergio Celis. Women in STEM in Chilean Higher Education: Social movements and institutional transformations. In *Gender Equity in STEM in Higher Education*, pages 105–120. Taylor and Francis Inc., 2021.
- [5] Deepak Kumar. REFLECTIONS Lab Culture versus Hackathons. *ACM Inroads*, 10(3):16–18, August 2019.
- [6] Rafa Prado, Wendy Mendes, Kiev Santos da Gama, and Gustavo Pinto. How trans-inclusive are hackathons? *IEEE Softw.*, 38(2):26–31, 2021.
- [7] Diana Katherine Pérez López. “Am I not good at it, or am I not good at all?” STEM gender gaps in higher education. Master’s thesis, Universidad de los Andes, Colombia, 2021.
- [8] Gema Rodríguez-Pérez, Reza Nadri, and Meiyappan Nagappan. Perceived diversity in software engineering: a systematic literature review. *Empir. Softw. Eng.*, 26(5):102, 2021.
- [9] Jocelyn Simmonds, María Cecilia Bastarrica, and Nancy Hitschfeld-Kahler. Impact of affirmative action on female computer science/software engineering undergraduate enrollment. *IEEE Softw.*, 38(2):32–37, 2021.
- [10] Zhendong Wang, Yi Wang, and David F. Redmiles. Competence-Confidence Gap: A Threat to Female Developers’ Contribution on GitHub. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society, ICSE (SEIS) 2018, Gothenburg, Sweden, May 27–June 3, 2018*, pages 81–90. ACM, 2018.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 24

Economical Accommodations for Neurodivergent Students in Software Engineering Education: Experiences from an Intervention in Four Undergraduate Courses

Grischa Liebel, Reykjavik University, Iceland.*

Steinunn Gróðardóttir, Reykjavik University, Iceland.

Neurodiversity is an umbrella term that describes variation in brain function among individuals [1], including conditions such as autism spectrum disorder (ASD), attention deficit hyperactivity disorder (ADHD), or dyslexia. Neurodiversity is common in the general population, with an estimated 5.0–7.1% [19, 21] and 7% [17] of the world population being diagnosed with ADHD and dyslexia, respectively. Neurodivergent (ND) individuals often

experience challenges in specific tasks, such as difficulties in communication or a reduced attention span in comparison to neurotypical (NT) individuals [6]. However, they also exhibit specific strengths, such as high creativity [20] or attention to detail [2]. Therefore, improving the inclusion of ND individuals is desirable for economic, ethical, and talent reasons.

In higher education, struggles of ND students are well-documented [7, 8, 18]. Common issues in this area are a lack of awareness among other students and staff, forms of assessment that are particularly challenging for some students, and a lack of offered accommodations. These factors commonly lead to stress, anxiety, and ultimately a risk of dropping out of the studies.

Accommodations for ND students can require substantial effort. However, smaller changes in course material can already have major impact. In this chapter, we summarize the lessons learned from an intervention in four courses in undergraduate computer science programs at Reykjavik University, Iceland, over a period of two terms. Following accessibility guidelines produced by interest groups for different ND conditions, we created course material in the form of slides and assignments specifically tailored to ND audiences. We focused on small, economical changes that could be replicated by educators with a minimal investment of time. We evaluated the success of our intervention through two surveys, showing an overall positive response among ND students and neurotypical (NT) students. Example materials we produced are available at <https://doi.org/10.5281/zenodo.7199162>.

In summary, in this chapter you will learn

1. Which obstacles ND students commonly face in higher education
 2. What accommodations are recommended for ND students
 3. What economical interventions for computer science students can look like
 4. What feedback we received on making such an intervention
-

The State of Knowledge in Research

Work on neurodiversity is broad and covers different areas. In addition to work within the realms of medicine and psychology, there are many studies investigating neurodiversity in higher education, neurodiversity and employment, and piloted

Neurodiversity in higher education and industry

1. ND students often underperform in a specific task, due to the task's nature.
 2. Awareness and accommodations for ND students are commonly lacking.
 3. ND students often feel anxious, depressed, or afraid of being stigmatized.
 4. ND individuals in industry often choose to not disclose their condition.
 5. Accommodations for ND individuals in industry vary considerably.
 6. There is only initial work on neurodiversity in software engineering.
-

ND conditions are often diagnosed in early childhood. Therefore, much of existing work targets primary and secondary school education. As inclusion of ND students improves in these areas, the amount of ND students enrolling in higher education increases. That is, educators in higher education need to increase their awareness of neurodiversity and possible accommodations for ND students [9]. In particular, educators should understand that ND students make mistakes or underperform in certain tasks due to the tasks' nature, not due to a lack of intelligence.

Experiences of ND students in higher education are published in several studies. A summary of research themes surrounding neurodiversity in higher education is presented by Clouder et al. [4], based on 48 screened publications. The findings show that many higher education institutes provide insufficient support for neurodiversity, primarily due to “low levels of staff awareness, ambivalence and inflexible teaching and assessment approaches.” ND students often feel frustrated due to the lack of available accommodations, anxious or frightened due to challenging situations or stigmatization, and often embarrassed to ask questions. Finally, inclusive and trusting environments, as well as accommodations such as extended exam times or flexible assignment schedules, benefit ND students.

Experiences and support of college and university students with autism are summarized in a systematic literature review (SLR) by Gelbar et al. [7]. The authors emphasize the need for support in higher education and that anxiety and depression among autistic students are common.

Regarding students with dyslexia in higher education, Pino and Mortari [18] summarize existing research, finding that research is fragmented and contains many gaps. The article reports coping strategies such as getting support from family and friends or adapting their writing styles for written assignments. Furthermore, the authors highlight a common lack of awareness and acceptance by academic staff and mention that accommodations are usually appreciated by students.

In the IT industry, inclusion of ND individuals is typically at an early stage, and companies need to become more welcoming and inclusive [22]. Similar to higher education, ND individuals often do not disclose their condition due to fear of stigmatization [10, 16]. However, depending on the workplace, the number of ND individuals disclosing their condition and the offered accommodations differ considerably.

Driven by the COVID-19 pandemic, several studies investigate how remote work affects ND individuals. Das et al. [5] find that ND individuals create accessible workplaces at home, negotiate communication and meeting practices, and balance tensions between productivity at work and fatigue. The authors suggest to commonly record audio and video, enable automated meeting transcripts, and make meeting notes available routinely. In a similar direction, the needs of adults with ASD in video calling are studied by Zolyomi et al. [23]. The authors find that adults with ASD develop several coping strategies, such as adopting neurotypical behavior, and that they experience more stress than NT individuals without adaptations. The authors provide suggestions for adaptations, for example, translating and communicating social and emotional information to adults with ASD.

Specific to software engineering, there is only initial work on neurodiversity and potential accommodations for ND individuals. Morris et al. [16] investigate challenges faced by ND individuals. Specific themes related to SE are the tendency to get bored with mundane tasks or expressing inappropriate emotions, for example, when criticized during code reviews. Furthermore, ND individuals perceive several tasks more challenging than NT individuals, for example, working in shared offices or noisy settings or deciding when to seek help for tasks. Begel et al. [3] report the results of a successful 13-day game programming camp for incoming college students with ASD. In addition

CHAPTER 24 ECONOMICAL ACCOMMODATIONS FOR NEURODIVERGENT STUDENTS IN SOFTWARE ENGINEERING EDUCATION: EXPERIENCES FROM AN INTERVENTION IN FOUR UNDERGRADUATE COURSES

to teaching game development, students were instructed specifically in communication skills. In a line of research investigating how dyslexic individuals read and comprehend software code, McChesney and Bond [11, 12, 13, 14, 15] find that, contrary to intuition, dyslexic individuals do not exhibit over-proportional deficiencies compared with NT individuals. Potential explanations are that reading code is significantly different from regular text, for example, due to indentation or spacing.

Possible Accommodations for Neurodivergent Students

Several guidelines on how to accommodate ND students have been published by interest groups for different neurodivergent conditions. These guidelines typically target one specific condition at a time. Deciding on possible accommodations for a class or a university therefore requires synthesizing these guidelines into one set of concrete accommodations that somehow align well with several existing guidelines. This is a difficult step, given that existing guidelines might contradict each other and might propose too many accommodations for a feasible intervention in practice. In the following we will summarize three guidelines, one for each of the most common ND conditions: ASD, ADHD, and dyslexia.

Accommodation areas for different ND conditions

1. **ASD:** Sensory environment, communication, escape ways, awareness
 2. **ADHD:** Patience and understanding, clear structure, communication
 3. **Dyslexia:** Style and presentation of visual material
-

For ASD, we consider a checklist for autism-friendly environments published by the NHS.¹ The checklist describes accommodations to four areas: the sensory environment, for example, reducing strong colors or overly patterned surfaces or reducing smell and noise; the communication environment, for example, the use of clear and unambiguous

¹https://positiveaboutautism.co.uk/uploads/9/7/4/5/97454370/checklist_for_autism-friendly_environments_-september_2016.pdf

For ADHD, we consider a report by the ADHD Foundation on how to teach and manage students with ADHD.² The report provides a general introduction to ADHD and to common behaviors of individuals with ADHD. Then, suggestions are made on how classrooms can be made more accessible to students with ADHD. First, teachers are encouraged to show patience and understanding for behavior that might seem odd, such as excessive movement or inappropriate comments. Second, providing a structure and flexibility is important, for example, through regular routines or by providing overviews before starting a class or a checklist for an assignment. Finally, the report lists a few hints on communicating with students with ADHD, for example, trying to tell the student what they should do instead of what not to do or addressing them by name.

Finally, for dyslexia, we consider the style guide by the British Dyslexia Association.³ This guide focuses on accommodations in the style of visual material, for example, assignment texts or lecture slides. Some of the changes they recommend are using fonts that are easier to read for dyslexic individuals, for example, Arial and Comic Sans;⁴ increasing font size; increasing spacing between letters, words, and lines; using left-aligned instead of justified text; avoiding multi-column texts; and using cream-colored backgrounds instead of plain white. Additionally, adapting the writing style is recommended, for example, by avoiding passive voice, using concise sentences, or avoiding jargon. Many of these recommendations overlap with traditional style guidelines for presentations, for example, using large font sizes and avoiding red and green as colors, and for academic writing, for example, avoiding overly long sentences and jargon.

An Example Intervention in Four Courses

Based on the guidelines and the existing academic work outlined in the previous section, we designed an intervention in initially three courses, which we were teaching in the fall term 2021 at Reykjavik University, Iceland. In the spring term 2022, we used

²www.adhdfoundation.org.uk/wp-content/uploads/2022/03/Teaching-and-Managing-Students-with-ADHD.pdf

³www.bdadyslexia.org.uk/advice/employers/creating-a-dyslexia-friendly-workplace/dyslexia-friendly-style-guide

⁴Yes, you read that correctly.

the intervention in a slightly altered way in one more course. All four courses are part of the undergraduate computer science and software engineering programs in the first and second year at Reykjavik University. The courses are an introductory software engineering course, an introductory web development course, and two courses on discrete mathematics. During the interventions, we were operating in a hybrid mode of on-site teaching and remote teaching through video conferencing tools. Prior to our intervention, the only accommodations ND students could receive were extended exam times, usually limited to the final exams only. Approximately 10% of the students at Reykjavik University have a diagnosed ND condition and applied for this extension, though the percentage was higher in some of the four courses.

The accommodations were made as part of the regular teaching. That is, there was no specific time, budget, or workforce assigned to making the chosen accommodations. Therefore, we focused on small, economical changes that we could implement under regular time pressure. In summary, we made the following changes.

Implemented accommodations

1. Emphasis on style changes in assignments and lecture slides
 2. Improved clarity and structure
 3. Minor changes in the sensory environment, for example, consistent lighting
 4. Discussing neurodiversity and the accommodations to raise awareness
-

Specifically, we decided to primarily change the style of our assignments and slides. We changed font type to OpenDyslexic⁵ and the background color to a cream color; increased inter-letter, intra-letter, and line spacing; and avoided italics, multiple columns, and specific colors, such as red and green, or strong colors in general. We did not change the default fonts in the case of mathematical formulas and program code. To leave the choice of style up to the students, we offered assignments both in a traditional style and in our updated, ND-friendly style.

In terms of clarity and communication, we added bullet-point summaries to all assignments. Additionally, we proofread the assignments and tried to reduce ambiguity, jargon, and unclear acronyms and improve overall clarity. Examples of the slides

⁵<https://opendyslexic.org>

In addition to adapting the style and content of our slides and assignments, we made minor changes to the sensory environment, for example, avoiding clothes with strong patterns or colors, and in remote teaching using high-quality microphones and consistent artificial lighting. Finally, we discussed the changes and the reasons behind them in the classroom, with the intention to raise awareness for neurodiversity and make ND students feel seen and welcome.

Overall, the accommodations were efficient. For assignments, we used LaTeX and thus only had to recompile the assignment with slightly changed parameters. Re-reading assignments and slides and improving clarity were done as part of updating existing assignments, which we argue should be part of preparing assignments in any case. Changing the sensory environment primarily required us to be conscious about potential issues, such as avoiding clothes with strong patterns or colors. Finally, the largest effort was required for updating lecture slides in Microsoft PowerPoint. While we made style changes on the slide master level, that is, globally for all slides, increased font size and spacing required changes to individual slides to fit all content. To update existing slides took approximately 30 minutes per slide deck (per lecture). This effort is likely much lower if you create lecture slides from scratch and not, as in our case, update existing ones.

Through two surveys, one after each term, we evaluated the impact our intervention had on all students in our courses. We received answers from 169 students in all courses, corresponding to an overall response rate of 23.57%. Of those who answered, we only considered those that completed the survey, resulting in 155 valid answers. Out of the 155 students, 63 (40.64%) self-identified as neurodivergent and 89 as neurotypical, and 3 did not disclose. This corresponds to 108 students (48 ND, 59 NT, 1 not answered) in the fall term 2021 (three courses) and 47 students (15 ND, 30 NT, 2 not answered) in the spring term 2022 (one course).

Intervention Experiences

While our students had the choice which material they used, ND-friendly material was used by over 50% of the respondents, among them many NT students. The detailed breakdown is depicted in Figure 24-1 for the three courses in the fall term 2021 and for the remaining course in the spring term 2022. Interestingly, several NT students

CHAPTER 24 ECONOMICAL ACCOMMODATIONS FOR NEURODIVERGENT STUDENTS IN SOFTWARE ENGINEERING EDUCATION: EXPERIENCES FROM AN INTERVENTION IN FOUR UNDERGRADUATE COURSES commented in free-text answers that they also found the ND-friendly material more readable compared with the traditional style and therefore preferred it.

Many students, both NT and ND, further expressed gratitude that we raised awareness for neurodiversity. Finally, five ND students stated that they felt welcome in our courses. Given that academic literature regularly points out that ND students experience unwelcoming environments in higher education, we consider this one of the most important outcomes of our intervention.

We further asked participants to rate whether they (a) appreciate that material is provided in two styles, (b) find the ND-friendly material harder to read, and (c) find that the ND-friendly material helps them understand the content better. The results are depicted in Table 24-1. Note that the statements in the first column are shorter, simplified versions of the actual statements, which can be found in the material we provide. The answers show that accommodations were received well by most of the participants. Furthermore, a large proportion of all participants found that the ND-friendly material helped them understand the material better. Considering only ND participants, the agreements are naturally higher. That is, 48% of the ND students in the first survey agree that the ND-friendly material helped their understanding (compared with 34% of the NT students), and 54% of the ND students agree in the second survey (compared with only 12% of NT students). This shows that the accommodations reached their goal of primarily supporting ND students. Nevertheless, the divided agreements and disagreements also clearly show that our simple accommodations are no silver bullet.

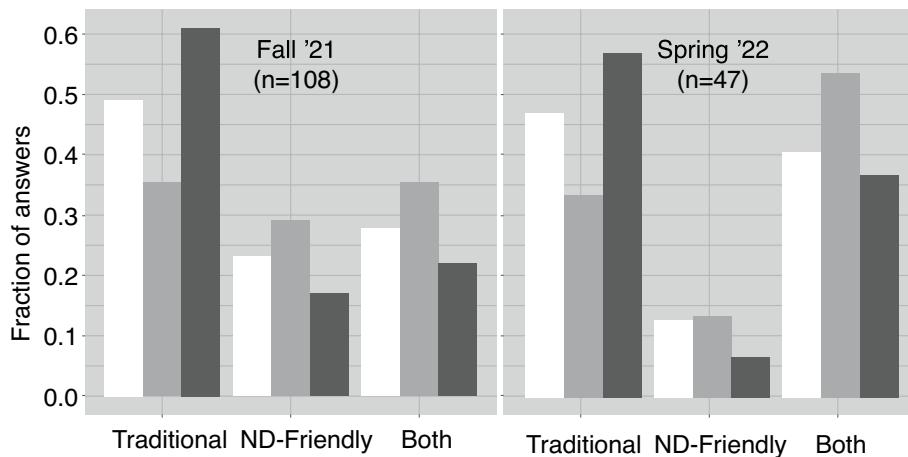


Figure 24-1. Used materials in the fall term 2021 ($n=108$) and spring term 2022. All students are depicted in the white bars, while ND students are shown in gray and NT students in black

Table 24-1. Perceptions by all participants regarding the ND-friendly course material in fall and spring terms

Statement	Fall '21 (n=108)	Spring '22 (n=47)
I appreciate that material is provided in two styles.	1% disagree, 5% neutral, 94% agree	4% disagree, 4% neutral, 91% agree
The ND-friendly material is harder to read.	65% disagree, 13% neutral, 22% agree	52% disagree, 8% neutral, 40% agree
The ND-friendly material helps me understand the content better.	27% disagree, 30% neutral, 42% agree	34% disagree, 31% neutral, 34% agree

In addition to closed questions, we received a lot of noteworthy feedback as free-text answers. Four students found the changed font particularly hard to read, especially in assignments, and therefore preferred the traditional material, even though it might not be ideal either. For example, one NT student commented in free text that they felt “very distracted” by the new font. Given this feedback and the fact that we used a nonstandard

font, we are currently considering to use a standard font that is considered suitable for dyslexic audiences in the future, for example, Arial. Three ND students commented that they did not want accommodations or that they were satisfied with extended exam time already provided by the university. Finally, a few comments showed the individual nature of neurodiversity. One dyslexic student commented on the difficulty of following a course offered in English only, while another student expressed difficulties as the course they attended was given in Icelandic, but the course book was English. That is, they struggled primarily with the multilingual nature of the course. Similarly, one ND student stated that working from home had a positive impact on their productivity, while another student stated during one of our courses that they found it especially hard to focus at home due to their ND condition. Overall, these comments show that simple changes are unlikely to cater to all ND audiences and that flexibility in accommodations is needed. However, it is worth pointing out that we did not receive any negative feedback.

In addition to direct feedback on our accommodations, several students provided suggestions on additional accommodations. Particularly common were suggestions to provide video or audio recordings in which we explain assignments or simply read them out loud. This clearly highlights difficulties many ND students face with text comprehension. However, it also shows the potential benefit of text-to-speech tools that have been improving substantially in the last years. Another common point for improvement is the organization and presentation of a single course. Several ND students pointed out that it can be hard to find all the relevant information on a course. Additionally, syllabi in our programs are currently not standardized, which leads to large differences in how courses present information such as learning outcomes. This poses an additional obstacle for many ND students. We believe these two directions are alternatives to our style- and presentation-based approach that could provide particularly valuable accommodations, that is, different modalities for course material and improved organization.

Summary

In this chapter, we summarized potential accommodations for neurodivergent (ND) students in higher education, focusing on the three most common conditions: autism spectrum disorder (ASD), attention deficit hyperactivity disorder (ADHD), and dyslexia. We outlined which accommodations we made to four courses in computer science

CHAPTER 24 ECONOMICAL ACCOMMODATIONS FOR NEURODIVERGENT STUDENTS IN SOFTWARE ENGINEERING EDUCATION: EXPERIENCES FROM AN INTERVENTION IN FOUR UNDERGRADUATE COURSES

and software engineering undergraduate programs, focusing on changes in style to assignments and slides that are easy to implement with limited time. Our results show that minor accommodations for ND students can have an important impact, from some ND students feeling more seen and welcome in courses to actual perceived improvements in understanding the course material. Researchers and educators can use our provided material to conduct studies and develop their own interventions.

Our accommodations were of general nature, even though we applied them in a computer science and software engineering context. We consider this an important starting point before applying accommodations that target specific characteristics of or tasks in software engineering. Future work in this area should target how to convey course organization in a better way and how to use other media than text for expressing tasks, assignments, and project descriptions, such as video or audio recordings. Furthermore, we believe that more work is needed that studies what tasks in software engineering are particularly difficult or easy for ND individuals and why. To this end, we have started investigating which strengths ND individuals exhibit with respect to software engineering tasks and how they can use these in a better way.

Takeaway Points

1. Neurodiversity is not rare and is increasing in higher education.
 2. Many accommodations are difficult and effort intensive.
 3. Small, economical changes can make a difference.
 4. Awareness is a first, important step.
 5. Feedback clearly shows the individual nature of neurodiversity.
-

Bibliography

- [1] Thomas Armstrong. The myth of the normal brain: Embracing neurodiversity. *AMA Journal of Ethics*, 17(4):348–352, 2015.
- [2] Simon Baron-Cohen, Emma Ashwin, Chris Ashwin, Teresa Tavassoli, and Bhismadev Chakrabarti. Talent in autism: hyper-systemizing, hyper-attention to detail and sensory hypersensitivity. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1522):1377–1383, 2009.

CHAPTER 24 ECONOMICAL ACCOMMODATIONS FOR NEURODIVERGENT STUDENTS IN SOFTWARE ENGINEERING EDUCATION: EXPERIENCES FROM AN INTERVENTION IN FOUR UNDERGRADUATE COURSES

- [3] Andrew Begel, James Dominic, Conner Phillis, Thomas Beeson, and Paige Rodeghero. How a remote video game coding camp improved autistic college students' self-efficacy in communication. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 142– 148, 2021.
- [4] Lynn Clouder, Mehmet Karakus, Alessia Cinotti, María Virginia Ferreyra, Genoveva Amador Fierros, and Patricia Rojo. Neurodiversity in higher education: A narrative synthesis. *Higher Education*, 80(4):757–778, 2020.
- [5] Maitraye Das, John Tang, Kathryn E. Ringland, and Anne Marie Piper. Towards accessible remote work: Understanding work-from-home practices of neurodivergent professionals. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–30, 2021.
- [6] Equality and Participation Unit, UCU. Neurodiversity in the work-place – supporting neurodivergent members at work and campaigning for neurodiversity-friendly workplaces. Technical report, University and College Union, January 2022.
- [7] Nicholas W. Gelbar, Isaac Smith, and Brian Reichow. Systematic review of articles describing experience and supports of individuals with autism enrolled in college and university programs. *Journal of Autism and Developmental Disorders*, 44(10):2593–2601, 2014.
- [8] Edward Griffin and David Pollak. Student experiences of neurodiversity in higher education: insights from the BRAINHE project. *Dyslexia*, 15(1):23–41, 2009.
- [9] Ann Jurecic. Neurodiversity. *College English*, 69(5):421–442, 2007.
- [10] Sally Lindsay, Victoria Osten, Mana Rezai, and Sunny Bui. Disclosure and workplace accommodations for people with autism: A systematic review. *Disability and Rehabilitation*, 43(5):597–610, 2021.
- [11] Ian McChesney and Raymond Bond. Gaze behaviour in computer programmers with dyslexia: considerations regarding code style, layout and crowding. In *Proceedings of the Workshop on Eye Movements in Programming*, pages 1–5, 2018.

CHAPTER 24 ECONOMICAL ACCOMMODATIONS FOR NEURODIVERGENT STUDENTS IN SOFTWARE ENGINEERING EDUCATION: EXPERIENCES FROM AN INTERVENTION IN FOUR UNDERGRADUATE COURSES

- [12] Ian McChesney and Raymond Bond. Eye tracking analysis of computer program comprehension in programmers with dyslexia. *Empirical Software Engineering*, 24(3):1109–1154, 2019.
- [13] Ian McChesney and Raymond Bond. Observations on the linear order of program code reading patterns in programmers with dyslexia. In *Proceedings of the Evaluation and Assessment in Software Engineering*, pages 81–89, 2020.
- [14] Ian McChesney and Raymond Bond. The effect of crowding on the reading of program code for programmers with dyslexia. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 300–310. IEEE, 2021.
- [15] Ian McChesney and Raymond Bond. Eye tracking analysis of code layout, crowding and dyslexia-an open data set. In *ACM Symposium on Eye Tracking Research and Applications*, pages 1–6, 2021.
- [16] Meredith Ringel Morris, Andrew Begel, and Ben Wiedermann. Understanding the challenges faced by neurodiverse software engineering employees: Towards a more inclusive and productive technical workforce. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, pages 173–184, 2015.
- [17] Robin L. Peterson and Bruce F. Pennington. Developmental dyslexia. *The Lancet*, 379(9830):1997–2007, 2012.
- [18] Marco Pino and Luigina Mortari. The inclusion of students with dyslexia in higher education: A systematic review using narrative synthesis. *Dyslexia*, 20(4):346–369, 2014.
- [19] Guilherme Polanczyk, Maurício Silva De Lima, Bernardo Lessa Horta, Joseph Biederman, and Luis Augusto Rohde. The worldwide prevalence of ADHD: a systematic review and metaregression analysis. *American Journal of Psychiatry*, 164(6):942–948, 2007.
- [20] Holly A. White and Priti Shah. Creative style and achievement in adults with attention-deficit/hyperactivity disorder. *Personality and Individual Differences*, 50(5):673–677, 2011.

CHAPTER 24 ECONOMICAL ACCOMMODATIONS FOR NEURODIVERGENT STUDENTS IN SOFTWARE ENGINEERING EDUCATION: EXPERIENCES FROM AN INTERVENTION IN FOUR UNDERGRADUATE COURSES

- [21] Erik G. Willcutt. The prevalence of DSM-IV attention-deficit/hyperactivity disorder: a meta-analytic review. *Neurotherapeutics*, 9(3):490–499, 2012.
- [22] Sarah Wille and Daphne Sajous-Brady. The inclusive and accessible work-place. *Communications of the ACM*, 61(2):24–26, 2018.
- [23] Annuska Zolyomi, Andrew Begel, Jennifer Frances Waldern, John Tang, Michael Barnett, Edward Cutrell, Daniel McDuff, Sean Andrist, and Meredith Ringel Morris. Managing stress: The needs of autistic adults in video calling. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–29, 2019.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 25

Effective Interventions to Promote Diversity in CS Classroom

Lucia Happe, University Karlsruhe, Germany.*

The direction toward a sustainable and more balanced society asks for higher engagement of diverse people in innovation, shedding light on the critically low number of women choosing computer science (CS) education and career. Despite interventions by the research community, governmental, and educational institutions all aimed at increasing gender diversity in CS over the past two decades, the gender gap seems to hold [13].

Although progress in the higher involvement of women in CS is barely visible, we are starting to understand the reasons and myths behind the trend. Harvey Mudd's president, Maria Klawe, has summarized their experience as: "*Number one is they think it's not interesting. Number two, they think they wouldn't be good at it. Number three, they think they will be working with a number of people that they just wouldn't feel comfortable or happy working alongside*" [11]. Although the negative view of CS and the confidence gap, as described by Maria Klawe, play their role in narrowing the number of girls interested in CS, we see that there is still a substantial number of girls and women enthusiasts who would like to pursue CS careers [7] but get discouraged by unnecessary frustrations they experience along the way.

This chapter elaborates on these frustrations, all of which seem to be preventable, and goes further by summarizing the existing knowledge on effective interventions to target and mitigate them in CS education. The intent is to provide educators with a comprehensive and easy-to-navigate map of the interventions, as well as highlight the most promising solutions, such as the interdisciplinary approach, that could be

of tremendous help in engaging girls in CS. Readers will gain better understanding of frustrations of girls in CS and orientation in the recommendations for CS educators committed to sparking and retaining the interest and participation of girls in their classes.

Frustrations Steering Girls Away from CS

What is CS lacking that makes girls self-select away from it and seek other interests?

To answer this question, we had to understand the perceived frustrations that women themselves report as the reasons they dropped out of CS education despite being keen about it in general. To this end, we have designed and used a retrospective questionnaire study, which has revealed numerous interesting insights about frustrations that women interested in computing experience along different phases of their education and career. The admirable ability of participants to retrospect on their previous experiences helped us follow a pattern in the responses and, based on their reports, identify the causes that underlie women's attrition in CS.

The study was realized via a questionnaire (in English), designed to understand how and why women engage with CS, as well as the challenges girls and women face when participating in CS activities at school or at home and factors that enable their entry and ongoing participation in the further education. The questionnaire consisted of a number of questions, including six open questions asking the respondents to retrospect and analyze their previous studies and ambitions. These questions asked about participants' understanding of who computer scientists are, what were the drivers and obstacles on their way to CS, what makes them enthusiastic about it, and what they would recommend improving CS education for girls. The results presented here are based on the responses to these open questions (quoted as written), in combination with basic classification questions about participants' age, gender, and major interests. We distributed the questionnaire among groups with the affinity toward CS, especially educational institutions providing late-education offers (for adult women), such as Czechitas (www.czechitas.cz).

We successfully collected 139 responses from women¹ in three age groups (18% between 18 and 26, 41% between 27 and 34, 33% over 34 years old, and 8% without age indication). This study's population was represented by a near-even distribution of respondents over three regions: the Czech Republic, Germany, and others (constituting one-third of the responses, however, distributed all around the world in small numbers). The questionnaire resonated with the audience, 90% of respondents filled out all the open questions, and many did it very thoughtfully and expressively. The responses were distributed among three groups: (1) women who studied and stayed in computing, 39% of respondents; (2) women who transitioned to computing later in life, after studying another discipline, 32% of respondents; and (3) women who never considered entering computing, 29% of respondents.

We narrowed our attention to the frustrations more strongly reported by women from the second group. We specifically looked for respondents who likely had high potential to study CS in their earlier years (manifested by expressing previous interest, however, decided to study another discipline and returned to CS later in life), which makes our study different from other similar studies, for example, [2, 9, 17]. To find out what made them select away at the first place, we compared their responses with the women who stayed in – focusing on their views of the moments that formed the direction they decided to take (stay in or disengage, although overall interested in it). We were asking them to elaborate retrospectively about the biggest obstacles and drivers on girls' way to CS [15].

Thus, what is CS lacking that makes women seek other interests and study programs and professions as an alternative to CS? Firstly, the participants reported, on average, five other major interests (languages, biology, mathematics, sociology, psychology, business, history, and music are most common) competing for their attention with CS at that time, which made it easy for them to simply skip CS as an option and focus on other interests that took the space in their thoughts and time. A typical quote illustrating this is: *"In retrospect, I'd like there to be someone who noticed that I had my head on computers and kept me there. I had a lot of other interests, guitar twice a week, and volleyball twice a week. I took computers like, 'Yeah, I'd probably like that,' but I had a lot of other things."* In many cases they reported on the early dilemma whether to follow their other interests or CS. Those who were still considering CS as one of the preferred options concluded that choosing CS meant leaving all their other interests behind,

¹ Filtered from 151 responses after removing incomplete responses and responses representing gender groups not targeted by this study.

which they did not want to do. This also indicates what “engaging education” means to them. They want to understand technology as the means to have relatable impact in other areas of their interest as well, not for the sake of technology as the end goal.

Table 25-1. *The most mentioned categories of obstacles on their way to CS*

Reason to Opt Out	Typical Quote About Experienced Obstacles
Missing Access	
...to suitable education	<i>“Lack of good opportunities to learn – in school it was boring, more about using specific software or programming language ...”</i>
...to support and encouragement	<i>“There was none (in a family or at school) to show me the options the computer can be used for (except playing games).”</i>
...to a computer	<i>“I wasn’t allowed to install anything on my computer as my dad believed I would break it.”</i>
Gender-Related Stereotype	
...carried by others	<i>“My oldest brother (computer scientist) telling me I should not go into CS because women are not being taken seriously in that area.”</i>
...about the unrelated purpose of SE	<i>“Boring computer programming in school put me off for decades. Why would I want to write a game I wouldn’t want to play? Waste of time.”</i>
...about themselves	<i>“Biggest obstacle on my way to computing was my rather stereotypical knowledge of it. I assumed you have to be super smart, very emotionless, etc. And I was not, so pursued at a first different path.”</i>
Missing Confidence	
...low self-efficacy	<i>“I didn’t dare to take a CS course in high school because I felt I was lacking the necessary knowledge.”</i>
...experiencing imposter syndrome	<i>“Many girls think that if they try, they will fail, and people will laugh at them.”</i>
...missing success experiences	<i>“Encouragement. That’s what girls need. And community where they don’t feel embarrassed that they are not experts in computing, but they want to learn anyway ... they will be amazing at it in the future, it’s not the obstacle!”</i>

(continued)

Table 25-1. (*continued*)

Reason to Opt Out	Typical Quote About Experienced Obstacles
Missing Sense of Belonging	
...not comfortable to express themselves	<i>"Other people watching and/or judging me."</i>
...sexism and unwanted attention	<i>"Men. Sometimes it's hard to survive in the collective."</i>
...missing relatable peers	<i>"I don't like the people that work with computers. Too narrow-minded ... Guys in computing usually talk with 'different language' that ladies don't understand."</i>
Feeling Not Valued	
...experiencing defensive culture	<i>"It is a man-dominated field ... As a woman, you have to prove them wrong."</i>
...feeling that women are not valued	<i>"The work of female developers is rather not acknowledged."</i>
... <i>feeling that their skills/interests are not valued</i>	<i>"... It is necessary to change herself, to change her field, to get into the 'men's world' and she does not want to do that ..."</i>

Secondly, the three factors hindering girls' entrance in CS brought in the quote by Maria Klawe (see the introduction and [11]), that is, *stereotypes, confidence, and sense of belonging*, were confirmed by the responses of the participants and also by other existing studies, which further add *early access* as a factor [19]. These factors were the first items added to our list of codes in the first cycle of our exploratory analysis. The qualitative responses were coded based on these four factors. Within our code structure, besides the four key codes that we used initially to structure the discussion of the responses, a fifth factor emerged from analyzing the responses, being *feeling valued* as women in computing careers. The identified categories with examples of quotes in the results of the study are outlined in Table 25-1.

Thirdly, when trying to understand what made them select away from CS in the first place, we found the following reasons as most prevalent: the vast majority of 92% of participants in the second group were missing suitable courses to learn about

computing; 58% of them were concerned about the possibility to follow their other multidisciplinary interests despite learning computing; and 18% of them could not relate to computing at all because of other interests that took the space in their thoughts and time. Thus, the funnel into CS education is leaky. When following the major leaks, there is a likelihood those girls will not have access to an engaging educational offer in the area of their interest, a likelihood that they will be convinced that they and their interests do not fit and are not connected to CS, a likelihood of that they falsely believe that because of having other interests and not investing all their time into computing they cannot be as successful as others in CS, or a likelihood that they experience their non-stereotypical skills and interests being considered second-class and the advantage of having them not being understood and appreciated in CS.

Effective Interventions to Engage Girls in CS

To respond to reported reasons girls dropped out of CS education, we looked for the existing knowledge on effective interventions to target and mitigate experienced frustrations in CS education. The effective interventions to recruit and retain girls in CS education include interventions that combat wrong gender-related stereotypes, interventions that spark initial interest by providing suitable access through actively engaging interdisciplinary strengths for relatable purposes, interventions managing suitable first contact through relatable activities compensating for missing access, interventions for creating a less hostile environment through building a sense of belonging, interventions increasing self-confidence, and interventions for sustaining long-term commitment, as we have summarized in [16] accumulating joined knowledge from over 800 publications via a cumulative review of literature on the topic. With that position, we have also recognized that education design can actively use numerous techniques (see Table 25-2) to attract and retain girls engaged during the education process. The aim of this chapter is to contribute to the area of CS education by summarizing proven teaching interventions that support diverse and inclusive CS.

Our results show that one of the most powerful elements, resulting in student participation and retention, is interest [1, 8, 18]. Interest energizes the learning process, guides the learning trajectory, and is crucial for the success of the overall endeavor. We hence organized the identified recommendations according to the phases in which

interest is cultivated and evolves into confidence and commitment. Table 25-2 maps this conceptual model of interest emergence showing how interventions promote its development and subsequent goals in five chronological phases.

Targeting gender-related stereotypes: The research by [14] shows that teachers and classrooms that do not make explicit efforts to provide a women-friendly environment for exploring CS will naturally end up promoting CS as a male-oriented domain. It is being acknowledged as a result of the differences in leisure-time preferences among girls vs. boys [19]. Girls usually start using a computer much later, for homework, research, and socializing, while at that time, boys already tend to have a few more years of experience with computers [19], which makes it hard to reverse differences in computing literacy. In effect of that, boys, who are on average one year ahead of girls in computer usage, according to the findings, tend to monopolize the instructor's time, computer labs, and the curriculum material [5, 26, 28]. This situation leads to even fewer opportunities to gain experience and increase their confidence with computers, which can be observed among girls as well as among less experienced boys. When students, whether girls or boys, fall into this vicious circle [27], it is not very likely they escape it without the explicit initiative of the school and parents who would create an engaging environment effectively pulling these kids back into CS education. As girls fall into this vicious circle more often than boys, it is natural to ask what the girl-friendly CS education environment looks like and what requirements it poses on the curriculum, culture, and overall environment. Recs. 1.1.–1.4. list interventions that could be applied by teachers to prepare positive conditions for interest to emerge and combat the negative gender-related stereotypes in their classroom.

Table 25-2. Effective interventions to promote diversity in CS

Interest Phases	No.	Intervention Recommendation (Rec.)	Goal of the Intervention
Preparing conditions for emergence of interest	1	Targeting the frustration of gender-related stereotypes	
	1.1.	Provide non-stereotypical role models [4].	Revising stereotypes, showing that they have other interests and life outside CS
	1.2.	Portray work in computing as helpful, altruistic, and community-oriented [4].	Revising what computer scientists do and the real impact they have on our world
	1.3.	Provide opportunities to do computing activities as part of a group [4, 5, 24].	Revising how and with whom computer scientists work and what are the other tasks they do except coding (e.g., brainstorming, product design, modeling)
	1.4.	Expose girls to successful and relatable women role models in computing [4, 24].	Inoculating girls against bad stereotypes (“psychological vaccines”) and counterarguments, following good examples
Triggering initial situational interest	2	Targeting the frustration of missing access	
	2.1.	Emphasize the social impact and interdisciplinary nature of computing work [5, 10, 12, 19, 23, 26].	Increasing sense of belonging, perceived similarity, relevance, and opportunity to succeed by utilizing non-stereotypical skills and strengths
	2.2.	Include physical reminders of women’s success in computing [4].	Providing role models and how they had impact and used their other interdisciplinary strengths

Maintaining situational interest	3	Targeting the frustration of missing access
3.1.	Digital gaming and creative arts activities designed for girls [19, 21].	Providing an entry point to computing for girls illustrating the level of creativity in CS relatable to girls
3.2.	Start with offline contextualized activities [12, 20].	Decreasing barrier to start and practice, achieving some intellectual satisfaction before teaching to be resilient in the face of frustrations involved in coding
3.3.	Visual programming environments to teach introductory programming [20, 26].	Providing engaging, less rigorous introduction
3.4.	Involve discussions and reflections on the activities [12].	Promoting learning, emphasizing the bigger context and impact of CS work

(continued)

Table 25-2. (continued)

Interest Phases	No.	Intervention Recommendation (Rec.)	Goal of the Intervention
Developing individual interest	4	Targeting the frustration of missing sense of belonging	
	4.1.	All-women education programs and classes [12, 14].	Allowing girls to get more instructor attention
	4.2.	Splitting classes by experience, not gender [26].	Working against monopolization of instructor time by the most experienced students
	4.3.	Noncompetitive environment [4].	Minimizing building of classroom hierarchy based on coding competence usually hostile to girls
	4.4.	Collaborative assignments [4, 5, 24].	Broadening the spectrum of skills and experiences gained in the classroom
	5	Targeting the frustration of missing confidence	
	5.1.	Provide low-stakes opportunities for girls to experience success [4, 24].	Gaining confidence through early contribution
	5.2.	Install a growth mindset, the belief that abilities can be improved with effort, strategies, and mentoring [4].	Inoculating girls against bad stereotypes and hostile culture (“psychological vaccines”)
	5.3.	Adopt a positive, constructive attitude toward failure as a valuable learning opportunity [4].	Inoculating girls against hostile culture (“psychological vaccines”), staging a failure example as a learning opportunity

Sustaining long-term interest		Targeting the frustration of feeling valued
6	6.1.	Education of teachers providing them with tools and approaches to better engage girls [5, 10, 24]
	6.2.	Interesting and engaging real-world inquiry-based open-ended hands-on experiences [5, 19, 24, 26]
	6.3.	Emphasizing the social impact and interdisciplinary nature of computing work [5, 10, 12, 19, 23, 26].
	6.4.	Providing long-term self-directed projects [5]
	6.5.	Community and being part of a group [4, 5, 24]
		Increase the ability of teachers to prevent girls' disengagement.
		The vision of having impact, intellectual inquiry, and achievement motivate girls more as a coding challenge.
		Increase sense of belonging, perceived similarity, relevance, and opportunity to succeed.
		Gain experience with self-efficacy.
		Increase the sense of belonging, peer feedback, etc.

Targeting missing access: The preference for other subjects may be attributed to the fact that what makes an appealing first contact with computers for boys is not that appealing for girls. Girls hence opt out from this first attraction and start exploring CS only later [19]; however, at that time they often feel behind. Little research exists on what could be an appealing first contact with computers for girls. A successful example, which is, however, connected to a slightly later age, is interdisciplinary explanatory activities [12, 26]. Girls also tend to be good at typical school achievements, for example, using a computer for homework and writing assignments [19, 26]. Hence, they tend to be attracted to computing once they understand it as a tool impacting their potential academic achievements in all disciplines as a way to test hypotheses and a resource of new knowledge [26]. Attempts should be made to help girls understand these benefits of CS early, showing them CS as a facilitator toward their goals and activities that are naturally appealing to them [4, 19, 21, 26]. The mentioned interventions Recs. 2.1–2.2. include providing girls with exciting and engaging hands-on experiences, increasing motivation to pursue computing through emphasizing the interdisciplinary nature and the social impact of computing work [5, 10, 12, 19, 23, 24, 26], introducing girls to positive role models [4], offering relatable access to CS, and providing information to teachers to encourage interest in computing.

The most common approach to teaching CS is to gradually engage in programming through a process of solving tasks, from very simple to more complex ones [12]. For many girls, this process might make it very difficult to achieve real intellectual satisfaction, which may be a significant obstacle in retaining girls in computing, as this way of thinking degrades digital literacy to pure coding literacy [12]. CS is, however, not only about coding. It requires fundamental skills, such as creativity, imagination, innovation, solution design and problem-solving, understanding of human behavior and needs, experience design, and a combination of mathematical and engineering thinking, to use concepts of computer science effectively [12, 19]. An endeavor to present computer science as a tool to realize and scale any idea, originating from and innovating any discipline possible and available, to anyone independent of previous coding expertise seems to be the key to bringing girls to understand the opportunities in computing and space for their creativity. There is no silver bullet to solve this problem, and it will not merely fix itself with time. A well-designed package of interventions (Recs. 2.1. –2.2. and 3.1. –3.4.) is needed in each classroom for all the students who otherwise miss the opportunities provided by computing in any discipline.

Targeting missing sense of belonging: Works exist that examine the benefits of students' separation into groups within the classroom, allowing educators to tailor CS education to meet students' needs best. Research suggests that the separation of classes based on gender (Rec. 4.1.) [3, 12, 14] has a similar effect as separation based on previous experience (Rec. 4.2.) [26]. This needs further investigation. All-girls classrooms are shown to be beneficial for adolescent girls (as well as adult women) [6, 7] to create a safe environment when entering CS education and building the initial confidence, making it easier to experiment and express their creativity freely. Teachers need to implement preventive measures to ensure that no group of students can monopolize lessons based solemnly on their preferences [3]. Girls are likely to have a purposeful and value-based approach to technology, which distances them from the stereotype they perceive as necessary when seeking success in computing, of being fascinated and passionate about the technology itself. Many girls might find it easier to find their way to technology in homogeneous girl groups, while in mixed groups, their self-image might be a hindrance [12]. On the other hand, this measure might not be easy to implement. To overcome the practicality burden, segregation by experience could serve the purpose in classes where segregation by gender is not practically feasible.

Existing research suggests that the learning environment and the signals girls receive in the classroom play a critical role in determining their interest in computing [4]. The strategies to make the environment less hostile for girls are of enormous impact. According to the examined literature, educators need to work to diminish the usual informal hierarchy and defensive climate based on computing skills, which may take place in CS education [19], and need to install a growth mindset (Rec. 5.2.) [4], where everyone can genuinely believe that they can improve, having a positive and constructive attitude toward failure (Rec. 5.3.) [4], with the failure being seen as an opportunity to improve.

Targeting missing confidence: Girls' leisure-time preferences show not only that girls are statistically better in collaborative and social tasks but they prefer to participate in such activities more [24]. This opens a way to provide low-stakes opportunities (Rec. 5.1.) [4, 24] for girls to succeed in solving classroom tasks and see themselves as contributors to the solution. This could be achieved, for example, by creating more opportunities for discussion and presentation of solution design, making these skills (which are also seen as crucial for success in a computing career) an integral part of the feeling of success in computing [4]. It seems that if the competencies in which girls tend to excel become part of the CS education design, we can stimulate and facilitate

the identification of girls with computing, even for girls who might be stronger in other skills than those we stereotypically link to computing. This is actually in alignment with the situation in the computing industry, where a diverse cohort of individuals is needed to build various teams, involved in product design, implementation, testing, and management, making the soft skills of similar importance to the hard technical skills for success in computing (Rec. 3.4.) [12].

Targeting frustration of not feeling valued: In general, the contribution of girls in computer science is not as widely acknowledged as it could be. Despite the fact that women have made significant contributions to the field throughout its history, they are often underrepresented in the field, and their contributions are often not given the same level of recognition or visibility as those of men. This is partly due to the fact that computer science is a field that has traditionally been male-dominated, and as a result, the perspectives and contributions of women may not be as well-represented in the literature, in the classroom, or in the industry. Even if the contributions of women are presented and recognized often, they map the expectations for recognition defined by the male majority. In fact, differences in leisure-time activities at a very young age influence stance toward CS later in life and result in different experiences of impact, understanding of the valued contribution, and expectations for recognition. Girls seem to need more appealing purposeful contact with a computer [19, 21]; they tend to be more interested in and recognize, as achievements from others as well from themselves, computing contribution that solves social and humanitarian problems and are more motivated when they can see the impact of their work. This mismatched understanding of success may attribute to the fact that what makes an appealing challenge for boys is not that appealing for girls; girls hence opt out from this attraction and look for having their contribution somewhere else [19], where they feel that their contributions to the discussion are valued by the community or their understanding of having impact fits.

The study in [26] analyzed successful extracurricular activities and concluded that the provision of pedagogically effective extracurricular activities in the long term requires unsustainable high effort. It usually either results in ending the activity too early or in limiting the effort to a one-shot intervention with no long-term effect. The study pointed out that the integration of such activities in a regular CS education classroom is a goal without which a change can hardly be achieved. However, to reach this goal, professional teacher development programs need to be introduced that equip the teachers with the knowledge, easy-to-use tools, and guidelines

(Recs. 6.1.-6.5.) and provide information to teachers to encourage interest in computing (Recs. 6.2.-6.3.). The teachers need this guidance to build up the necessary confidence and, hence, be able to transfer their confidence to motivate students into a career in computer science [5, 10, 24, 26].

Several programs, initiatives, and interventions have been launched to address women's underrepresentation in computer science. At the high-school level, several studies observe differences in computer science course behavior between male and female adolescents [4, 12, 19, 24]. Girls are observed to be less likely to choose computer science courses due to classroom environments perceived by them as hostile to their way of self-expression and achievement understanding. The related stereotypical images and messages, as well as individual interactions in the classroom, reduce girls' sense of belonging. Effective intervention strategies often include providing girls with engaging hands-on experiences, increasing motivation to pursue computing through emphasizing the social impact of computing work, providing girls opportunities to succeed with their strengths, introducing girls to positive but relatable role models, and providing training to teachers and key individuals to encourage greater interest in computing and technology. If given the necessary support through professional development programs, educators could react and adapt. For the moment, CS education might be the essential factor that can be changed to influence the recruitment and retention of girls in computing. Yet, sufficient pedagogical guidance is essential to make this endeavor successful.

Conclusion

Overall, there is a silver lining connected to these pipeline leaks – multidisciplinarity. The women in the study showed to have many other interests, on average 5.5 other major interests besides computing. Thus, the possible time slot where computing could be practiced was immediately filled with another without further conscious notice by the girls. There is thus a potential to create alternative pathways [25] into the field by merely building on individual interests. As many women find it hard to identify themselves with computing as such (also indicated by the confidence gap and missing sense of belonging), we might want to leverage their personal interests to create identities that do resonate. We suggest that a different learning approach, that is, *interdisciplinary approach* [22], could have a particularly strong potential for strengthening women's engagement in computing. These different interdisciplinary subcultures can provide

an environment where all the students who currently feel left behind can learn CS without feeling trapped by the dominant culture associated with the field nowadays. This would further expand different entryways in computing and help students be more comfortable exploring and experimenting with computing, have the stability of a familiar knowledge base, and have the ability to self-identify with relevant problems. While mixing the “unfamiliar with the familiar,” they might be more intrigued when unexpected things happen and feel more competent because of the possibility of explaining the new findings using their strengths in a familiar context. Interdisciplinary approaches could further enrich formal education by integrating other sciences and humanities, promoting versatility for future workplaces and real innovation, which can hardly be achieved without computing crossing its own boundaries. Thus, we should stop compartmentalizing learning by discipline and inquire with the help of computing across boundaries. Real innovation also requires the divergent skills that are the focus of arts and humanities. Without these broader skills, it is very difficult to apply any solutions to problems in the real world.

Our comprehensive overview of interventions points to some exciting directions in the field of CS education. Computer science encompasses many perspectives: it is creative and social and provides for well-paid and often interdisciplinary careers. Future CS education research needs to demonstrate the usefulness of CS in contributing to solving critical societal, natural, and economic challenges. There are many interesting opportunities for women to engage in this field.

Bibliography

- [1] Mary Ainley. Being and feeling interested: Transient state, mood, and disposition. In *Emotion in Education*, pages 147–163. Elsevier, 2007.
- [2] Deborah J. Armstrong, Cynthia K. Riemenschneider, and Laurie G. Giddens. The advancement and persistence of women in the information technology profession: An extension of Ahuja’s gendered theory of it career stages. *Information Systems Journal*, 28(6):1082–1124, 2018.
- [3] Lecia J. Barker and William Aspray. The state of research on girls and IT. In *Women and Information Technology: Research on Underrepresentation*. The MIT Press, 2006.

- [4] Jilana S. Boston and Andrei Cimpian. How do we encourage gifted girls to pursue and succeed in science and engineering? *Gifted Child Today*, 41(4):196–207, 2018.
- [5] Jennie S. Brotman and Felicia M. Moore. Girls and science: A review of four themes in the science education literature. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, 45(9):971–1002, 2008.
- [6] Barbora Buhnova, Lucie Jurystova, and Dita Prikrylova. Assisting women in career change towards software engineering: experience from Czechitas NGO. In *Proceedings of the 13th European Conference on Software Architecture – Volume 2*, pages 88–93, 2019.
- [7] Barbora Buhnova and Dita Prikrylova. Women want to learn tech: Lessons from the Czechitas education project. In *2nd International Workshop on Gender Equality in Software Engineering (GE)*, pages 25–28. IEEE, 2019.
- [8] A. Bull, J. Gilbert, H. Barwick, R. Hipkins, and R. Baker. Inspired by science. *A paper commissioned by the Royal Society of New Zealand and the Prime Minister's Chief Science Advisor. Accessed on the*, 20(03):2013, 2010.
- [9] Kaylene Clayton, Jenine Beekhuyzen, and Sue Nielsen. Now I know what ICT can do for me! *Information Systems Journal*, 22(5):375–390, 2012.
- [10] Tom Crick. Final draft: Computing education: An overview of research in the field. 2017.
- [11] Mark Fidelman. Here's the real reason there are not more women in technology. 2012. Available online at URL www.forbes.com/sites/markfidelman/2012/06/05/heres-the-real-reason-there-are-not-more-women-in-technology/.
- [12] FJ García-Peña, D Reimann, M Tuul, AM Rees, and I Jormanainen. TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. *Belgium*, doi: <https://doi.org/10.5281/zenodo.165123>, 2016.
- [13] Elena Gorbacheva, Jenine Beekhuyzen, Jan vom Brocke, and Jörg Becker. Directions for research on gender imbalance in the IT profession. *European Journal of Information Systems*, 28(1):43–67, 2019.

- [14] Denise Gürer and Tracy Camp. An ACM-W literature review on women in computing. *SIGCSE Bull.*, 34(2):121–127, June 2002.
- [15] Lucia Happe and Barbora Buhnova. Frustrations steering women away from software engineering. *IEEE Software*, 39(4):63–69, 2022.
- [16] Lucia Happe, Barbora Buhnova, Anne Koziolek, and Ingo Wagner. Effective measures to foster girls' interest in secondary computer science education. *Education and Information Technologies*, pages 1–19, 2020.
- [17] KD Joshi, Eileen Trauth, Lynette Kvasny, and Sterling McPherson. Exploring the differences among it majors and non-majors: Modeling the effects of gender role congruity, individual identity, and IT self-efficacy on IT career choices. In *Proceedings of Frontiers in Education: Computer Science and Computer Engineering*, 2013.
- [18] Andreas Krapp and Manfred Prenzel. Research on interest in science: Theories, methods, and findings. *International Journal of Science Education*, 33(1):27–50, 2011.
- [19] Joyce B. Main and Corey Schimpf. The underrepresentation of women in computing fields: A synthesis of literature using a life course perspective. *IEEE Transactions on Education*, 60(4):296– 304, 2017.
- [20] Divya Menon, Sowmya Bp, Margarida Romero, and Thierry Viéville. Going beyond digital literacy to develop computational thinking in k-12 education. *Smart Pedagogy In Digital Learning*, 2019.
- [21] Jennifer Milam. Girls and stem education: A literature review. *Atlanta: Georgia Institute of Technology*, 2012.
- [22] Victoria Millar. Trends, issues and possibilities for an interdisciplinary stem curriculum. *Science & Education*, 29(4):929–948, 2020.
- [23] Alex Murphy, Ben Kelly, Kai Bergmann, Kyrylo Khaletskyy, Rory V O'Connor, and Paul M. Clarke. Examining unequal gender distribution in software engineering. In *European Conference on Software Process Improvement*, pages 659–671. Springer, 2019.
- [24] Jennifer Nash. *Understanding How to Interest Girls in STEM Education: A Look at How Lego® Education Ambassador Teachers Engage Female Students in STEM Learning*. PhD thesis, University of Florida, 2017.

- [25] Elizabeth Ann Patitsas. *Explaining Gendered Participation in Computer Science Education*. PhD thesis, 2019.
- [26] Leo A. Siiman, Margus Pedaste, Eno Tõnisson, Raivo Sell, Tomi Jaakkola, and Dimitris Alimisis. A review of interventions to recruit and retain ICT students. *International Journal of Modern Education and Computer Science*, 6(3):45, 2014.
- [27] Jeanna R. Wieselmann, Emily A. Dare, Elizabeth A. Ring-Whalen, and Gillian H. Roehrig. "I just do what the boys tell me": Exploring small group student interactions in an integrated stem unit. *Journal of Research in Science Teaching*, 57(1):112–144, 2020.
- [28] Teena Willoughby. A short-term longitudinal study of internet and computer game use by adolescent boys and girls: Prevalence, frequency of use, and psychosocial predictors. *Developmental Psychology*, 44(1):195, 2008.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 26

Software Engineering Through Community-Engaged Learning and an Inclusive Network

Nowshin Nawar Arony, University of Victoria, Canada.*

Kezia Devathasan, University of Victoria, Canada.

Ze Shi Li, University of Victoria, Canada.

Daniela Damian, University of Victoria, Canada.

Retaining diverse, underrepresented students in computer science and software engineering programs is a significant concern for universities. In this chapter, we describe the INSPIRE: STEM for Social Impact¹ program at the University of Victoria, Canada, which leverages the three principles of self-determination theory – competence, relatedness, and autonomy – in the design of strategies to empower women and other underrepresented groups in using software and other engineering solutions to approach sustainability, community-driven problems. We also describe lessons learned from a first successful year that involved over 30 students, 6 community partners (sustainability problem owners), and over 20 industry and academic mentors and reached out to more

¹ <https://inspireuvic.org/>

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

than 200 solution end users in our communities. Finally, we provide recommendations for universities and organizations who may want to adopt our approach.

In the program 24 diverse students (in terms of gender, sexual orientation, ethnicity, academic standing, and background) divided into six teams paired with six community partners worked on solving society-impactful problems and developed solutions for a number of respective community partners. Each team was supported by an experienced upper-year student and mentors from industry and community throughout the program. The experiential learning approach of the program allowed the students to learn a variety of soft and technical skills while developing a solution that has a social and/or environmental impact. Having a diverse team and creating a solution for real end users motivated the students to actively collaborate with their peers, community partners, and mentors resulting in the development of an inclusive network. A network of like-minded people is crucial in empowering underrepresented individuals and inspiring them to remain in the computer science and software engineering fields.

Introduction

Computer science and software engineering university programs have long suffered from a lack of diversity where recruitment and retention of students from underrepresented groups are challenging [3]. Working on society-impactful projects has the potential to motivate women and other underrepresented individuals to continue in Science, Technology, Engineering, and Mathematics (STEM) as they are often drawn toward care-oriented and humanistic careers [4, 6]. Experiential learning, a method employing learning through working [8], has proven to increase confidence and motivation in students to continue and persist to graduation [9].

In recent years, some universities have begun launching initiatives and protocols regarding increasing equity, diversity, and inclusion (EDI) in engineering programs to train faculty, staff, and even teaching assistants [16]. These protocols are a means to help reduce the potential harm that could inflict on a student. In Part 4, Chapter 15, “Beyond Classroom: Making a Difference in Diversity in Tech,” the Czechitas initiative has been described that supports and trains girls in computing education to succeed in tech careers. The community developed through this initiative resulted in the increase of women participants in their following year. Having a supportive network to grow develops a sense of belonging in underrepresented individuals. Furthermore, in Part 5, Chapter 25, “Effective Interventions to Promote Diversity in CS Classroom,” the authors describe how “interest” enhances the learning process and engraves motivation in girls.

Engaging students in projects with social and/or environmental causes and allowing them to interact with real clients can increase interest and confidence of students. Hence, we launched INSPIRE that aims to foster EDI through community-engaged experiential learning for underrepresented students and support them through a network of like-minded individuals. We incorporate design thinking methodology [10] to facilitate students' learning experiences as they collaborate with local nonprofit and for-profit organizations, which we refer to as "community partners."

In this chapter we share the student experiences working in this program. The preliminary goals of the program included (1) *developing an inclusive network of individuals*, (2) *motivating students through empowerment*, (3) *providing experience with realistic and impactful problems through community engagement*, and (4) *learning to work in diverse teams*. Through analyzing program participants' experiences, we identified important lessons and present recommendations that may be beneficial for future practitioners and educators implementing diversity-centered experiential learning opportunities in other institutions.

Program Overview

The INSPIRE program offered a four-month-long summer internship position to science and engineering students at the university to work in teams to solve society-impactful problems for real clients. The program was open to undergraduate and graduate students from varying science and engineering degrees ranging from first year to final year. The students were required to work 40 hours/week and received co-operative education (co-op) credits for their work in the program.

The fundamental idea of the program originated from the principles of self-determination theory [14], which describes the factors that contribute to different types of motivation rooted in three basic needs: *competence*, a need to be able to effectively handle the environment; *relatedness*, a need to have close bonds with others; and *autonomy*, the freedom to make one's own choices. Intensive training was provided for all aspects of the students' projects to provide learners a sense of competence. An emphasis was placed on team bonding activities (e.g., ice-breaking discussions, giving constructive feedback to each other, playing games in groups) and group activities (e.g., hiking, beach walk) with all teams together to create a sense of relatedness in the program network. Finally, while students received extensive training, the details of their projects were not micromanaged by the program management team, providing a sense of autonomy within an otherwise structured program.

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

The program secured sponsors from across Canada that included IBM Canada Advanced Studies, Riipen, Redbrick, Checkfront, McElhanney, PBX Engineering, WSP, iWIST (Island Women in Science and Technology), Viatec, Actua, Inter-cultural Association of Greater Victoria, KWL Consulting Engineers, Axolotl Biosciences, Animikii, and Ocean Networks Canada. Many of the industry partners further offered mentorship to the students; as such, we had over 20 industry mentors supporting the students throughout the four months in various project-related aspects. Some of the mentors were located in countries other than Canada like China, Pakistan, Brazil, and the United States.

Along with that, we had six local community partners: Swan Lake Christmas Hill Nature Sanctuary, Greater Victoria Coalition to End Homelessness, Victoria Brain Injury Society, NatuR&D, local schools in the Greater Victoria area (Claremont Secondary, GNS, and Ecole Victor-Brodeur), and Redbrick. The community partners were local to Victoria as the students needed to work with the clients in person. Many of the community partners were nonprofit organizations who actively engaged in helping those in the community that are most vulnerable.

Finally, the program timeline consisted of eight phases encompassing four months. Table 26-1 shows the different phases during the four months and the tasks involved.

Recruitment and project selection: To maximize the reach of potential participants, an open call was sent through the university platform (i.e., email, social media, etc.) four months prior to the launch of the program. Since the program was based on experiential learning and empowering students, the recruitment did not include any previous course experiences for the student teams. However, a separate call for upper-year experienced students, preferably fourth- or fifth-year undergraduate or graduate students, was made in parallel who would be supporting the students with administrating logistical or teamwork issues. Over 50 students responded who were interviewed through a two-step filtering process that was designed to test different soft skills such as primary communication skills, conflict resolution, and leadership.

The first step consisted of a team activity to test each student's ability to (1) work in a team, (2) overcome any conflict that would occur in that short time span, and (3) self-organize the team decisions. Students were broken into teams of five to six students and tasked with completing a project that addressed a hypothetical problem and constraints, which would require students to demonstrate their skills in these three areas. The second phase of the recruitment was an interview process where each student was interviewed individually on different situational questions, for example, how would they

behave under a conflicting situation, how would they deal with failure, or what would their approach be when conversing with the community partner. At the end of this recruitment process, 30 students were recruited in the program.

Table 26-1. *Timeline of the program*

Month	Phases	Tasks Involved
May	Problem Definition	Students were provided with interpersonal skills training like equity, diversity, and inclusion (EDI), professional conduct and communication with clients, and conflict resolution, project-specific training, design thinking, as well as technical skills training like web frameworks, version control, and Agile. Students were required to finish defining the initial problem.
	Problem Planning and Framing	Students met with their respective community partners and end users to further plan with them and start framing the problem based on collected user data (interviews, focus groups, surveys).
June	Problem Validation and Early Prototyping	Students revisited the clients to validate their findings in order to start prototyping. They further started developing an initial prototype.
	Midterm Presentation	Students showcased their work to the public and received feedback from professionals regarding their project.
July	Validation of Prototype	The students reiterated their prototype with the clients before finalizing it for implementation.
	Solution Development	Students started the solution development and were required to learn the necessary technology.
August	Continued Solution Development	Fully dedicated to the development of the solution, students worked in collaboration with the clients to make sure the clients were satisfied with the product.
	Finalizing Solution Development and Documentation	Students began wrapping up the project and creating proper documentation for the clients so that the clients could take this forward and utilize it in their community.

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

Similarly, an open call for the community to propose projects was made through the community-engaged learning (CEL) department at the university. We selected a total of six projects as they were the most pertinent for the students and program. Not only did each project address a pressing social or environmental challenge affecting the broader community but each community partner also committed to mentoring the students in project-specific training such as dealing with brain injury patients and norms of engaging with vulnerable clients.

Team formation: Out of the 30 students, 24 students were placed in a group of six teams, and each team was assigned a senior experienced student to support them. Prior research showed that working in a project that resonates with one's value gives a sense of motivation to work on the project [13]. Thus, each student was asked to provide their top three project preferences as well as their academic skills, experiences, and project requirements, and then each student was placed in a team accordingly. Each team ended up with diverse members in terms of gender, sexual orientation, ethnicity, academic standing, and background. Moreover, each project had a diverse set of end users, making the projects more challenging and motivating.

Demographic diversity: Among the 24 students, 10 identified as female, 12 identified as male, and 2 preferred not to disclose their gender. Furthermore, 19 students came from the undergraduate level and 5 from the graduate level. The students were further diverse in terms of (1) academic background as they came from computer science, software engineering, electrical engineering, mechanical engineering, biomedical engineering, physics, chemistry, and business and (2) ethnicity, including South Asian, East Asian, Black, Arab, Hispanic, Indigenous, and White. Out of the six experienced students, there were five female and one male student, and each of them had different engineering and science backgrounds. The teams were further matched with industry mentors who guided them on different social and technical issues. Therefore, while each project team had a diverse blend of experiences, skills, and perspectives, all team members had an equal opportunity to work on a project that they were deeply motivated to work on.

Projects

The four-month program was both an accomplishing and turbulent journey for all the students, as the teams had to overcome various adversities. The first month was heavily dedicated to training and learning about the project-specific requirements. Since all the

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

community problems appeared to require technological solutions, the students had to learn different technical skills including programming languages, frameworks, version control, databases, API integration, PCB design, geographic information systems, and many more depending on their project.

Table 26-2. Summary of the projects and the solutions

Project Name	Community Problem	Solution
Swan Lake	Due to the increase in the number of people visiting a local nature	An IOT monitoring system to track and visualize where visitors are
Christmas Hill Nature Sanctuary	sanctuary, preserving and maintaining specific areas is becoming difficult.	trekking in the park
Greater Victoria Coalition to End Homelessness	Women+ fleeing violence and facing homelessness encounter difficulty finding safe and appropriate support, services, and housing.	A website that allows support workers to easily find up-to-date available emergency housing and services for women+ fleeing violence
Victoria Brain Injury Society	Nonprofits supporting brain injury survivors lack a centralized, accessible hub to provide patients with relevant services.	A mobile application with a custom interface directing brain injury survivors toward necessary and appropriate services and support
Carbon footprint awareness for teens	Youth lack the motivation to take climate action due to inadequate knowledge and inspiration.	A gamified classroom app that is designed to help teenagers learn and take action about their personal carbon emissions
The Resilient Urban Systems & Habitat Initiative	Existing climate change data is disorganized and fails to provide informed guidance on potential climate health.	An interactive website that centralizes and reports information about regional climate change vulnerabilities
Carbon impact of web browsing	Digital activities are part of everyday life, but people are unaware of the carbon impacts of browsing the Internet.	A web application that accurately calculates the carbon impact of web browsing

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

The teams were introduced to IBM's design thinking [10] and Agile [1]; as such, they extensively utilized these processes in their project management, software development, and requirement elicitation. Due to the heavy emphasis on experiential learning, the students mostly developed skills through implementing these skills during the software development process. In addition, the students learned a plethora of soft skills, some of which were unique to their project due to having specific clientele. For example, a team of students working with patients who had acquired brain injury required training on methods of interaction with such patients. Gathering requirements from vulnerable groups is sometimes difficult; hence, learning such skills was a significant part of their project. All six projects had pressing community problems, which allowed the students to explore the phases of design thinking [10], learn, and develop solutions that could cater toward the respective end users. Table 26-2 summarizes a brief description of the community problems and the final solutions developed by the teams.

All six projects had unique experiences in terms of overcoming different kinds of obstacles and pivots. However, by the end of the four months, they were left with the sense of achievement and learned different aspects of working in a software development team. Comparing the program with a course, one of the students said: *"This is so much better than taking a course. I think because in a course, the projects feels contrived. And I don't feel like the end result actually does anything. I mean, you learn through it. But it's, that's the intention is learning. Where here learning is not the only intention. It's about, you know, building community and making, you know, building interpersonal skills and really setting ourselves up for the future while also making a product that actually will go out into the world and do some good."* This quote is a perfect summary of the program's goal of creating a network of like-minded people and contributing to their success through motivation, empowerment, mentorship, and curating a safe space in the community.

Research Methods

Over a period of four months, we collected reflections and conducted focus group interviews with students and community partners to improve our program for the next iteration. Workshops were provided to help students reflect on their learning outcomes from working on real-world problems in diverse teams. As part of the deliverables, students were required to write weekly individual and team reflections based on their project development and teamwork experience. Two focus group interviews were

conducted with each team, and community partners were also interviewed to gather their perspective. We collected approximately 300 individual reflections, 90 team reflections, 12 focus groups, and 5 community partner interviews. The collected data was analyzed through Braun and Clarke's [5] six-step thematic analysis method that includes (1) familiarizing with the data, (2) generating initial codes, (3) searching for themes, (4) reviewing potential themes, (5) defining and naming themes, and (6) producing the report. An external member outside the research team reviewed the findings to avoid assumptions and biases. The research team used an iterative discussion cycle and peer debriefing process to extract key themes from the collected data, which we present in the form of the lessons learned in the following section.

Lessons Learned and Recommendation

This section highlights the lessons learned from the four-month-long projects. Working on a community project with *real clients* and *community problems* poses both rewarding and challenging experiences that students would not otherwise be exposed to until post-graduation employment. The students in our program were provided with a unique combination of working with real clients tackling real problems in diverse teams. During this time, we analyzed collected personal and team reflections as well as conducted focus groups with the students to understand their experience. We describe in the following the lessons based on this analysis, which after much reflection, we propose, can be supported pedagogically.

Lesson 1 – Training on soft skills should be emphasized in building a successful network of diverse individuals in software engineering: In a successful network, people are able to socialize and support each other outside of work and develop meaningful relationships. Previous research suggests that such nonprofessional relationships, even in organizational networks, are critical to companies employing research and development projects [14]. Soft skills are immensely important in building such a community, yet university students often lack the soft skills that are required to operate in the real workplace [2]. Thus, training in such area is critical, as a lack of adequate training can cause conflicts, customer dissatisfaction, and team fallout [11]. This program was the first exposure for many students to work with real clients; therefore, we provided them with a number of training sessions covering soft skills such as communication with clients; team management; professional conduct; equity, diversity, and inclusion (EDI); and leadership. This was done to ensure successful

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

interactions between students and clients such that positive relationships between them could be built, thus facilitating the growth and strengthening of their networks.

Due to the unique end users each project catered toward, eliciting requirements from clients and end users while also behaving in a professional and respectful manner was likewise an important skill to learn. Positive interactions with potential end users also contributed to students learning to successfully expand and integrate into their own networks. As a result of the provided soft skills training, community partners were immensely satisfied with the students as one of them stated: *"I thought they were very organized and professional."* This suggests a willingness to interact with our students again, which confirms that a strong rapport has been established.

As such, the community partners further connected them to individuals who could help them with their project. Describing their experience building connections, one student said: *"The connections are amazing; we couldn't have had those connections and get in touch with them as quickly as possible without [our community partner]."* This quote implicitly indicates the importance of communication skills for building connections. To help students in communication development, we trained them with techniques such as writing down constructive feedback for team members. *"This exercise has made it clear to me that communicating constructive feedback is something that I need to learn, and I hope that I may learn how to do it kindly."* This confirms that open communication, while difficult to establish, is critical for effective relationship-building, thus justifying a need for soft skills training to help students build their own professional networks.

Recommendations:

- Dedicate and emphasize a significant time toward soft skills training to help students build connections and develop a supporting network.

Lesson 2 – The right amount of guidance empowers students to balance autonomy and motivates them: While we encouraged our teams to be largely self-organized, this proved to be a delicate balancing act. We realized throughout the duration of this program that not enough autonomy would lead to feelings of being micromanaged. On the other hand, too much autonomy meant that students sometimes felt lost and were facing overwhelming uncertainty. We observed this closely in our cohort, with some students expressing a lot of anxiety and stress early in the semester when faced with independence: *"I would say, the very beginning with the whole trying to figure out research and stuff on the different keywords, I think we were all kind of in the*

same boat at that point, trying to like figure out what we're doing." Work by Noll et al. [13] supports these feelings, speculating that individuals with lower competence, such as our students at the start of the term before learning new skills, will not benefit from high levels of autonomy. For this reason, we front-loaded substantial technical, soft skills, and EDI training in the program so that every student would receive some initial guidance in a variety of soft and technical skills to increase feelings of competence.

As the term progressed, the contrary was observed through the increase in competence and relatedness of students in the program. Throughout the term, students developed a variety of skills and had the opportunity to bond with their team members. Team members often helped each other overcome different challenges or navigate knowledge gaps leveraging their diverse backgrounds and skillsets. As the term progressed further and students further developed their competence, they expressed an appreciation for autonomy; one student says, "*We're given the space to come together and kind of figure it out what's needed, in my opinion, as a team to kind of figure out how to grow together.*"

With the teams' progressing through changes in their own competence and relatedness, the teaching team required to adjust how much intervention was needed with the team's processes. We realized that students built up autonomy over and relied on our guidance less and less as the semester progressed. Thus, incorporate motivating the students to work in their own pace and empower them to succeed.

Recommendations:

- Provide adequate support to ensure students are not overwhelmed by autonomy and monitor student feedback to adapt if necessary.
- Consider changing skillsets of the students and adjust the level of support accordingly.

Lesson 3 – Proper structure enhances community-engaged collaboration between the students and the community partners: In this program, community partners' collaboration was crucial to the success of the experiential learning process. Students expressed feeling highly motivated as a result of working in a real project with the community partners. The students felt more accountable in producing a successful final product as this product would be deployed to the community. When comparing the program to a typical course, one student described, "*You just have an imaginary community partner or user requirements that doesn't change over time. It's just solid; they give you a problem statement, and you solve it.*" However, they expressed that in this

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

program “*the element of getting real people [had] a big role, because what [they] build might actually end up saving lives.*”

Despite real clients being so important to the students’ motivation and success, the community partner’s vision for the solution collided with the students’ skillsets. For some of the projects, the scope was so large that the students were required to conduct extensive user research in the first two months to define the scope to a workable state as well as consider their own skills and knowledge to set the scope. Project 5, for example, had a somewhat unclear and broader scope as mentioned by one of the students:

“*It’s clear that the project’s scope is significant and, at times, very daunting.*” Thus, they negotiated the scope of their project with the community partner. The community partner for this project mentioned having to bring students back on track as oftentimes they would deviate. “*We seem to have been pivoting and spinning our wheels a little bit more ... I do know that because of the scope of the existing problem. It was really huge. And to try and keep the team focused on just chunking out something small, as part of it, was a task in itself.*”

As a solution for such expectation conflicts, the community partners expressed the need for more guidance (written guidelines) regarding how much their involvement should be. Students conveyed similar needs. To facilitate this improvement, one of the community partners suggested “*more touch points across groups, offering everybody a chance to get together or something.*” They also expressed that it would have been helpful to “*hear from the other community partners to see how it’s going for them.*” Hence, more support is desired by both students and clients to make this collaboration more successful to build a supportive network.

Recommendations:

- Provide a guideline to the community partners to mitigate uncertainty regarding how they are expected to interact with students.
- Add more instruction and training in communication, negotiation, and scoping skills, since many students are engaging with community partners for the first time.

Lesson 4 – Through mentorship and EDI training, students learn to overcome challenges of working in a diverse team. Diversity has the potential to both benefit and hinder team performance [7]. One student described a frustrating experience in which they were encountering too many perspectives, saying, “*The most eye-opening*

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

thing to me is like, how we can have like, the same objectives but like, the same goals but like different ways and like solving the same problem." However, the EDI training helped them realize the importance of different views in a team. The EDI training consisted of targeted workshops where a EDI trainer from the university explained the various facets related to equity, diversity, and inclusion.² The training consisted of several exercises that enforced critical thinking on understanding one's own privileges and the use of correct wordings while engaging with different people. The concept of empathy and equity was a key discussion in this training.

In the early stages, the students would often prioritize their own work as "important" over others, which would result in frustration for the other members. This highlights the teams going through the storming phase of Tuckman's [16] model of group development. However, with adequate mentoring from their industry mentors and instructional team, they soon realized that it was necessary to learn to discard personal biases for the betterment of the project. One student said, *"I have a process or method that I have developed on my own and naturally think it is the best and most efficient system ever, but it's clashing with these other three or four [team members]. I have never worked in a group setting before where our views and opinions could differ so significantly on such a small detail, mostly fascinating than something to be concerned about. I do see what they mean and try to understand why that is truly the best solution, most of the time it is, which is really cool to see how the collaboration worked to create the most efficient solution."*

As a result of the continuous guidance on practicing EDI, soon the students started leveraging their team diversity through efficient work distribution. Diversity was a prominent part of the program, and practicing inclusion in teams was constantly encouraged through training and mentorship. Hence, by the end of the program, the students realized how impactful working in a diverse team can be. *"I have found working on a diverse team enjoyable. It has given me the opportunity to learn new things, as everyone is in a different discipline and has different specialties. My ability to learn from my teammates is made possible through the team culture, which encourages asking questions, getting feedback from each member, and offering as much assistance as possible."*

²www.uvic.ca/equity/education/workshops/index.php

Recommendations:

- Include explicit EDI training throughout the course of the project and emphasize the benefits of working with diverse teammates.
- Be prepared to provide mentorship to students as conflict is expected from time to time.

Conclusion

We presented a pioneering program that aimed to motivate students from underrepresented backgrounds to stay and succeed in computer science and software engineering through community-engaged experiential learning. Over the course of four months, the students first received soft skills and technical training from the instructor team. Consecutively, they engaged with the community partners in identifying and scoping the problem before conducting weeks of prototyping and solution validation. Solving the diversity problem is not a small feat, and we hope that our program design, lessons learned, and recommendations are useful for other universities and organizations looking to help tackle the issue in their own communities. The experiences described in this chapter represent the first step of our INSPIRE program aiming to help address diversity in computer science and software engineering. We summarize our lessons learned and recommendations as a number of takeaways (see Table 26-3) from our experience running the first cohort of INSPIRE.

Table 26-3. Takeaways

Lessons Learned	Recommendation
Training on soft skills facilitates building a successful network of diverse individuals in software engineering.	– Dedicate and emphasize a significant time towards soft skills training to help students build connections and develop a network.
The right amount of guidance empowers students to balance the autonomy and motivates them.	– Provide adequate support to ensure students are not overwhelmed by autonomy. – Consider how skilled the students are becoming and adjust the level of support accordingly.
Proper structure enhances the community engagement experience between the students and community partners.	– Provide a guideline to the community partners to mitigate uncertainty regarding their expected interaction with students. – Add more instruction and training in communication, negotiation, and scoping skills since many students are engaging with community partners for the first time.
Through mentorship and EDI training, students learn to overcome challenges of working in a diverse team.	– Include explicit EDI training throughout the course of the project and emphasize the benefits of working with diverse teammates. – Be prepared to provide mentorship to students as conflict is expected from time to time.

Bibliography

- [1] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis. Preprint at *arXiv:1709.08439*, 2017.
- [2] Sarah Andreas. Effects of the decline in social capital on college graduates soft skills. *Industry and Higher Education*, 32(1):47–56, 2018.

CHAPTER 26 SOFTWARE ENGINEERING THROUGH COMMUNITY-ENGAGED LEARNING AND AN INCLUSIVE NETWORK

- [3] William Aspray and Andrew Bernat. Recruitment and retention of underrepresented minority graduate students in computer science. In *Report on a Workshop by the Coalition to Diversity Computing*, 2000.
- [4] Carlo Barone. Some things never change: Gender segregation in higher education across eight nations and three decades. *Sociology of Education*, 84(2):157–176, 2011.
- [5] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, January 2006. Publisher: Routledge eprint: www.tandfonline.com/doi/pdf/10.1191/1478088706qp063oa.
- [6] Linda B. Glaser. Research offers new hope for gender equity in STEM fields. 2017.
- [7] Sujin K. Horwitz. The Compositional Impact of Team Diversity on Performance: Theoretical Considerations. *Human Resource Development Review*, 4(2):219–245, June 2005.
- [8] David A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development*. FT Press, 2014.
- [9] Scott A. Lee. Increasing student learning: A comparison of students' perceptions of learning in the classroom environment and their industry-based experiential learning assignments. *Journal of Teaching in Travel & Tourism*, 7(4):37–54, 2008.
- [10] Percival Lucena, Alan Braz, Adilson Chicoria, and Leonardo Tizzei. IBM design thinking software development framework. In *Brazilian Workshop on Agile Methods*, pages 98–109. Springer, 2016.
- [11] Valerie Martinelli. What Happens When Your Soft Skills Kill Your Career? April 2019.
- [12] Michael Miles, David Melton, Michael Ridges, and Charles Harrell. The benefits of experiential learning in manufacturing education. *Journal of Engineering Technology*, 22(1):24, 2005.

- [13] John Noll, Sarah Beecham, Abdur Razzak, Bob Richardson, Ann Barcomb, and Ita Richardson. Motivation and autonomy in global software development. In *International Workshop on Global Sourcing of Information Technology and Business Processes*, pages 19–38. Springer, 2017.
- [14] Polly S. Rizova. Are you networked for successful innovation? *MIT Sloan Management Review*, 47(3):49–55, 2006.
- [15] Kaela S Singleton, De-Shaine RK Murray, Angeline J. Dukes, and Lietsel NS Richardson. A year in review: Are diversity, equity, and inclusion initiatives fixing systemic barriers? *Neuron*, 109(21):3365–3367, 2021.
- [16] Bruce W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63(6):384, 1965.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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.

PART VI

Methodologies Supporting Studies of Diversity and Inclusion in Software Engineering

CHAPTER 27

How to Measure Diversity Actionably in Technology

Md Montaser Hamid, Oregon State University, USA.*

Amreeta Chatterjee, Oregon State University, USA.

Mariam Guizani, Queens University, Canada.

Andrew Anderson, Oregon State University, USA.

Fatima Moussaoui, Oregon State University, USA.

Sarah Yang, Oregon State University, USA.

Isaac Escobar, Oregon State University, USA.

Anita Sarma, Oregon State University, USA.

Margaret Burnett, Oregon State University, USA.

How do you measure technology's support for diverse populations in a way that is actionable and can lead to more inclusive designs of the technology? This chapter presents a method and the validated GenderMag survey that powers the method. The survey measures diversity gaps in technology in a fine-grained way, and the method shows how to use it to translate an empirical study's findings into actionable design directions.

Introduction

Measurement is the first step that leads to...improvement (IBM quality expert H. James Harrington) [11].

Many scientists and researchers, including us, agree with Mr. Harrington. When considering diversity, a reason for measurements is often a desire to change something to improve the support for diversity.

Our interest lies in measuring the diversity of a user population that a software system intends to support. Improving how well a software system supports diverse users in technology requires diversity measurements that are truly actionable – not just a demographic measurement (e.g., “we don’t support women as well as other people” or “only 37% of women would recommend our software, compared with 51% of other people”). Demography-based measurements can point out what features disproportionately affect diverse users and how often these issues arise but are incapable of explaining why these issues exist in the first place. Those why’s are the missing link that enables translating the empirical study findings into actionable design fixes.

To obtain those missing why’s, what is needed is a fine-grained measurement device that relates technology misfires with diverse individuals’ traits relevant to the usage of technology. Toward that end, we have developed a diversity measurement method based on the GenderMag facets enabled by a GenderMag facet survey. The GenderMag facets represent different cognitive styles that impact how individuals go about using technology, in which the differences (statistically) cluster by gender. The GenderMag facet survey provides a new, fine-grained method for understanding diversity gaps in technology and in technology-related artifacts (e.g., user interfaces, documentation, user manuals). Although our previous work used facet surveys, this is the first time we explain the exact steps of the scoring and the validation process. The survey enables (1) extracting information on who runs across which inclusivity bugs and why, (2) comparisons between a technology’s before and after diversity support, and (3) developers and designers to understand how to make the empirical results *actionable*.

Background: The GenderMag Facets

The GenderMag facet survey is a companion to the GenderMag method [2]. GenderMag is an evidence-based inclusivity evaluation method that software practitioners can use to find and fix inclusivity bugs. GenderMag has been used for a wide range of applications [4, 8, 9, 10, 12, 14, 15, 17].

At the core of GenderMag are five problem-solving styles called facets in GenderMag (Figure 27-1), each of which is backed by extensive foundational research [2, 16] and has a range of possible values. A few values within each facet's range are brought to life by the three GenderMag personas: "Abi," "Pat," and "Tim." Statistically, Abi's facets are more common among women and Tim's are among men, whereas Pat has a mix of Abi's/Tim's facets plus a few unique ones.

Each facet describes how different individuals approach problem solving when using technology. For example, some women may have a process-oriented learning style like Abi, which means they would prefer to learn new technologies in the context of a tutorial or an explicit process. When looking for information to progress, some individuals may be more selective (Tim's processing style) as in they pick the first promising option instead of reading through all the information. These facets can help designers pinpoint how to better support all genders within technology, and the facet survey enables them to measure a respondent's facet values (Figure 27-1).

The Facet Survey: What It Is

The GenderMag facet survey (Figure 27-2) is a validated Likert-scale survey that collects a respondent's particular facet values for each of the five facets in Figure 27-1. We initially created it as a part of a longitudinal field study at Microsoft that occurred in 2015–2016 [3].

At that time, Microsoft had just developed a strong interest in supporting diversity and inclusion within its *products* – not just its workforce climate. This timeframe coincided with the emergence of GenderMag and Burnett's sabbatical at Microsoft, and subgroups of Microsoft employees were considering using all or portions of it. However, GenderMag's generality and applicability to their products had not been established yet, and some employees wondered whether the facet distributions across genders that the GenderMag team had seen elsewhere really applied to their customers.

Self-Efficacy

Abi: Lower self-efficacy than their peers about unfamiliar computing tasks. If tech problems arise, often blames self and might give up as a result.

Tim: Higher self-efficacy than their peers with technology. If tech problems arise, usually blames the technology. Sometimes tries numerous approaches before giving up.

Pat: Medium self-efficacy with technology. If tech problems arise, keeps on trying for quite a while.

Motivations

Uses technology... *Abi:* Only as needed for the task at hand. Prefers familiar and comfortable features to keep focused on the primary task.

Tim: To learn what the newest features can help accomplish.

Pat: Like Abi in some situations and like Tim in others.

Learning Style

Abi: Learns best through process-oriented learning; (e.g., processes/algorithms, not just individual features).

Tim: Learns by tinkering (i.e., trying out new features), but sometimes tinkers addictively and gets distracted.

Pat: Learns by trying out new features, but does so mindfully, reflecting on each step.

Information Processing

Abi and Pat: Gather and read everything comprehensively before acting on the information.

Tim: Pursues the first relevant option, backtracking if needed.

Attitude Toward Risk

Abi and Pat: Risk-averse, little spare time; like familiar features because these are predictable about the benefits and costs of using them.

Tim: Risk tolerant; ok with exploring new features, and sometimes enjoys it.

Figure 27-1. The GenderMag facet types and their values for each persona [2]. The colors are used throughout this chapter to associate the survey questions/scoring with these facets

Microsoft's "Team C2" was the first to raise this question, and to answer it, they sought validation within their product's customers. Thus, they collaborated with the GenderMag team to develop and run a GenderMag facet survey, which is framed within the GenderMag method. This survey helped them in validating the GenderMag facet values and accompanying gender distributions in their customer base. The survey results also answered the question that Team C2 had sought to answer: whether the GenderMag facets were indeed pertinent to their own customers.

1. I am able to use <insert type of technology> when...
 - a. I have just the build-in help for assistance.
 - b. I have seen someone else using it before trying it myself
 - c. no one is around to help me if I need it.
 - d. someone else has helped me get started.
 - e. someone has shown me how to do it first.
 - f. I have used similar technology before to do the same task.
 - g. I have never used anything like it before.
2. I am not confident about my ability to use and learn <insert type of technology>. I have other strengths.
3. I make time to explore <insert type of technology> that is not critical to my job.
4. One reason I spend time and money on <insert type of technology> is because it's a way for me to look good with peers.
5. It's fun to try new <insert type of technology> that is not yet available to everyone, such as being a participant in beta programs to test unfinished <insert type of technology>.
6. I enjoy finding the lesser-known features and capabilities of the <insert type of technology> I use.
7. I explore areas of <insert type of technology> before it is time for me to use it.
8. I'm never satisfied with the default settings for my <insert type of technology>; I customize.
9. I want to get things right the first time, so before I decide how to take action, I gather as much information as I can.
10. I always do extensive research and comparison shopping before making important purchases.
11. When a decision needs to be made, it is important to me to gather relevant details before deciding, in order to be sure of the direction we are heading.
12. I avoid "advanced" buttons or sections in <insert type of technology>
13. I avoid activities that are dangerous or risky.
14. Despite the risks, I use features in <insert type of technology> that haven't been proven to work.

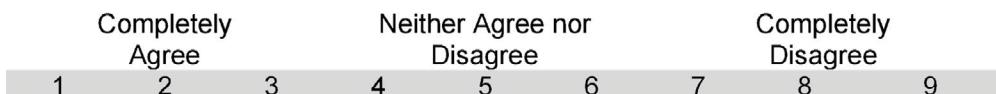


Figure 27-2. The facet survey. Top: Question colors indicate the facet being measured (Figure 27-1). Bottom: All questions use a nine-point Likert scale

More importantly, the survey results revealed a measurement benefit we hadn't anticipated: it offered a measure of diversity outcomes at a higher resolution than standard demographic measures. In this chapter, we define a *higher-resolution measure* as one that can discriminate between two points that, with a lower-resolution measure, cannot be discriminated. Applying this concept to diversity outcome measurements, suppose that 67% of women run into barriers with a particular feature of a technology product and 33% do not. What are the differences between someone in the 67% and someone in the 33% group other than the outcome? If all we have is their gender demographics, all we know is that they are women, and we cannot see their differences. However, if we also have their facet values, we can see differences that gender demographics alone cannot reveal.

The facet survey can be used in several ways, but the primary use we discuss in this paper is to obtain fine-grained measurements of diversity in an empirical investigation.

Scoring the Survey

After participants have responded to the facet survey, we can score their responses using the survey key in Figure 27-3. Since Abi and Tim are the personas who represent the endpoints of each facet's spectrum of possible values, we use those persona names to relate a participant's responses to these two endpoints. We score using the following steps:

Step 1 (Complement): Convert the answer scores to numbers from 1 (Completely disagree) to 9 (Completely agree). For some questions, closer to 9 is Tim-like. But the opposite is true for how Questions 2 and 9–13 are worded, so for these questions, “reverse” the participants’ responses to their tens’ complement (i.e., convert “9” to “1,” “8” to “2,” etc.).

Step 2 (Sum each facet): For each participant, sum the results of Step 1 for each facet. The colors in Figures 27-1 and 27-2 represent facets. This step results in five scores per participant, one score for each facet.

Step 3 (Calculate facet medians): The scores are not “absolute.” Rather, they are *relative to a participant’s peer group*. For example, a group of college students would be expected to have different levels of computer self-efficacy, different styles of learning technology, etc. than a group of retired people. To find the middle of the peer group (we assume a peer group is the participants recruited for the study), calculate the median “sum of scores” of all the participants from the same peer group, for each facet.

Step 4 (Tag each participant's facet score): To the right of each facet's median (above) is Tim-like; otherwise, it is Abi-like. If the participant's facet score is the facet median, then it is up to you to decide whether they are an Abi or Tim. You can decide on this in a way that helps balance the sample sizes, or you can add a third tag (Pat-like).

1a, 1b, 1c, 1d, 1e, 1f, 1g	High Self Efficacy (Tim)
2	Low Self Efficacy (Abi)
3, 4, 5	Motivations: Technology for its own sake (Tim)
6, 7, 8	Learning: Tinkerer (Tim)
9, 10, 11	Comprehensive Information Processing (Abi)
12, 13	Risk Averse (Abi)
14	NOT Risk Averse (Tim)

Figure 27-3. Facet survey key. The more strongly the participant agrees on a question, the closer their facet value is to the endpoint (persona name) shown

In the end, each participant has a five-tuple tag representing each facet. Most participants turn out to have a mix of facet values. For example, a participant might have self-efficacy, motivations, and a risk attitude closer to Abi's, but information processing style and learning style closer to Tim's. The scores calculated from the facet survey then can be used to analyze when a technology is failing to be inclusive, why it is so, and exactly who are affected by it, as we detail next.

From Scores to Understanding to Actions

We show how to analyze these scores in a way that points toward fine-grained understanding and then actionability using Team V as a running example [17]. Team V had two versions of a prototype: the “Before” version was the one currently in production usage, and the “After” version was a redesign to fix six inclusivity bugs of the Before version that Team V had found using the GenderMag evaluation method.

To understand in a fine-grained way their inclusivity progress, equity progress, and where design actions were still needed, Team V empirically evaluated both versions in a between-subject user study, in which Team V's participants responded to the facet survey and then worked their way through the prototype's main use cases.

Fine-grained diversity comparisons between versions: Team V used the facet survey to compare their Before vs. After versions' participants' encounter with the inclusivity bugs. Figure 27-4 aggregates the results for all six inclusivity bugs by counting

the facet responses of Team V's participants who faced the bugs. For example, if a Before participant had three Abi facets and two Tim facets, they would add 3 to the "Before" orange bar and 2 to the "Before" blue bar.

As the figure shows, in the Before version, Abi facets were more impacted by inclusivity bugs than Tim (34 Abi facets (●) vs. 26 Tim facets (■)). The After version reduced the facets impacted for both: Abi 13 (●) and Tim 17 (■). Therefore, the After version improved inclusivity for both Abi- and Tim-faceted users; but it was still not equitable since Tim-like facet values were more impacted than Abi-like facet values.

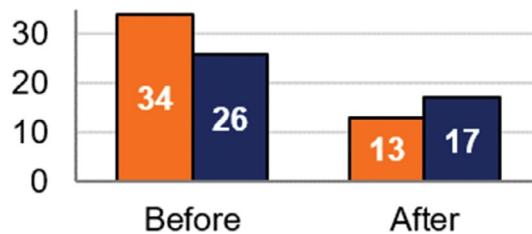


Figure 27-4. Number of observed facets in the facet survey responses of Team V's participants who faced inclusivity bugs. Orange: Abi facets. Blue: Tim facets

Fine-grained understanding of who and why: To understand how to fix a bug, such as Bug#4 (Figure 27-5), we first need to know who experienced it and why. In this bug, six of Team V's participants (Before1, Before2, Before4, Before5, Before8, and Before10) faced the inclusivity bug in the Before version and two (After7 and After10) in the After version. (The participants are ordered by the number of their Abi (●) vs. Tim (■) facet values, with Abi's at the top and Tim's at the bottom.)

In the Before version, six of Team V's participants spanning every facet value experienced Bug#4 difficulties, with 16 Abi-facet count (●) and 14 Tim-facet count (■). In the After version, only two of Team V's participants faced the bug, with the Abi-facet count (●) now down to 2 and the Tim-facet count (■) down to 8. This reduction means that the Bug#4 fixes brought inclusivity by improving the prototype for both Abi-and Tim-like users. But the After version did not achieve equity, with Tim-like facet values facing more difficulties than Abi's (8 ■ vs. 2 ●).

Participant	M	SE	R	IP	L		M	SE	R	IP	L	Participant
Before						After						After
Before1	●	●	●	●	●	-	-	-	-	-	-	After1
Before2	●	●	■	●	●	-	-	-	-	-	-	After2
Before3	-	-	-	-	-	-	-	-	-	-	-	After3
Before4	●	■	●	●	■	-	-	-	-	-	-	After4
Before5	●	■	■	●	●	-	-	-	-	-	-	After5
Before6	-	-	-	-	-	-	-	-	-	-	-	After6
Before7	-	-	-	-	-	■	●	■	●	■	■	After7
Before8	■	●	■	■	■	-	-	-	-	-	-	After8
Before9	-	-	-	-	-	-	-	-	-	-	-	After9
Before10	■	■	■	■	■	■	■	■	■	■	■	After10

Figure 27-5. Results of Bug#4 in [17]. Facets of the six Before and two After versions' participants with action failures. -: participant had no action failures. M=Motivations, SE=Self-Efficacy, R=Risk, IP=Info Processing, L=Learning

Where designers' actions helped and where more were needed: These counts showed that designers' Bug#4 remedies in the After version had been very successful: users with all five Abi-like facets and four Tim-like facets fared better than they had with the Before version. However, support for users with Tim-like *motivations* had not improved. This points designers directly toward designing further Bug#4 improvements to better support Tim's motivations (without sacrificing support for Abi's motivations); Guizani et al.'s "Why/Where/Fix" inclusivity debugging approach gives examples of how to do this [10].

To summarize, the Team V example shows how the facet survey can enable fine-grained diversity comparisons between two versions, fine-grained understanding of "who" are being left out and their facet values, and shows designers where to take action by fixing inclusivity bugs based on the facet values of who's still being left out.

How We Validated the Survey

To validate the survey, we followed these steps, but they were intertwined with each other and with the creation process:

Step 1:(Pre-validation) Started with questions from other validated surveys

Step 2: (Reliability) Ran the survey and assessed response consistency using Cronbach alpha tests

Step 3: (Cross-validation) Cross-analyzed results from administering to other populations, intertwined with Step 4

Step 4: (Cluster analysis + condense) Cluster analyses to reduce the number of questions needed

Step 5: (Demographic validation) Quantitative comparison of facet responses with participants' gender identities

Step 6: (Empirical) A validation study comparing participants' survey responses with their verbalizations while working with the technology

Much of this intertwined process was a joint effort with Microsoft's Team C2 [3]. In Step 1, we worked with Team C2 in a formative way, which we'll refer to as pre-validation. In this step, we drew applicable existing questions from other validated surveys/questionnaires in the literature. This approach provided about two-thirds of the questions from established, validated questionnaires such as [6]. Although excerpting portions of a validated questionnaire cannot bring "validation" to the new questionnaire, the strong provenance of those excerpts enabled an evidence-based start to the survey. For facets with no validated questionnaire, we had to develop pertinent questions ourselves by drawing on existing research as much as possible. Stumpf et al.'s summary of gender-meets-technology literature covers much of the research base from which we drew [16].

After several iterative improvements, in a 2015 study, Team C2 ran the survey on 500 men and 500 women who were Microsoft's customers. For Step 2 (Reliability), we analyzed their results using Cronbach alpha tests [7], a widely used way of measuring the reliability of a set of questions. The results validated the survey's inter-item reliability. Specifically, the results were above the 0.8 level for two of the five facets (Information Processing Style and Self-Efficacy), above the 0.7 level for two others (Motivations and Risk), and at 0.691 for Learning Style. Cronbach alpha's above 0.8 is generally considered to be good and above 0.7 to be acceptable, but Churchill also argues that 0.6 should also be considered acceptable [5].

Word of the survey spread, and by Step 3 (Cross-validation), other interested teams began to work together to share and cross-analyze survey results. At the same time (Step 4, Cluster analysis and condense), several Microsoft data scientists got involved to whittle down the number of questions needed by analyzing responses, so as to reduce possibility of survey fatigue. One team also validated their survey with user interviews and think-aloud studies.

One validity question that needed to be answered was whether the facet values reported by Team C2's 1,000 survey respondents differed (quantitatively) by gender identity (Step 5, Demographic validation). One qualitative study in 2019 to answer this

question involved 20 participants [17]; another in 2021 involved 1,000 participants [1]. As Figure 27-6 shows, these participants' facet values did cluster by their gender where women skewed more toward Abi than men did.

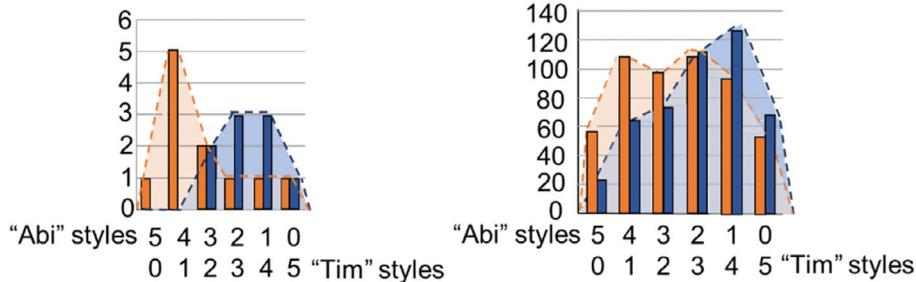


Figure 27-6. Facet survey results for two genders. x-axis: # of facet values scored as (top row): "Abi"-like; and (bottom row): "Tim"-like. y-axis: # of participants. Example: The bar at "5 0" shows the number of participants with all five Abi-like facet values. (Left chart, [17]; right chart, [1])

Finally (Step 6, Empirical), in a 2022 think-aloud study [10], we compared participants' facet survey responses with their in situ verbalizations during a think-aloud problem-solving task. Figure 27-7 shows results from this comparison. When an outline color (their in situ verbalized facet value) is the same as the shape's fill color (their survey response), then their survey response matched that participant's verbalized facet value in that moment of their work. In total, 78% of participants' in-the-moment facet verbalizations aligned with their facet survey responses, which suggests that the facet survey was a reasonable measure of participants' actual facet values.

	Motiv	SE	Risk	Info	Learn	Motiv	SE	Risk	Info	Learn	
	Original	DiversityEnhanced									
P1	●	□	■	□	●	-	-	-	-	-	P10
P2	●	□	●	■	●	-	-	-	-	-	P11
P3	●	■	●	●	■	-	-	-	-	-	P12
P4	●	■	●	●	■	-	-	-	-	-	P13
P5	■	●	●	■	●	■	□	●	●	■	P14
P6	■	●	■	●	■	-	-	-	-	-	P15
P7	■	■	●	●	■	-	-	-	-	-	P16
P8	■	■	●	■	■	-	-	-	-	-	P17
P9	■	■	■	■	■	-	-	-	-	-	P18

Figure 27-7. Think-aloud study participants [10] who ran into one set of inclusivity bugs with their facet values, validated with their in situ responses. ●/■: the facet scores from the participants' survey responses for Abi-like and Tim-like facet values, respectively. □/■: Abi-like/Tim-like facet values participants verbalized in situ when they ran into a bug

Key Takeaways

The key takeaways from this chapter are as follows:

Fine-grained diversity measurements: The GenderMag facet survey measures technology's diversity gaps in a fine-grained way, showing not only who experiences which inclusivity bugs but also *why* they experience each inclusivity bug they encounter.

Fine-grained comparisons: The survey enables comparisons between different prototype versions showing not only which version is more inclusive but also why and for whom that version is more inclusive.

Actionable: The why's are *actionable*: designers can design fixes to an inclusivity bug around the facet values of those experiencing it.

Validated: The survey has been thoroughly validated.

The survey has uses beyond measurement, such as to select a facet-diverse set of participants [10] or for team building [13], but its main purpose is measuring diversity *actionably*. We invite researchers, developers, and designers everywhere to use it to gain new insights into both how to address their technology's diversity failures and how to repeat their technology's diversity successes.

Acknowledgments

We thank the many people who have helped improve the survey. We thank Robin Counts, Hannah Hanson, and Ronette Lawrence. Robin initiated the idea of a facet survey, and Ronette enabled Microsoft's support. Robin and Hannah not only contributed substantially to the research and development of its questions but also to the administration and analysis of the survey's first trials. Thanks to Ronette, Microsoft permitted us to freely share the survey questions. This work has been supported in part by Microsoft, by National Science Foundation grants 1901031 and 2042324, and by USDA-NIFA/NSF grant 2021-67021-35344. Views expressed in this chapter are those of the authors and not necessarily those of the sponsors.

Bibliography

- [1] Anderson A, Li T, Vorvoreanu M, and Burnett M. (2022). Diverse humans and Human-AI interaction: What cognitive style disaggregation reveals. <https://arxiv.org/abs/2108.00588v3>
- [2] Burnett M, Stumpf S, Macbeth J, Makri S, Beckwith L, Kwan I, Peters A, and Jernigan W. (2016). GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers* 28(6):760–787. <https://doi.org/10.1093/iwc/iwv046>
- [3] Burnett M, Count R, Lawrence R, and Hanson H. (2017). Gender HCI and Microsoft: Highlights from a longitudinal study. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 139–143. <https://doi.org/10.1109/VLHCC.2017.8103461>.
- [4] Chatterjee A, Letaw L, Garcia R, Reddy DU, Choudhuri R, Kumar SS, Morreale P, Sarma A, and Burnett M. (2022). Inclusivity bugs in online courseware: A field study. *2022 ACM Conference on International Computing Education Research – Volume 1 (ICER '22)*, ACM, New York, NY, USA, 356–372. <https://doi.org/10.1145/3501385.3543973>

- [5] Churchill GA. (1979). A paradigm for developing better measures for marketing contrasts. *Journal of Marketing Research* 16(1):64–73. <https://doi.org/10.2307/3150876>
- [6] Compeau DR and Higgins CA. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly* 19(2):189–211. <https://doi.org/10.2307/249688>
- [7] Cronbach LJ. (1971). Test validation. In: Thorndike RL (ed), *Educational measurement*. American Council on Education, Washington DC, USA, 443–507.
- [8] Cunningham SJ, Hinze A, and Nichols DM. (2016). Supporting gender-neutral digital library creation: A case study using the GenderMag Toolkit. *2016 International Conference on Asian Digital Libraries (ICADL '16)*, Springer, Cham, Switzerland, 45–50. https://doi.org/10.1007/978-3-319-49304-6_6
- [9] Gralha C, Goulão M, and Araújo J. (2019). Analysing Gender Differences in Building Social Goal Models: A Quasi-experiment. *2019 IEEE International Requirements Engineering Conference (RE '19)*, IEEE, New York, NY, USA, 165–176. <https://doi.org/10.1109/RE.2019.00027>
- [10] Guizani M, Steinmacher I, Emard J, Fallatah A, Burnett M, Sarma A. (2022). **How to debug inclusivity bugs? A debugging process with Information Architecture**. *2022 IEEE/ACM International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS '22)*, IEEE, New York, NY, USA, 90–101. <https://doi.org/10.1145/3510458.3513009>
- [11] Harrington HJ, Hoffherr GD, Reid RP, and Harrington R. (1999). *Area activity analysis*. McGraw-Hill, New York, NY, USA.
- [12] Kanij T, Grundy J, McIntosh J, Sarma A, and Aniruddha G. (2022). A new approach towards ensuring gender inclusive SE job advertisements. *2022 IEEE/ACM International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS '22)*, IEEE, New York, NY, USA, 1–11. <https://doi.org/10.1145/3510458.3513016>

- [13] Letaw L, Garcia R, Garcia H, Perdriau C, and Burnett M. (2021). Changing the online climate via the online students: Effects of three curricular interventions on online CS Students' inclusivity. 2021 ACM Conference on International Computing Education Research (ICER '21), ACM, New York, NY, USA, 42–59. <https://doi.org/10.1145/3446871.3469742>
- [14] Oleson A, Mendez C, Steine-Hanson Z, Hilderbrand C, Perdriau C, Burnett M, and Ko AJ. (2018). Pedagogical content knowledge for teaching inclusive design. 2018 ACM Conference on International Computing Education Research (ICER '18), ACM, New York, NY, USA, 69–77. <https://doi.org/10.1145/3230977.3230998>
- [15] Shekhar A and Marsden N. (2018). Cognitive walkthrough of a learning management system with gendered personas. 2018 ACM Conference on Gender & IT (GenderIT '18), ACM, New York, NY, USA, 191–198. <https://doi.org/10.1145/3196839.3196869>
- [16] Stumpf S, Peters A, Bardzell S, Burnett M, Busse D, Cauchard J, and Churchill E. (2020). Gender-inclusive HCI research and design: A conceptual review. Foundations and Trends in Human-Computer Interaction 13(1):1–69. <https://doi.org/10.1561/1100000056>
- [17] Vorvoreanu M, Zhang L, Huang Y-H, Hilderbrand C, Steine-Hanson Z, and Burnett M. (2019). From gender biases to gender-inclusive design: An empirical investigation. 2019 SIGCHI Conference on Human Factors in Computing Systems (CHI '19), ACM, New York, NY, USA, 1–14. <https://doi.org/10.1145/3290605.3300283>



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 28

How to Ask About Gender Identity of Software Engineers and “Guess” It from the Archival Data

Alexander Serebrenik, Eindhoven University of Technology, The Netherlands.*

Multiple studies of gender in software engineering require identifying gender of the individuals involved either by asking them (when conducting interviews and surveys) or by “guessing” it from archival data recorded in software repositories. In this chapter we discuss ways to ask about gender in surveys and interviews as well as three groups of automated genderization approaches proposed in the literature: name-to-gender, face-to-gender, and artifact-to-gender. For each one of the approaches, we discuss the way they work, the associated ethical concerns, the reliability and accuracy concerns, and the assumptions they make.

Introduction

When it comes to studies of diversity in software engineering, gender is by far the most studied diversity dimension: 61% of the scientific studies recently surveyed by Rodriguez-Perez et al. have considered gender [34]. Indeed, previous studies have shown that women participating in open source projects disengage faster than men [32], that while women concentrate their work across fewer projects and organizations,

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

men contribute to a higher number of projects and organizations [14], men and women follow different comprehension strategies when reading source code [40], and men tend to switch more frequently between debugging strategies [8]. Several studies in this volume also consider gender as a variable of interest: for example, Hashmati and Penzenstadler in Chapter 5, “How Users Perceive the Representation of Non-binary Gender in Software Systems: An Interview Study,” report on an interview study of representation of gender in software; Gama et al. in Chapter 16, “Toward More Gender-Inclusive Game Jams and Hackathons,” focus on experiences of transgender (binary and non-binary) and gender-nonconforming people related to jams and hackathons; Kohl and Prikladnicki in Chapter 11, “Gender Diversity on Software Development Teams: A Qualitative Study,” conduct a survey of gender diversity in software development teams; Simmonds et al. in Chapter 23, “Rethinking Gender Diversity and Inclusion Initiatives for CS and SE in a University Setting,” discuss the findings of the focus group of women and non-binary students; and Happe in Chapter 25, “Effective Interventions to Promote Diversity in CS Classroom,” studies frustrations steering women away from computer science. All these studies require the researchers to obtain information about gender identity of the study participants (for controlled experiments, interviews, and surveys) or of the individuals that have contributed to the dataset analyzed (for data-driven archival studies such as repository mining). As we are going to see in the following, obtaining such an information is fraught with challenges, and inappropriate ways of doing this might both alienate study participants and threaten validity of the scientific results. The challenges are not limited to researchers: indeed, everyone conducting internal surveys, performing marketing analysis, adding “gender” questions in the user interface of the software, or aiming at understanding user satisfaction necessarily has to find their way of recording information about gender identity.

To support both researchers and practitioners, in this chapter we take a look at the techniques used to obtain information about gender and the associated advantages and challenges.

Before we even start discussing how information about gender can be obtained, one has to remember that gender is a complex social construct of norms, behaviors, and roles that varies between societies and over time. Hence, study of gender in general, gender identities, or gender expression of individuals should be done with utmost care. Whatever technique we use, we should keep in mind that gender is privacy sensitive and should be treated as such even if such regulations as the General Data Protection Regulation (GDPR) do not consider this information as sensitive. In particular, open source contributors might be hiding their gender on purpose, for example, many women

developers prefer not to disclose their gender due to safety concerns. Moreover, some open source projects do not necessarily want us to know the genders of their members (but some do!), and companies might be sensitive to this topic as well.

Talking to People

One of the most popular ways of obtaining information about gender is asking the individuals themselves, as part of a survey or an interview. We should keep in mind, however, that reliability of this method strongly depends on the ability of the respondents to understand the question and find an answer corresponding to the way they see themselves.

In *Asking Questions: The Definitive Guide to Questionnaire Design*, Bradburn et al. [6] suggest recording the respondent’s gender by asking, “What is your/NAME’s sex?” and offering two answer options, male and female. This question conflates biological sex and socially constructed gender and reduces the spectrum of options to merely two. However, by now it is well known that both the biological reality of sex and the social reality of gender are much more complex [17]. For example, the recent survey of Stack Overflow indicates that 1.42% of software developers identify as non-binary, genderqueer, or gender nonconforming and 0.92% prefer to self-describe.¹ In the survey of the Linux Foundation, 4% of the respondents have indicated their gender as “non-binary/third gender.”² Surprisingly, in December 2018, a popular survey platform SurveyMonkey was still offering “female” and “male” as the only options for the “What is your gender?” question [43].

Hence, at the very least, the phrasing of the question about gender should reflect existence of genders other than women and men. One of simplest ways of phrasing such a question would be “Are you...male, female, something else? Specify ____.” In December 2018, a similar phrasing has been the default in Google forms [43], and a similar question has been used by Roberts et al. in a 2022 study of Australian adolescents’ eating pathology [33]. Such a phrasing is profoundly problematic as it expresses a preference toward “male” and “female” pushing all other gender identities outside of the norm – this process is known as *othering* [10, 47], “differentiating discourses that lead to moral

¹<https://insights.stackoverflow.com/survey/2021#developer-profile-demographics>

²www.linuxfoundation.org/wp-content/uploads/LFResearch_DEISurvey_ResultsDeck_121321.pdf

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

and political judgments of superiority and inferiority between ‘us’ and ‘them’” [12]. Moreover, by allowing the respondents to select only one answer option, this phrasing excludes people who are, for example, women *and* non-binary. Finally, in the empirical evaluation performed by Bauer et al. [3], cisgender participants had no problems answering this question, but transgender participants tried to understand what exactly the researchers were asking and reached different conclusions: both transfeminine (assigned male at birth and identify as women/non-binary) and transmasculine (assigned female at birth and identify as men/non-binary) respondents have given all three possible answers (male, female, other) rendering this question useless. When used in the interviews, this item was cognitively taxing for transgender interview participants [3].

The previous discussion suggests that (a) one should avoid referring to certain gender identities as “other”; (b) if answer options are provided, respondents should be able to select several options; and (c) the phrasing should explicitly refer to gender identity. Several proposals satisfying these requirements have been made in the literature. For example, Spiel et al. [43] recommend asking, “What is your gender?” with the following five checkboxes: “woman,” “man,” “non-binary,” “prefer not to disclose,” and “prefer to self-describe.” If the last option is checked, a free-form field opens up. Nikki Stevens, author of the Open Demographics project,³ suggests phrasing this question as “Where do you identify on the gender spectrum?” followed by a list of 30 gender identities taken from *The ABC’s of LGBT+* by Ashley Mardell [24], as well as “prefer not to answer” and “self-identify: ____.” One should be aware, however, that a lengthy list of gender identities might be experienced as confusing and take too much time if used as part of a larger survey.

Instead of offering answer options, one might also ask an open question as recommended by Scheuerman et al. in “HCI Guidelines for Gender Equity and Inclusivity.”⁴ In this case researchers will be required to manually code the responses, so the expected number of participants should not be too large, which is often the case for software engineering surveys. Moreover, open questions might elicit absurd reactions such as “bagel” or aggressive reactions such as “attack helicopter” originating from a meme ridiculing non-binary gender identification [16].

³<http://nikkistevens.com/open-demographics/questions/gender.html>

⁴www.morgan-klaus.com/gender-guidelines.html#Surveys

Summary When conducting interviews or small surveys, and the risk of aggressive or absurd responses is deemed small, prefer an open question such as “Where do you identify on the gender spectrum?” For larger surveys or surveys of populations that are more likely to provide absurd or aggressive responses, consider offering the following five checkboxes: “woman,” “man,” “non-binary,” “prefer not to disclose,” and “prefer to self-describe ____.”

Mining Software Repository Data

Repository mining studies analyze contributions from tens of thousands [32] to tens of millions of individuals [35]. This wealth of data allows one to carefully distinguish fine-grained statistical effects or perform longitudinal studies spanning over 50 years. However, when analyzing these amounts of data, it becomes no longer feasible to contact every single individual and ask them about their gender identity. In case of longitudinal studies, individuals might have retired or passed away; in case of large-scale studies of contemporary software development practices, contacting tens or hundreds of software developers might be technically possible, but it will likely lead to community disengagement, threatening the already low response rates [41]. To address this challenge, multiple tools have been proposed to automatically obtain gender information from the way developers present themselves, for example, by selecting their username or an avatar, or from the artifacts they produce such as source code or comments. The tools can be broadly classified as name-to-gender, face-to-gender, and artifact-to-gender. Many of these tools have not been designed with the software engineering data in mind, but software engineering data has its own peculiarities we discuss in the following.

Name-to-Gender

As Bradburn et al. [6] have put it, “<s>ometimes a person’s gender is obvious from his or her name.” Phrasing this more carefully, we can say that many cultures tend to associate specific names with specific genders: For example, Božidar is a Bulgarian name commonly given to men, while Nijol is a Lithuanian name commonly given to women. At the same time, טל (Tal) is a Hebrew name that can be given to a child of any

gender. In their most basic form, name-to-gender tools merely look up a given name in lists of names typically associated with women and men and return “woman,” “man,” or “unknown” depending on relative prevalence of a certain name within a specific gender. Prevalence is sometimes used to express degree of confidence of the tool in the gender inferred. For example, genderize.io states “male” with 0.99 confidence for “bozidar,” “female” with 0.99 confidence for “nijole,” and “male” with 0.68 confidence for “tal”.

However, this kind of basic approach fails to take into account differences between cultures: for example, Andrea is more commonly associated with men in Italy and with women in Germany, while Karen is mostly associated with women in the United States and with men in Armenia. International collaboration means that the same software engineering project or the same software engineering dataset might involve contributors from different cultures. This requires more advanced name-to-gender genderizers to take the cultural background into account. genderComputer that has been designed to analyze Stack Overflow data uses location as a proxy for cultural background [45]. However, less than 20% of Stack Overflow users in the sample analyzed by Vasilescu et al. have indicated their location, and location as indicated by users does not necessarily correspond to an actual geographic location (e.g., The Matrix) [45]. Moreover, using location as a proxy for national culture fails to take into account immigration-related effects.

This is why Namsor⁵ uses the individual’s surname as a proxy for national culture. This allows Namsor to infer that Andrea Rossini is (more likely) to be a man, while Andrea Parker is (more likely) to be a woman.⁶ This also makes Namsor one of the most accurate name-to-gender tools [36, 38]. Closer inspection reveals a different story, however. Santamaria and Mihaljević [36] reported that confidence of Namsor is almost perfect for European names, but the median confidence drops to 70% for Asian names. In particular, Eastern and Southeastern Asian names are difficult to genderize. Half of the East Asian names have a confidence score of 0, indicating that Namsor is essentially guessing randomly. This Eurocentric bias is problematic when trying to apply automatic gender inference techniques to software developers: the recent Stack Overflow survey shows that almost one out of four software developers have indicated different Asian regions as their ethnic background; in particular, 4.2% of the respondents are East Asian and 4.39% Southeast Asian.⁷ Recognizing this limitation of Namsor, Qiu et al.

⁵<https://namsor.app/>

⁶<https://namesorts.com/api/>

⁷<https://insights.stackoverflow.com/survey/2021>

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

combined it with genderComputer and designed a classifier trained on public name lists and celebrity name lists [32]. The features of this classifier included the last character (e.g., in Spanish, names ending in *a* are usually associated with women), the last two characters (e.g., in Japan, names ending in *ko* are usually associated with women), and tri-grams and 4-grams to capture romanized Chinese, Japanese, and Korean names. The combined name-to-gender tool outperformed both genderComputer and Namsor: for example, accuracy on Chinese names was 60% as opposed to 7% of Namsor and 18% of genderComputer [32]. A similar, character-based approach has been combined with deep learning models by Hu et al. [13]. The work of Qiu et al. has also inspired the Namsor developers to further develop special techniques for Chinese⁸ and Japanese names.⁹ Still, a 2022 study of Sebo shows that even for the current version of Namsor, the overall proportion of errors (misclassifications and non-classifications) is 53% [39]. What is even more problematic is that Namsor tends to perform worst for names associated with women as opposed to those associated with men (19.2% of the former names have been categorized correctly as opposed to the 66.5% of the latter) [39].

However, with all the improvements, Namsor cannot be applied if the individual is known by a mononym: for example, an Indian-American scientist and educator Govindjee is known by a single name only. This also limits the applicability of Namsor to such datasets as Stack Overflow: 43% of the Stack Overflow usernames do not use spaces and hence cannot be analyzed using Namsor. Since genderComputer has been designed for Stack Overflow, it implements several heuristics targeting software developers. In particular, if the name cannot be easily split into first name(s) and last name(s), genderComputer assumes it is formatted according to common naming conventions for usernames (e.g., “johns” for “John Smith”) [4] and restarts the genderization process (e.g., with “john” derived from “johns”) [45].

Another limitation of all the aforementioned approaches is their inability to take age into account. Indeed, for example, in Pennsylvania such names as Morgan and Robin that have been predominantly associated with men in the past have evolved to being associated with people of any gender and more recently to be more commonly associated with women [2].

Summarizing the discussion of name-to-gender tools, we can say that multiple name-to-gender tools have been developed by open source practitioners, academic researchers, and company-based software engineers. These tools tend to approximate

⁸<https://chinese-names.app/gender>

⁹<https://japanese-name.app/>

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

cultural background by analyzing the location or the surname of the individual. While age might have affected popularity of names among individuals of different genders, to the best of our knowledge, no currently available name-to-gender tool takes age into account.

To conclude this discussion, we list several examples of name-to-gender tools. As providing a complete overview of those tools would not be feasible, Table 28-1 only lists examples of the name-to-gender tools that (a) are available at the moment of writing and (b) have been empirically evaluated in scientific publications *other* than the paper that has introduced those tools.

Table 28-1. Examples of name-to-gender tools

Tool	URL	Empirical Evaluation
Gender API	https://gender-api.com/en/	[5, 27, 36, 38, 39]
genderComputer [45]	https://github.com/tue-mdse/genderComputer	[22, 30, 32]
gender-guesser [28]	https://pypi.org/project/gender-guesser/	[5, 36]
genderize.io	https://genderize.io/	[5, 27, 30, 36, 38]
Genni [41]	http://abel.lis.illinois.edu/cgi-bin/genni/search.cgi	[30]
NameAPI	www.nameapi.org/	[27, 36]
Namsor	https://namsor.app/	[27, 32, 36, 38, 39]
Wiki-Gendersort [7]	https://github.com/nicolasberube/Wiki-Gendersort	[38, 39]

Face-to-Gender

Another way developers present themselves on social platforms such as Stack Overflow and GitHub is by using avatars. This means that face recognition techniques such as

Facelytics,¹⁰ Face Analysis by Visage Technologies,¹¹ and PicPurify¹² can be applied to the avatars to identify gender of the individuals on these avatars. Indeed, on the task of identifying gender of Stack Overflow users based on their avatars, a face-to-gender tool Face++ has been shown to have a performance comparable to genderComputer [22], while on different avatar datasets, Face++, Amazon, and MS achieve more than 90% accuracy when identifying gender based on automatically detected faces [18]. However, not all faces can be correctly detected: in the study of Jung et al. [18], the very best tool has correctly identified faces in merely 76% of the analyzed images. Moreover, not everybody has a profile picture representing a human face. For instance, approximately 30% of the Stack Overflow users only have a default profile picture automatically generated based on the MD5 hash of the user’s mail, rendering approximately 70% of the Stack Overflow users *possibly* amenable for face-to-gender inference. However, not all Stack Overflow profile images represent faces (rather than logos or cat pictures). This is why Lin and Serebrenik have carefully selected 900 non-generated profile images of users of different ages and reputations and classified them manually. Reputation classes were selected according to different privileges Stack Overflow users might have; age intervals according to the general distribution of the ages on Stack Overflow. Among the 900 profile images, only 53% represent faces [22], suggesting that overall face-to-gender tools might be applicable to approximately $37\% = 70\% * 53\%$.

Artifact-to-Gender

Artifact-to-gender tools are based on the assumption that people of different genders express themselves differently in writing. Not surprisingly, the lion’s share of the research in this area has been based on personal writing on social media such as tweets and Facebook posts [21]. For example, the work of Company and Wanner [42] has been designed in the first place for attribution of authorship of blog posts and novels to one of the authors within a predefined set, and then the same technique has been retrained to predict gender of the author. Authorship attribution techniques have been designed for the source code as well [11, 15]; similarly to Company and Wanner [42], they are aiming at associating code fragments with one of the authors from the predefined set of approximately 100–160 candidates. This shows that deanonymization of source code is

¹⁰www.facelytics.io/en/

¹¹<https://visagetechnologies.com/face-analysis/>

¹²www.picpurify.com/demo-face-gender-age.html

possible despite a much more constrained use of language compared to social media texts. This is why Naz and Rice have applied similar techniques to predict gender of the authors. On a dataset of 100 student assignments, their approach has achieved the accuracy of 72% [29]. It remains to be seen whether these techniques can scale to tens of thousands of contributors common in repository mining studies.

Limitations and Concerns

The automated techniques discussed in the previous sections have shown that gender-related information can be obtained from such names, avatars, and code/text written. However, we need to remember that these methods are far from being perfect and one has to be very careful when applying them.

The first group of concerns are ethical. They are mostly raised in relation to face-to-gender techniques, but similar concerns can be raised for any automated genderization methods and are related to assigning any kind of categories to human beings without their explicit consent. While as humans we might be assigning categories to other people continuously, for example, when we are describing people, this kind of automation might be dangerous, for example, what if a tool recognizes a woman driving a car in a country where women are not allowed to drive cars? In fact, *Nature* has surveyed approximately 500 researchers in facial recognition, CS, and AI, and about two-thirds believe that application of facial recognition methods to recognize or predict personal characteristics (such as gender, sexual identity, age, or ethnicity) from appearance should be done only with the informed consent of those whose faces are used or after discussion with representatives of groups that might be affected [31]. Getting an informed consent from all GitHub or Stack Overflow developers is, of course, not realistic. Furthermore, individuals do not necessarily want to disclose their gender and sometimes take steps to hide it: one of the developers surveyed by Vasilescu et al. stated that they “have used a fake GitHub handle (my normal GitHub handle is my first name, which is a distinctly female name) so that people would assume I was male” [46]. In this case “correct” genderization would explicitly contradict the individual’s intention, which can hardly be seen as ethical.

The second concern is related to the gender binary assumption perpetrating the automatic techniques discussed previously. These are percentages of papers reviewed in two meta-studies. Keyes has shown that 92.9% of papers that introduce automatic face-to-gender tools assume gender binary, and this is also the case 96.7% of papers that use

automatic gender recognition [19]. For artifact-to-gender literature surveyed by Krüger and Hermann, this percentage goes up to 100% [21]. Finally, name-to-gender tools are doing a bit better: while they are still ignorant of non-binary genders, they at least tend to provide confidence scores, that is, they at least recognize their own lack of confidence [36]. Due to this gender binary assumption, automatic genderization tools can harm non-binary individuals as well as individuals with a limited ability to appear and be treated as their preferred gender [37].

Third, both the applicability and the accuracy of the techniques are not perfect. Restricted applicability might bias conclusions of a study since it is based only on data that the tools could analyze. Moreover, applicability and accuracy can be even lower for some subcommunities, for example, for Chinese names, when some of the gender-specific information is lost during the romanization.

All these reasons can lead to tools assigning an individual a gender that they do not agree with (e.g., because they do not want to disclose it, because this gender cannot be identified by the tool, or because the tool is imprecise), a problem known as *misgendering*, which can be seen as a form of verbal violence [26]. This is why we believe that (a) automated techniques should never be applied at the level of an individual subject but only at the level of large groups, (b) techniques should not be showing unequal performance on specific groups (e.g., if we know that name-to-gender techniques underperform on Asian names, conclusions based on application of these techniques to Asian names might be wrong), and (c) one has to continuously reflect on potential risks of the application of these techniques.

Summary Automated tools are necessary when analyzing large-scale data. When using the tools, one should never apply them at the level of an individual subject, but only at the level of large groups, and either ensure that performance of the tools is equal across different subpopulations or recognize unequal performance as a threat to validity of the conclusions derived.

Beyond Software Engineering

Several insights discussed previously can be also applied outside of the realm of software engineering. As the guidelines related to interviews and surveys are borrowed from the field of Human-Computer Interaction, they can be expected to be applicable to *any*

interview and survey looking to collect information about gender. Similarly, techniques discussed in the context of mining software repositories are applicable to analysis of any large-scale archival data ranging from social media sites such as Twitter and Facebook [20] to corpora of scientific publications [23], from a movie-related knowledge-sharing platform [25] to museum catalogs [44], and from Wikipedia [1] to collections of crowd-sourced recommendations [9]. Application of those techniques beyond software engineering might, however, require rethinking the aforementioned limitations and concerns as their relevance and importance might depend on the application domain.

Summary Aforementioned techniques can be applied beyond software engineering, but their application might require careful rethinking the aforementioned limitations and concerns.

Conclusions

Gender and gender diversity are popular topics in contemporary software engineering research. To conduct this research, one has to identify gender of the individuals involved. To this end we have discussed two large groups of identifying the contributor’s gender: by asking questions and by applying algorithmic tools. None of the techniques is perfect: questionnaires do not scale, and algorithmic tools guessing gender from GitHub information assume gender binary. Choice of the technique should, of course, be made in function of the research questions one is trying to answer. However, it might be equally important to discuss the limitations and problems of these techniques (and not only their advantages that made us choose them in the first place).

Summary

- For interview studies and small surveys, ask an open question: “Where do you identify on the gender spectrum?”
- For larger surveys use the same phrasing and the following five checkboxes: “woman,” “man,” “non-binary,” “prefer not to disclose,” and “prefer to self-describe ____.”

- When mining repositories evaluate name-to-gender and face-to-gender tools and either ensure that performance of the tools is equal across different subpopulations or recognize unequal performance as a threat to validity of the conclusions derived.
-

Bibliography

- [1] David Bamman and Noah A. Smith. Unsupervised Discovery of Biographical Structure from Text. *Transactions of the Association for Computational Linguistics*, 2:363–376, October 2014.
- [2] Herbert Barry and Aylene S. Harper. Feminization of unisex names from 1960 to 1990. *Names*, 41(4):228–238, 1993.
- [3] Greta R. Bauer, Jessica Braimoh, Ayden I. Scheim, and Christoffer Dharma. Transgender-inclusive measures of sex/gender for population surveys: Mixed-methods evaluation and recommendations. *PLOS ONE*, 12(5):1–28, May 2017.
- [4] Christian Bird, Alex Gourley, Premkumar T. Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In Stephan Diehl, Harald C. Gall, and Ahmed E. Hassan (editors), *Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR 2006, Shanghai, China, May 22–23, 2006*, pages 137–143. ACM, 2006.
- [5] Hanjo D. Boekhout, Inge van der Weijden, and Ludo Waltman. Gender differences in scientific careers: A large-scale bibliometric analysis. *CoRR*, abs/2106.12624, 2021.
- [6] Norman M. Bradburn, Seymour Sudman, and Brian Wansink. *Asking Questions: The Definitive Guide to Questionnaire Design – For Market Research, Political Polls, and Social and Health Questionnaires*. Research Methods for the Social Sciences. Jossey-Bass, revised edition, 2004.

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

- [7] Nicolas Brub, Gita Ghiasi, Maxime Sainte-Marie, and Vincent Larivire. Wiki-Gendersort: Automatic gender detection using first names in Wikipedia. March 2020.
- [8] Jill Cao, Kyle Rector, Thomas H. Park, Scott D. Fleming, Margaret M. Burnett, and Susan Wiedenbeck. A debugging perspective on end-user mashup programming. In Christopher D. Hundhausen, Emmanuel Pietriga, Paloma Diaz, and Mary Beth Rosson (editors), *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2010, Legané s-Madrid, Spain, September 21–25, 2010, Proceedings*, pages 149–156. IEEE Computer Society, 2010.
- [9] Abhijnan Chakraborty, Johnnatan Messias, Fabricio Benevenuto, Saptarshi Ghosh, Niloy Ganguly, and Krishna Gummadi. Who makes trends? Understanding demographic biases in crowdsourced recommendations. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):22–31, May 2017.
- [10] Ben Colliver, Adrian Coyle, and Marisa Silvestri. The online “othering” of transgendering and non-binary people. In Karen Lumsden and Emily Harmer (editors), *Online Othering: Exploring the Dark Side of the Web*. Palgrave Macmillan, 2019.
- [11] Edwin Dauber, Aylin Caliskan, Richard E. Harang, Gregory Shearer, Michael J. Weisman, Frederica Free-Nelson, and Rachel Greenstadt. Git blame who? Stylistic authorship attribution of small, incomplete source code fragments. *Proc. Priv. Enhancing Technol.*, 2019(3):389–408, 2019.
- [12] Fred Dervin. *Discourses of Othering*, pages 43–55. Palgrave Macmillan UK, London, 2016.
- [13] Yifan Hu, Changwei Hu, Thanh Tran, Tejaswi Kasturi, Elizabeth Joseph, and Matt Gillingham. What’s in a name? Gender classification of names with character based machine learning models. *Data Mining and Knowledge Discovery*, 35(4):1537–1563, July 2021.

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

- [14] Nasif Imtiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina R. Bai, and Emerson R. Murphy-Hill. Investigating the effects of gender bias on GitHub. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (editors), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25–31, 2019*, pages 700–711. IEEE/ACM, 2019.
- [15] Aylin Caliskan Islam, Richard E. Harang, Andrew Liu, Arvind Narayanan, Clare R. Voss, Fabian Yamaguchi, and Rachel Greenstadt. De-anonymizing programmers via code stylometry. In Jaeyeon Jung and Thorsten Holz (editors), *24th USENIX Security Symposium, USENIX Security 15, Washington, DC, USA, August 12–14, 2015*, pages 255–270. USENIX Association, 2015.
- [16] Samantha Jaroszewski, Danielle Lottridge, Oliver L. Haimson, and Katie Quehl. "Genderfluid" or "attack helicopter": Responsible HCI research practice with non-binary gender variation in online communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 115. Association for Computing Machinery, New York, NY, USA, 2018.
- [17] Joy L. Johnson and Robin Repta. Sex and gender: Beyond the binaries. In John L. Oliffe and Lorraine Greaves (editors), *Designing and Conducting Gender, Sex, & Health Research*, pages 17–38. SAGE Publications, Inc., Thousand Oaks, July 2012.
- [18] Soon-gyo Jung, Jisun An, Haewoon Kwak, Joni Salminen, and Bernard Jansen. Assessing the accuracy of four popular face recognition tools for inferring gender, age, and race. *Proceedings of the International AAAI Conference on Web and Social Media*, 12(1), June 2018.
- [19] Os Keyes. The misgendering machines: Trans/HCI implications of automatic gender recognition. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW), November 2018.
- [20] Athanasios Kokkos and Theodoros Tzouramanis. A robust gender inference model for online social networks and its application to LinkedIn and Twitter. *First Monday*, 19(9), August 2014.

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

- [21] Stefan Krüger and Ben Hermann. Can an online service predict gender?: on the state-of-the-art in gender identification from texts. In Ivica Crnkovic, Karina Kohl Silveira, and Sara Sprenkle (editors), *Proceedings of the 2nd International Workshop on Gender Equality in Software Engineering, GE@ICSE 2019, Montreal, QC, Canada, May 27, 2019*, pages 13–16. IEEE/ACM, 2019.
- [22] Bin Lin and Alexander Serebrenik. Recognizing gender of stack overflow users. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR ’16, pages 425–429. Association for Computing Machinery, New York, NY, USA, 2016.
- [23] Sarah Jill Mah, Mallika Makkar, Kathy Huang, Tharani Anpalagan, Clare J. Reade, and Julie My Van Nguyen. Gender imbalance in gynecologic oncology authorship and impact of COVID-19 pandemic. *International Journal of Gynecologic Cancer*, 32(5):583–589, 2022.
- [24] Ashley Mardell. *The ABC’s of LGBT+*. Mango Media Incorporated, 2016.
- [25] Antoine Mazières, Telmo Menezes, and Camille Roth. Computational appraisal of gender representativeness in popular movies. *Humanities and Social Sciences Communications*, 8(1):137, June 2021.
- [26] Chan Tov McNamarah. Misgendering. *California Law Review*, 109(6), December 2021.
- [27] David Arroyo Menéndez, Jesús M. González-Barahona, and Gregorio Robles. Damegender: Writing and comparing gender detection tools. In *SATToSE*, 2020.
- [28] Jrg Michael. 40000 namen, anredebestimmung anhand des vornamens. *c’t*, (17):182–183, 2007.
- [29] Fariha Naz and Jacqueline E. Rice. Sociolinguistics and programming. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM 2015, Victoria, BC, Canada, August 24–26, 2015*, pages 74–79. IEEE, 2015.

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

- [30] Ehsan Noei and Kelly Lyons. A study of gender in user reviews on the google play store. *Empirical Software Engineering*, 27(2):34, December 2021.
- [31] Richard Van Noorden. The ethical questions that haunt facial-recognition research. *Nature*, 2020.
- [32] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. Going farther together: The impact of social capital on sustained participation in open source. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 688–699, 2019.
- [33] Savannah R. Roberts, Phillipa Hay, Kay Bussey, Nora Trompeter, Alexandra Lonergan, and Deborah Mitchison. Associations among relationship status, gender, and sexual attraction in Australian adolescents’ eating pathology. *International Journal of Eating Disorders*, n/a(n/a).
- [34] Gema Rodríguez-Pérez, Reza Nadri, and Meiyappan Nagappan. Perceived diversity in software engineering: a systematic literature review. *Empir. Softw. Eng.*, 26(5):102, 2021.
- [35] Davide Rossi and Stefano Zacchiroli. Worldwide gender differences in public code contributions and how they have been affected by the COVID-19 pandemic. In *44th IEEE/ACM International Conference on Software Engineering: Software Engineering in Society ICSE (SEIS) 2022, Pittsburgh, PA, USA, May 22–24, 2022*, pages 172–183. IEEE, 2022.
- [36] Lucia Santamaría and Helena Mihaljević. Comparison and benchmark of name-to-gender inference services. *PeerJ Computer Science*, cs156, 2018.
- [37] Morgan Klaus Scheuerman, Jacob M. Paul, and Jed R. Brubaker. How computers see gender: An evaluation of gender classification in commercial facial analysis services. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.

- [38] Paul Sebo. Performance of gender detection tools: a comparative study of name-to-gender inference services. *Journal of the Medical Library Association*, 109(3):414–421, 2021.
- [39] Paul Sebo. How accurate are gender detection tools in predicting the gender for Chinese names? a study with 20,000 given names in Pinyin format. *Journal of the Medical Library Association*, 110(2):205–211, 2022.
- [40] Zohreh Sharafi, Zéphyrin Soh, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Women and men – different but equal: On the impact of identifier style on source code reading. In Dirk Beyer, Arie van Deursen, and Michael W. Godfrey (editors), *IEEE 20th International Conference on Program Comprehension, ICPC 2012, Passau, Germany, June 11–13, 2012*, pages 27–36. IEEE Computer Society, 2012.
- [41] Edward K. Smith, Robert T. Loftin, Emerson R. Murphy-Hill, Christian Bird, and Thomas Zimmermann. Improving developer participation rates in surveys. In *6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013, San Francisco, CA, USA, May 25, 2013*, pages 89–92. IEEE Computer Society, 2013.
- [42] Juan Soler Company and Leo Wanner. On the role of syntactic dependencies and discourse relations for author and gender identification. *Pattern Recognit. Lett.*, 105:87–95, 2018.
- [43] Katta Spiel, Oliver L. Haimson, and Danielle Lottridge. How to do better with gender on surveys: A guide for HCI researchers. *Interactions*, 26(4):6265, June 2019.
- [44] Chad M. Topaz, Bernhard Klingenberg, Daniel Turek, Brianna Heggeseth, Pamela E. Harris, Julie C. Blackwood, C. Ondine Chavoya, Steven Nelson, and Kevin M. Murphy. Diversity of artists in major US museums. *PLOS ONE*, 14(3):1–15, 3 2019.
- [45] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study. *Interacting with Computers*, 26(5):488–511, 2014.

CHAPTER 28 HOW TO ASK ABOUT GENDER IDENTITY OF SOFTWARE ENGINEERS AND “GUESS” IT FROM THE ARCHIVAL DATA

- [46] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Perceptions of diversity on git hub: A user survey. In Andrew Begel, Rafael Prikladnicki, Yvonne Dittrich, Cleidson R. B. de Souza, Anita Sarma, and Sandeep Athavale (editors), *8th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015, Florence, Italy, May 18, 2015*, pages 50–56. IEEE Computer Society, 2015.
- [47] Jock Young. *The Vertigo of Late Modernity*. SAGE Publications, 2007.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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 29

Strategies for Reporting and Centering Marginalized Developer Experiences

Denae Ford, Microsoft USA.*

Brittany Johnson, George Mason University USA.

There are many studies that investigate the experiences of marginalized software developers; however, they tend to include gaps in how people from a marginalized background compare against a majority group. Many researchers focus on where historically marginalized groups do not measure up, missing the opportunity to understand where participants are in fact excelling. Likewise, asking research questions that only seek to surface deficits rather than successes can unintentionally create an unmatched and negative precedence that participants do not hold about their experiences.

In an effort to set a new precedent for conducting and reporting research, we build on asset-based design to propose what we call abundance-based reporting. In this chapter, we outline how to investigate, report, and build interventions with historically marginalized communities.

Introduction

The focus on diversity, equity, and inclusion in software engineering is less often getting brushed off as a side project, becoming a core and pivotal part of moving the field of software engineering forward [1]. As this awakening continues, empirical

researchers may be seeking guidance on how to appropriately investigate and to continue empowering developers in the margins. While many researchers are realizing the value in doing this, they are struggling with the practical approaches and ethical considerations they should take to do this. At the same time, there are also researchers who are trying to determine why it is important to center historically marginalized perspectives. The briefest answer we can offer to that group of researchers is that centering the margins provides us with great insight on how to successfully prepare for a multidimensional workforce in the future. One common reason some researchers fail to center the experiences of those in the margins is because it may not seem practical for them to do. Another reason is likely because they don't see how it can impact their research long-term. An antithetical point that many researchers with this perspective often fail to recognize is that they are doing their own research a disservice when they exclude marginalized perspectives. There are unique experiences that live in the margins that can help us understand bespoke solutions to better serve the masses.

Throughout this chapter we will give examples and outline strategies for how researchers can overcome the issue of practicality and explain the longitudinal value of engaging deeper with people from historically marginalized communities. These strategies include approaches to initiate contact with organizations that cater to specific marginalized groups, approaches to engage with participants from study recruitment through data collection, how to report and share findings about their experiences, and finally ways to build a sustainable long-term relationship that can be mutually beneficial. Finally as we share these strategies, we also include exemplars of how this has been done successfully. Doing so gives us a model to follow and to adapt to our specific research settings. Many of the examples we present come from fields outside of traditional software engineering research; however, they are just as valuable and applicable to the work being conducted in our research community. That said, we encourage readers to consider this chapter to be a guidebook to researchers. Specifically, we hope this helps them feel encouraged to expand their view of how alternative methodologies can be relevant for their research and feel empowered to apply them in their work.

How to Center Marginalized Perspectives in Studies

Before understanding how to best center historically marginalized perspectives in research studies, let us first define what we mean by *historically marginalized*. We draw on the following description of marginalized groups [12]:

Historically marginalized communities are groups who have been relegated to the lower or peripheral edge of society. Many groups were (and some continue to be) denied full participation in mainstream cultural, social, political, and economic activities. Marginalized communities can include people of color, women, LGBTQ+, low-income individuals, prisoners, the disabled, senior citizens, and many more.

A few of the marginalized groups we will be referring to in this chapter are across physical abilities and race. However, as many empirical software engineering studies have not yet investigated the full range of marginalized developers' perspectives in depth, we should draw on other adjacent contexts and fields of science to glean insights on how to study and report on marginalized developer experiences successfully.

A few approaches some have taken to do this well are as follows:

- **Assets-based design:** Asset-based design is a unique approach that leverages the strengths, existing knowledge, and institutional resources that a group may already have as core, thus building research and tools centered in that [14, 16]. In these settings, these assets are intentionally being used from a non-deficit perspective.
- **Joy-centric and celebratory perspectives:** Joy-centric work strategically takes a non-deficit perspective to working with marginalized groups by focusing research on what the group or community celebrates [5]. This perspective of engaging and conducting research mirrors what a group chooses to celebrate – even if that may not mirror the research framing a researcher's community may choose to highlight. You can consider this approach to be intentionally “in spite of” everything that may be negative and “easier” to build on based upon prior research.

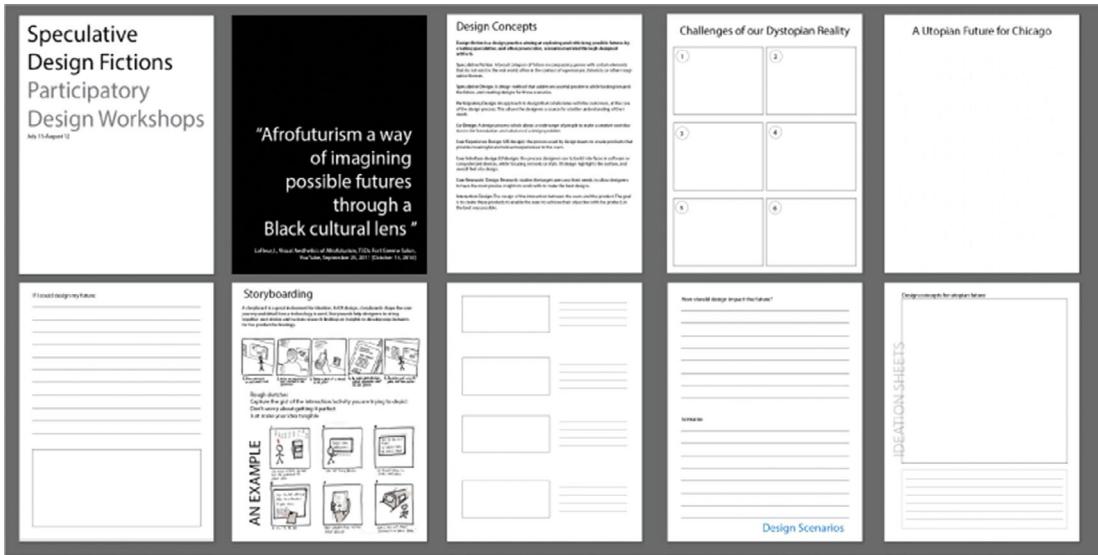


Figure 29-1. Speculative design workbook used in [6]. This can be used as a reference for future studies.

- **Design fiction and speculative design:** Design fiction is a specific approach to speculative design – a helpful research methodology that allows study participants to be critical of a design of a system and creatively imagine a new one [2]. In this approach participants are asked to describe what an ideal system for them could look like. This exercise allows participants to be in the driver seat of designing the tools and often results in supplementary design artifacts [6, 13]. (See Figure 29-1 as a reference.)

From these works, we can learn a lot about what it takes to be successful in highlighting perspectives the way that participants would like to be recognized. We should also note that each of these approaches is rooted in collaborating *with* the community at various stages of the research process. In the next section, we will highlight specific approaches within these paradigms that software engineering researchers can learn from.

Putting Methodologies in Action

In highlighting how researchers have investigated, designed, and built for joy in order to support the marginalized developer and technologist experience, we learn about the variety of formats that have been applied. Now that we have outlined specific approaches on how to center marginalized perspectives, we want to be concrete about approaches through which researchers have applied them. In the next subsections, we highlight two case studies for two different types of marginalized groups: one investigating the experiences of blind and low-vision (BLV) software developers and the other of Black and African American technologists.

Case Study: Investigating Online Communities for Blind Software Developers

To better understand how blind and low-vision (BLV) software developers use online communities as a resource for their needs, Johnson et al. investigated the use of Program-L [9]. Program-L is an online community “for users of Access Technology involved in programming to discuss any technical problems which are related to either the hardware or software they are using.”¹ In this particular study, researchers wanted to uncover the variety of help-seeking behaviors novice developers engaged in when participating in a community specific to two of their identities. One identity is their professional identity as a software developer, and the second is a personal identity as a person who is blind or has low vision. From their analysis of four years of novice behavior, authors were able to build a taxonomy of novice types and recommend design interventions for future demographic-specific online programming communities.

From this study there are several approaches taken that we highlight. One is the approach taken at the beginning of the study to contact the owners of the Program-L mailing lists before conducting the study. In this email (shown in Figure 29-2), the research team took several steps to establish legitimacy before conducting the study. For instance, the research team identified themselves and the work they had done previously. This showed to the community owner they had conducted similar studies in the past that the community owner could check. Next, the research team outlined their intent with analyzing the community and clarified that it would be conducted under the

¹www.freelists.org/list/program-l/

CHAPTER 29 STRATEGIES FOR REPORTING AND CENTERING MARGINALIZED DEVELOPER EXPERIENCES

guidance of a research review board. Finally, the research team also made themselves available to have a one-on-one conversation with the community owner so that they might answer any questions or address any concerns the community owner might have.

Hi [Listserv Owner Name Redacted],

First, let me introduce myself. I'm [Researcher A and Researcher A's affiliation]. For a number of years I have been working on accessible technology and, in recent years, technology and curricula that would help make K-12 computer science education more accessible.

I recently met [Researcher B and Researcher B's affiliation]. [They have] done interesting research about understanding and improving question/answer sites for developers such as Stack Overflow. Because of our overlapping interests, I mentioned to [them] the wonderful listserv Program-L that supports question/answer interaction for blind developers. Understanding the difficulties and needs of blind developers is very important to us. What we learn may also help improve the interaction to better satisfy users' expectations.

For this reason, we would like to know if we could obtain access to the Program-L email archive in order to explore the interactions there. Any study that would be done would be Institutional Research Board (IRB) approved, meaning it would follow the guidelines for an ethical human study. All data obtained from the archive would be anonymized so that individual users could not be identified.

If needed, I would be happy to talk with you over the phone to answer any questions you might have. I'm now at home, away from the office because of the Coronavirus pandemic. I can be reached at [redacted] or I can call you whichever you prefer. It would be best to schedule a time for the call because of the difference in time zones.

Sincerely,

[Name Redacted]

[Email Signature Redacted]

Figure 29-2. *Initial authorization email used in [9]*

There are several approaches to conducting research that center marginalized developers, for instance, asking permission before conducting studies, even if it may be a resource that a member of the research team may already belong to. It may not always be clear whom the permission should be requested from – in that case we would encourage researchers to reach out to members of the organization to figure out who the authorized leader in the community may be. Likewise, engaging in conversations with leaders and sharing the research findings back to the organization is one way to make sure the work conducted has the impacts intended.

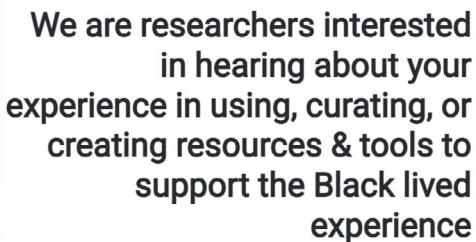
Case Study: Technology for Black Lives Project

As another example of how researchers have been able to practically center historically marginalized experiences, we reference the ongoing Technology for Black Lives study [3]. In this project the research team is investigating how Black software developers and technologists are using, curating, and creating resources and tools in support of the Black lived experience. Authors draw on the archival narrative styles of scholars such as McIlwain [11] to investigate what technology's role is in supporting the Black lived experience and what it means to design for it.

Although this study is not yet completed, the researchers have been publicly vocal about their study – publishing articles about the work as they collect data [7]. The research team has created a website (as shown in Figure 29-3) to announce the study. On this website, the research team publicly shared a variety of information pertaining to the study. Materials available on the public website include the recruitment flyer describing eligibility as well as compensation, who members of the research team are, a brief description of the projects, the intended project goals, how to contact the research team, and supplementary readings and resources.

Technologists for Black Lives

[Home](#) [The Team](#) [About](#) [Project Goals](#) [Contact](#) [Resources](#)



We are researchers interested
in hearing about your
experience in using, curating, or
creating resources & tools to
support the Black lived
experience

Eligibility:

User, builder or curator of resource to support
the Black lived experience

Time commitment:

45 minute interview

Compensation:

\$40 Amazon Gift Card

Conducted by:

Christina Harrington

Figure 29-3. Website for the Technology for Black Lives project. The website includes project descriptions, eligibility, time commitment, compensation, and principal investigators. There are also links to find out more about the research team, project goals, contact information, and supplementary resources

From this project, there are several successful approaches to highlight. One is the fact that the research team created a website to share during the recruitment phase of the project. Having a public presence attached to the work can help increase trust from participants about how the study is being conducted, accountability on the researchers' part, and transparency for potential participants and external researchers interested in tracking the work. Another attribute of this project to highlight is how researchers have used snowball sampling to recruit participants. Working with historically marginalized populations with a specific skill often requires researchers conducting studies with a network of participants that are likely to engage in some overlapping communities. In these settings, respectfully engaging with the network requires care. Researchers from this study used snowball sampling to recruit participants when applicable, which often resulted in participants referring to their personal network of technical colleagues - which is of great importance to them. Although the research team makes several resources public, they do not list all participants that have participated in the

study without their consent. This type of intentional sharing has been helpful in making sure that the researchers have kept participants protected and that the research team continues to respectfully conduct their research.

The Dos and Don'ts of Reporting Marginalized Experiences

Recruiting Historically Marginalized Populations

Critical to the ability of conducting research that recognizes and considers marginalized populations is the ability to reasonably recruit research participants from these historically marginalized groups. In many cases, researchers may not already have relationships in these communities (and that's okay). But how does one initiate a collaboration to form research relationships with a specific group?

DO *make explicit effort to recruit participants from historically marginalized populations by tapping into your existing networks and relationships.*

In this case, you want to focus on individuals and organizations with connections to the target audience or population. The goal here is to build trust through a trusted connection – individuals from historically marginalized communities may be more willing to engage in research if they trust the person conducting or connecting them to the research [4].

DON'T *bombard individuals or organizations with requests to participate in your research.*

There are many reasons you may not be getting responses when initially reaching out. If you're contacting someone in industry, their lack of response may be about timing and their finding the opportunity to think about the opportunity and/or respond. If you're contacting individuals or organizations that are community-centric (e.g., do work that services the community, not a company), their time is not only valuable but likely overloaded with existing commitments.

WHY? By tapping into existing trusted mutual networks and connections, you are building trust by association, which increases the likelihood for response, engagement, and building meaningful (and sustainable) relationships [4]. This essentially gives you a “foot in the door” to begin to build your own network. When reaching out, impatience and constant attempts to make contact can be off-putting and deter the response they originally intended to send (or further interactions in general).

Collecting Data from Marginalized Populations

Once you've acquired connections with your target audience for recruitment, the next obvious step is planning for and then conducting the data collection process. So the next question is, how do we build and maintain trust so that we can collect rich data that accurately reflects participants' lived experiences?

DO *be transparent about consent for participation and how the data will be collected and used.*

When engaging any group in research, but especially historically marginalized populations, one should always begin and proceed with a mutual understanding of the procedures involved [4]. This includes being explicit about what the group's participation entails, what data will and will not be collected, and how the data will be used in the near and distant future.

DON'T *change research plans or directions without informing participants.*

Research is fluid in that plans for collection or analysis may shift. Likewise, as research plans change and data be used in alternative ways than previously outlined, researchers should notify participants from which you have or plan to collect data and receive an updated consent.

WHY? The goal of any research collaboration should be to build trust for sustainable collaborations and outcomes. When the research is not conducted in a transparent and inclusive manner, this runs the risk of breaking trust [4] and has potential ethical implications [Chapter 9, "The Role of Ethics in Engineering Fair AI (and Beyond)"]. Even when not collecting data directly (e.g., via interviews), transparency is key to ensuring those whose data you are collecting and analyzing do not feel manipulated and used, rather than seen and heard [10].

Reporting Insights from Marginalized Populations

After data collection and analysis are complete, researchers are tasked with the sometimes daunting task of consolidating and reporting their findings. While it is always important to be mindful of how we report research findings, it is especially important to take care when reporting insights from historically marginalized groups. So the next question is, how do we accurately and respectfully report insights from these populations?

DO *be explicit and factual about the demographics of the populations in your sample.*

We should only report insights that clearly link to the questions we've asked or the data we've collected. It is also okay to make broad classifications or assumptions based on the given demographics. For example, if a participant states they are from Nigeria, we can assume they may belong to a broader African culture.

DON'T *make narrow assumptions or claims that you did not collect data to support.*

If a clear, and fact-based, connection cannot be made between insights and the data collected, it is likely that the assumption is too narrow. For example, while one can reasonably assume broader culture from geographic location, researchers cannot assume factors such as socioeconomic status from this information alone.

WHY? The assumptions researchers make may not always be true. In fact, it may be reinforcing stereotypes of a group and propagating false information. Furthermore, when done incorrectly, this can build false foundations for future research efforts.

Stating Your Positionality

In conducting studies on historically marginalized populations, the research team may or may not identify with the same background as participants. Either way, it is helpful for the audience consuming that work to be aware of the context the researchers implicitly bring to their analysis of the data.

DO *acknowledge your positionality via a statement that clarifies your background in relationship to the specific demographic the research team is studying.*

This can be done in a “Positionality Statement” or “Researcher Self-Disclosure Statement” in the Methodology or Introduction of a paper (See “Researcher Positionality Statement” in [8] for reference).

DON'T *impose your positionality or inject your experiences upon participants while conducting the study or analyzing the data.*

Researchers can resolve this by taking additional data validation steps. This could include having another researcher analyze data who is familiar with conducting studies where they have a different background from participants and comparing findings.

WHY? Researchers, especially when conducting qualitative analyses, are trained to identify various confounding factors (e.g., background of participants) to better understand how to analyze and report their findings. Likewise, it is also important for researchers to consider the effects their own identity and experiences might have on

that process. This becomes very insightful when researchers choose to propagate and replicate the work of others.

Sustaining Relationships with Communities

Once relationships with historically marginalized groups or the organizations that serve them are established, it is critical to find ways to sustain them. Likewise, it's important that this is done in a way that helps manage both the researchers' and participants' energy wisely.

DO *provide multiple suggestions to participants and community organizations on the variety of ways to remain connected.*

In doing so, researchers should make sure they are considering the size and goals of the participants or organization they are working with. For instance, every participant may not be interested in being featured at developer-specific conferences² such as Strange Loop³ or NeverWorkInTheory; there may be more of an interest in events like AfroTech,⁴ Grace Hopper,⁵ or even alternative forms of dissemination such as museum exhibitions that they would rather be featured in. In summary, researchers should be open to what may be most meaningful to participants and be sure to welcome their perspectives.

DON'T *hide findings or final reporting from participants.*

They should be aware of how their data was used as well as insights gathered from their experiences shared. It is understood that there may be a delay in responding and sharing findings back to participants, but this should be done in a reasonable time frame.

WHY? It is important to find strategies that make the relationship mutually beneficial. This helps the relationship flourish beyond the initial timeline of the project. Continuing the relationship should also be more than just having them participate in future research studies, but also supporting their initiatives that may be outside of traditional research conferences (e.g., community-based workshops) [15]. Keeping this in mind will help build a longitudinal relationship that can reinforce trust for their connections with the broader research community as well.

²<https://dev.events/>

³<https://thestrangeloop.com/>

⁴<https://afrotech.com/>

⁵<https://ghc.anitab.org/>

Conclusion

In closing, we encourage readers to continue to build on previous studies investigating approaches to supporting marginalized developer experiences. This especially includes conducting studies about how communities have been able to empower themselves. We hope that this work can serve as a guidebook for those understanding the best way to empower historically marginalized software developers and simultaneously contribute to the empirical fields of human aspects of software engineering.

Key Takeaways

- Do be mindful of how to engage with, conduct research with, and report on marginalized experiences.
- Do use case studies as a template for success and to avoid pitfalls in the future.
- When centering marginalized experiences, try to intentionally consider approaches that center the assets of a community, what attributes they celebrate, and how to respectfully consider their perspectives.

Bibliography

- [1] Khaled Albusays, Pernille Bjorn, Laura Dabbish, Denae Ford, Emerson Murphy-Hill, Alexander Serebrenik, and Margaret-Anne Storey. The diversity crisis in software development. *IEEE Software*, 38(2):19–25, 2021.
- [2] Mark Blythe. Research through design fiction: narrative in real and imaginary abstracts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’14, pages 703–712, 2014.
- [3] Lisa Egede, Leslie Coney, Brittany Johnson, Christina Harrington, and Denae Ford. Technologists for black lives project website. 2022. Retrieved October 6, 2022, from <https://go.gmu.edu/Tech4BlackLives>.

- [4] Sheba George, Nelida Duran, and Keith Norris. A systematic review of barriers and facilitators to minority research participation among African Americans, Latinos, Asian Americans, and Pacific Islanders. *American Journal of Public Health*, 104(2):e16–e31, 2014.
- [5] Andrea Grimes and Richard Harper. Celebratory technology: new directions for food research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 467–476, 2008.
- [6] Christina Harrington and Tawanna R. Dillahunt. Eliciting tech futures among black young adults: A case study of remote speculative co-design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.
- [7] Christina N. Harrington, Brittany Johnson, Denae Ford, and Angela DR Smith. Designing for the black experience. *Interactions*, 28(5):22–27, 2021.
- [8] Catherine Hu, Christopher Perdriau, Christopher Mendez, Caroline Gao, Abrar Fallatah, and Margaret Burnett. Toward a socioeconomic-aware HCI: Five facets. Preprint at *arXiv:2108.13477*, 2021.
- [9] Jazette Johnson, Andrew Begel, Richard Ladner, and Denae Ford. Program-L: Online help seeking behaviors by blind and low vision programmers. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–6, 2022.
- [10] Shamika Klassen. Black twitter is gold: Why this online community is worthy of study and how to do so respectfully. *Interactions*, 29(1):9698, January 2022.
- [11] Charlton D. McIlwain. *Black Software: The Internet and Racial Justice, from the AfroNet to Black Lives Matter*. Oxford University Press, USA, 2019.

- [12] Oregon State Historic Preservation Office. Researching historically marginalized communities. April 2018. Retrieved October 6, 2022, from www.oregon.gov/oprd/0H/Documents/HB34_Researching_Historically_Marganized_Communities.pdf.
- [13] Mairieli Wessel, Ahmad Abdellatif, Igor Wiese, Tayana Conte, Emad Shihab, Marco A. Gerosa, and Igor Steinmacher. Bots for pull requests: The good, the bad, and the promising. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 274–286. Association for Computing Machinery, New York, NY, USA, 2022.
- [14] Marisol Wong-Villacres, Carl DiSalvo, Neha Kumar, and Betsy DiSalvo. Culture in action: Unpacking capacities to inform assets-based design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 114. Association for Computing Machinery, New York, NY, USA, 2020.
- [15] Marisol Wong-Villacres, Sheena Erete, Aakash Gautam, Azra Ismail, Neha Kumar, Lucy Pei, Wendy Roldan, Veronica Ahumada-Newhart, Karla Badillo-Urquiola, J. Maya Hernandez, et al. Elevating strengths and capacities: the different shades of assets-based design in HCI. *Interactions*, 29(5):28–33, 2022.
- [16] Marisol Wong-Villacres, Aakash Gautam, Wendy Roldan, Lucy Pei, Jessa Dickinson, Azra Ismail, Betsy DiSalvo, Neha Kumar, Tammy Clegg, Sheena Erete, Emily Roden, Nithya Sambasivan, and Jason Yip. From needs to strengths: Operationalizing an assets-based design of technology. In *Conference Companion Publication of the 2020 on Computer Supported Cooperative Work and Social Computing*, pages 527–535, 2020.



Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, 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 changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter'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.

Index

A

- Access and social justice
armed conflict, 367
education, 368
GSE, 368–370
MENA, 367, 368
strategies to expand access, 369, 370
The United Nations (UN) High
Commissioner for Refugees
estimates, 367
- Accessibility, 128, 130, 131, 153
- Accessibility breakdowns, 130
- Accessibility modifiers, 128
- Accessible Rich Internet Applications
(ARIA), 129
- Accommodations for ND students
adapting the style and content, 420
- ADHD, 418
- ASD, 417
- clarity and communication, 419
- discrete mathematics, 419
- dyslexia, 418
- economical changes, 419
- feedback, 423
- free-text answers, 421, 422
- friendly course material, 421, 422
- guidelines, 417
- implemented, 419
- intervention, 418–420
- intervention experiences, 420–423
- and neurodiversity, 415, 416
- and NT, 421
- OpenDyslexic, 419
- our style-and presentation-based
approach, 423
- participants to rate, 421
- re-reading assignments and slides, 420
- surveys, 420
- syllabi, 423
- text-to-speech tools, 423
- Actionable, 136, 141, 316, 395, 469, 470
- Adaptive software, 28
- Aequitas, 145
- Aforementioned techniques, 493, 498
- AfroTech, 518
- Age bias, 18, 22
- Aggie Engineering Ambassadors, 391, 393
- Agile, 114, 155, 175, 263, 456
- AI-based tools, 115
- AI Fairness 360 (AIF360), 144, 145
- All In for Maintainers, 394, 395
- All In for Students, 392–394
- Amazon's recruiting tool, 92
- Android application, 128
- Anonymous author code review (AACR)
author, change request, 328
author identity, 325
authors' time zones, 333
chain/stack, changelists, 329
code change's author, 324
code review tool, 324, 329
deploy anonymization

INDEX

Anonymous author code review

(AACR) (*cont.*)

authors, 326

leadership, 326

reviewers, 326

review types, 327

design perspective, 332

double anonymous, 325

email notifications, 330, 331

experience, 333

hide the author's identity,

reviewers, 328

implementation

Critique Code Review Tool at Google, 328

human authors, 328

review, 327

linked bug report, 332

overview, 325

preference data, 333

remove identity, software engineering

systems, 333

software development tools, 332

task-tracking tool, 328

within the code, 331

Appearance-based gender inference, 233

Apple App Store, 95, 97, 98

Artifact-to-gender tools, 495, 496

Artificial Intelligence (AI), 135, 142,

372, 374

AR/VR tools, 28

Assets-based design, 509

Association of Computing Machinery (ACM), 42, 139, 368, 374

Attention deficit hyperactivity disorder (ADHD), 413, 417, 418, 423

Augmented reality (AR), 23

Augmented reality browser plug-ins, 23

Authorship attribution techniques, 495

Autism spectrum disorder (ASD), 413, 416, 417, 423

Automated techniques, 496, 497

Automatic face-to-gender tools, 496

Automatic gender inference tools, 233, 234, 238

Automatic gender recognition, 497

B

Biases, 5–7, 9, 63, 145, 197, 232, 270, 324, 386

Binary construct, 400

Black Lives Project, 513–515

Blind and low-vision (BLV) software developers, 511–514

Blog post, 310, 495

Brogrammer culture, 177

Brogrammer stereotype, 279, 287

C

Campus spaces (CS)

general perspectives, 405, 406

restrooms, 406, 407

specific spaces, 406

virtual spaces, 407

Career transition, 258, 263–264, 266

Challenged end users, 24, 112

Changelists (CLs), 323, 326–329

Chromium, 237, 313

Cisgender, 174, 175, 276–278, 296, 404

Code author's profile, 324

Code reviews, 57, 123, 212, 309, 416

Codes of conduct (CoCs), OSS, 62, 64

advocacy, 296

cisgender women, 296

- community consensus, 303
- community conversations
- adoption and creation, 298
 - moderation and enforcement, 299, 300
- contributor experiences, 301–303
- Contributor’s Covenant, 295
- custom CoC, 303
- GitHub, 303
- governance document,
- moderation, 303
- moderators, 303
- moderator’s contact info, 303
- open research questions, 304, 305
- signal, perceived project, 304
- structures, 296–298
- support, 299
- Code writing preferences, 131
- Cognitive disabilities, 91–97
- Cognitive diversity, 39, 349
- Cognitive features, 9
- Collaborative development activities, 212
- question, 213
- Collective intelligence, 278
- Community-engaged learning (CEL), 454
- Community partners’ collaboration, 459, 460
- Community problems, 152, 159, 455–457
- Competence-confidence gap, 60, 279, 408
- Consequentialism, 139, 140
- Contributor’s Covenant (CC), 295–297
- Cost prediction, 137
- COVID-19 apps, 21
- COVID-19 pandemic, 25, 63, 193, 196, 405, 416
- COVID-19 pandemic lockdowns, 22
- CRAN, 241, 242
- Critique, 57, 139, 324, 334
- Critique AACR, 324
- Crowd-based elicitation, 94–96, 100
- Crowd-sourcing software
- requirements, 98
- CSD Gender Taskforce, 400, 402, 407
- CS education
- and career, 429
 - categories, 432
 - confidence, 433
 - early access, 433
 - effective interventions to engage girls in
 - frustration of not feeling valued, 442, 443
 - gender-related stereotypes, 435
 - missing access, 440
 - missing confidence, 441, 442
 - missing sense of belonging, 441
 - to promote diversity, 435–439
 - feeling valued, 433
 - frustrations steering girls, 430–434
 - prevalent, 433
 - sense of belonging, 433
 - stereotypes, 433
- Cultural-historical context, 187, 190
- Czechitas, 430
- biases, 269
 - change takes time, 270
 - communication, 270
 - community, 265
 - create and share stories, 266
 - Czechia, 258, 264
 - diversity, 268
 - inclusive environment and encouragement, 266
 - knowledge, 266
 - leadership, 265
 - obstacles and challenges, 267, 268

INDEX

- Czechitas (*cont.*)
- remove barriers, 270
 - sense of belonging, 265
 - support women in tech
 - awareness, 261
 - career transition, 263
 - community, 264
 - participation, 263
 - pillars, 262
 - training, 262
 - sustainable financial model, 266
 - talents, 269
 - tech professional volunteers, 258
 - Tuckman's Model of Team
 - Dynamics, 269
 - visual and playful communication, 265
 - Women ICT professional, 258
 - Women ICT students, 265
 - women in tech
 - access, 259
 - confidence, 260
 - feeling value, 260
 - sense of belonging, 260
 - stereotypes, 259
- D**
- Data analytics, 263
- Data bias, 71, 77–81, 83, 84
- Data collection process, 516
- Data-driven archival studies, 488
- Data for analysis, 10
- Decision-making, 138
 - software development, 135
- Demographic diversity, 454
- Demography-based measurements, 470
- Deontology, 139, 140
- Design thinking (DT)
- community partner, 155
- empathy, 162
- empathy mapping, 157
- fifth Inspire project, 155
- hopes and fears, 158
- IBM, 155, 156
- inclusive teams, 161
- Inspire program, 156, 162
- journey mapping, 158, 159, 162
- playbacks, 159
- revisit traditional
 - techniques, 159, 160
- team morale, 162
- Developer-specific conferences, 518
- Developer Toolkit, 122
- Development activities, 6, 9, 10, 212, 213, 217
- DevOps, 114, 170, 394
- Digital Academy, 263, 264
- Discrete mathematics, 419
- “Dissecting Racial Bias”, 137
- Diverse community, 403, 405
- Diversity, 4, 11
 - dimensions of, 400
 - in software engineering, 400
 - two solution spaces, 6
- Diversity and inclusion (D&I), 3–6, 56, 151, 190, 245, 338, 339
- Diversity crisis, 178, 186, 187
- Diversity, equity, and inclusion (DEI)
 - open source, 386
- Diversity in software engineering, 8
 - context, 9
 - data and methods, 10
 - objective, 9
 - stakeholders, 9
- Diversity management models, 38
- Dyslexia, 413, 416, 418, 423

E

Ecommerce software, 22
 Education from the Middle East and Africa
 access and social justice, 367–370
 electronic databases, 366
 Google Scholar, 366
 software engineering
 education, 373–375
 software engineering
 policy, 376–378
 software engineering practice,
 366, 370–373
 software engineering research,
 365, 373–375
 ehealth, 27
 Electronic databases, 42, 44, 366
 Elicitation, 91, 93–98
 Emotional reactions, 18, 23, 25
 Empathy, 23, 28, 162, 163
 Empathy-based design thinking
 processes, 161
 Empathy mapping, 157
 End user app review analysis, 21
 End user human-centric aspects, 105
 ages, 106
 emotions, 107
 engagement and
 entertainment, 107
 ethnicity and culture, 106
 gender, 106
 “human-centered” RE and design
 approaches, 114
 human values, 107
 improved tools, 115
 languages, 107
 online survey responses, 109
 participants, 109

physical/mental challenges, 107
 requirements
 engineering, 112–114
 software development, 106
 study design, 107
 qualitative analysis, 108
 recruitment and data
 collection, 108
 survey and interviews, 108
 survey respondents, 110–112
 use standards/guidelines, 114
 Equity, diversity, and inclusion (EDI),
 377, 378
 in engineering programs, 450
 training, 461
 Ethical by design, 141, 142, 146
 Ethical decision-making, 146
 ethical frameworks, 141, 142, 144
 ethics tools, 144, 145
 software development, 141
 Ethical decisions, 135, 136, 141–146
 Ethical design, 142
 Ethics, 11, 135
 consequentialism, 139
 deontology, 139
 practice, 140, 141
 virtue ethics, 140
 Ethics-aware software engineering,
 142, 143
 awareness, 143
 conscious valuing, 143
 ethics knowledge, 143
 transparency, 144
 Ethics case studies
 cost prediction, 137
 “Dissecting Racial Bias”, 137
 future avoidable costs, 137
 outcomes based, label choice, 138

INDEX

- Ethics tools, 144, 145
- Ethnic-racial diversity
- accepted articles, 45
 - black people, 39
 - challenges, 40, 41
 - classifications of indicators, 47
 - cognitive diversity, 39
 - identity diversity, 39
 - identity groups, 39
 - personal biases and stereotypes, 37
 - research field of industry, 47
 - research methodology, 42–44
 - research question (RQ), 41
 - systematic literature
 - mapping, 43–50
- Ethnic-racial education, 40
- Ethnic-racial expression, 39
- Ethnic-racial inclusion, 41
- Evidence-based inclusivity evaluation method, 470
- Exclusion criteria (EC), 42, 43, 340, 341
- Expectation conflicts, 460
- Experiential learning, 450–452, 456, 459, 462
- F**
- Face recognition systems, 4
- Face recognition techniques, 494
- Face-to-gender techniques, 496
- Face-to-gender tools, 494, 495, 499
- Facets, 22, 154, 470–472, 477–479
- Facial recognition methods, 496
- Fairkit-learn, 145
- Fairlearn, 145
- FairML, 144
- Fairness, 5, 7, 8, 144–146, 179, 181
- FairVis, 145
- FAT Forensics, 145
- Fine-grained comparisons, 476–478
- Fine-grained diversity measurements, 470, 475, 476
- Fine-grained method, 470
- Free/libre open source software (FLOSS/OSS), 295
- Frustrations, 260, 266, 429–434, 488
- Funkify, 23
- Future avoidable costs, 137
- G**
- Game jams
- create games, 275
 - gender-inclusive events (*see* Gender-inclusive events)
 - low gender diversity, 279, 280
 - research, gender-related issues, 280, 281
- GDP growth
- Ghana, 371
 - Jordan, 371
- Gender
- API, 234
 - woman-man binary, 277
- Gender bias, 9, 18, 24, 26, 344
- in SE job advertisements, 24
- Gender distributions, 233–238, 240–242, 245, 246
- Gender-diverse environments, 173
- Gender-diverse organizations, 377
- Gender-diverse workplaces, 169, 180
- Gender diversity, 245, 344
- information technology industry, 278
- Gender diversity, software
- development teams
 - characteristics, 180

- companies do not support diversity and inclusion, 177
- intersectionality, 178, 180
- lack of awareness, 181
- leadership and management roles, 175
- meritocracy and elitism, 178
- missing affirmative actions and initiatives, 177
- perceived benefits, 173–175
- perceived difficulties, 175, 179
- sexism and prejudice, men, and professional insecurities, 176
- survey, 169–171, 181
- themes and sub-themes, 171
- unawareness, 179
- women support, 180
- Gender Equality Admissions and Faculty programs, 403
- Gender expression, 277, 281, 407, 488
- Genderfluid, 277
- Gender gap, 55, 429
- Gender-guesser, 234, 494
- Gender identity, 277
 - The ABC's of LGBT+, 490
 - bagel/aggressive reactions, 490
 - data-driven archival studies, 488
 - definitive guide to questionnaire design, 489, 490
 - empirical evaluation, 490
 - Linux Foundation, 489
 - mining software repository data, 491–498
 - NAME's sex, 489
 - othering, 489
 - researchers and practitioners, 488
 - Stack Overflow, 489
 - study participants, 488
 - transfeminine, 490
- transmasculine, 490
- user interface, 488
- Gender inclusion, 278, 360
- Gender inclusive, 26, 70, 71, 276
- Gender-inclusive events
 - competitive/collaborative ambience, 286
 - equipment, 283
 - fun/joy of joining, hackathons/game jams, friends, 284
 - gender-inclusive organizing team, 281
 - healthier habits, 286
 - inclusive language, 282
 - introduce elements, underrepresented genders, 285
 - learning activities, technical and soft skills, 285
 - recommendations, 281
 - safety visible, code of conduct, 283
 - underrepresented genders, 284
- Gender inference, 234, 240, 495
- Gender in software, 70
 - answering research questions
 - data bias, 80
 - for dating software, 77
 - gender-inclusive software, 78
 - for government/tax software, 78
 - inductive themes and codes, 79, 80
 - for medical software, 77
 - non-binary discrimination, 77
 - non-binary people, 76
 - participants, 76
 - personnummer, 78
 - surveys/questionnaires/sign-ups, 81
 - thematic coding, 76
 - user's gender, 78

INDEX

- Gender in software (*cont.*)
- credibility, 85
 - data bias, 71
 - design of systems, 83
 - gendered aesthetics on website, 70, 71
 - gender-inclusive requirements, 70, 71
 - internal validity, 85
 - interview study
 - data analysis, 75
 - data collection, 73, 74
 - inductive coding, 75
 - interviewees' demographic information, 75
 - purpose, 72
 - research questions, 72, 73
 - medical software and data bias, 83
 - non-binary person, 70
 - privacy concerns in e-health
 - applications, 70, 71
 - quality of requirements, 70
 - social goal models, 71
 - software development process, 82
 - user experience (UX) design, 71
 - user interface and design, 83
- GenderMag, 18, 24, 82, 353, 470–473
- GenderMag facet survey
- calculate facet medians, 475
 - cluster analysis and condense, 479
 - cognitive styles, 470
 - complement, 475
 - cross-validation, 478, 479
 - demographic validation, 479
 - designers' actions, 478
 - empirical, 479, 480
 - evaluation method, 476
 - evidence-based inclusivity evaluation method, 470
 - fine-grained comparisons, 476, 477
- fine-grained diversity
- measurements, 475
- fine-grained understanding, 477
- higher-resolution measure, 475
- pre-validation, 478, 479
- reliability, 478, 479
- sum each facet, 475
- survey key, 475, 476
- tag each participant's facet score, 476
- think-aloud study participants, 481
- timeframe, 471
- types and values, 471, 472
- validation, 473, 478–481
- Gender-nonconforming people, 488
- Gender non-conformity, 277, 400
- Gender reductionism, 234
- Gender-related biases
- male-gendered environments, 348
- Gender-related information, 496
- Gender-related initiatives
- campus spaces, 405–407
 - CSD members, 402
 - faculty members, 403
 - focus group, 403
- Gender Equality Admissions and Faculty programs, 403
- gender taskforce, 401, 402
- and inclusion initiatives, 401
- learning environment, 404, 405
- learning environment on campus, 403
- participants, 403
- recommendations
- carry out awareness campaigns
 - within the CSD, 407
 - foster student self-confidence and job readiness, 408
 - rethink CSD courses with a more inclusive lens, 408

- revise the use of campus spaces, 408
 - work toward creating inclusive workplaces in industry, 409
 - work with campus authorities on general issues, 409
 - TAs, 403
 - use of campus spaces, 403
 - Gender-related stereotype, 432, 434–436
 - Gender roles, 63, 345
 - Gender taskforce, 400–402
 - General Data Protection Regulation (GDPR), 85, 488
 - Gerrit's respectful code review reminders, 310
 - context-sensitive suggestions, 316
 - design, 310–312
 - evaluation approach, 312–315
 - feedback dashboard, 317
 - feedback on feedback, 317
 - limitations, 317, 318
 - toxicity, 318
 - toxicity scores, 315, 316
 - Ghana, 365
 - development initiatives, 373
 - GDP growth, 371
 - geographic regions, 365
 - Google search engine, 372
 - populations, 366
 - software engineering practice, 370
 - in West Africa, 365
 - GHTorrent, 238, 240–242
 - GitHub, 216, 219, 233, 239, 279, 323, 386, 387, 393, 494, 496
 - GitHub human-centric issues analysis, 21, 22
 - Glass ceiling, 57–58, 63
 - Global Diversity, 386
 - Global Game Jam (GGJ) 2021, 280, 284
 - Globalization within software development, 376
 - Global software engineering (GSE), 368–370
 - Google, 187, 208, 210, 216, 218
 - Google/Alphabet, 222, 223, 227
 - Google forms, 489
 - Google readability reviews, 327
 - Google Scholar, 340, 341, 366, 372
 - Google search engine, 372
- ## H
- Hackathons
 - gender-inclusive events (*see* Gender-inclusive events)
 - hacker culture, 279
 - hardcore ethos, 279
 - low gender diversity, 279, 280
 - research, gender-related issues, 280, 281
 - software application types, 275
 - Hawthorne effect, 6
 - HCI Guidelines for Gender Equity and Inclusivity, 490
 - Healthcare costs, 137, 138
 - Historically marginalized communities
 - assets-based design, 509
 - Black Lives Project, 513–515
 - BLV software developers, 511–514
 - consolidating and reporting, 516, 517
 - data collection, 508, 516
 - design fiction, 510
 - empowering developers, 508
 - joy-centric and celebratory perspectives, 509
 - positionality, 517, 518

INDEX

Historically marginalized

communities (*cont.*)

recruiting, 515

relationships, 518

in research studies, 508

speculative design, 510

Human aspects

impacting requirements

engineering (RE)

age, 18

culture and ethnicity, 19

emotional reactions, 18

gender bias, 18

geographic location, 19

physical/mental challenges, 18

socio-economic status, 19

impacting software engineering (SE)

diverse end user

requirements, 20

SE activities, 20–22

Human Capital Index (HCI), 371, 372

Human-Computer Interaction (HCI),
486, 497

|

IBM design thinking, 155, 156

Identity diversity, 39

Implicit biases, 55

Impostor syndrome, 57, 63

Inclusion, 11

- two solution spaces, 6

Inclusion criteria, 42, 341

Inclusion in software engineering, 8

- context, 9
- data and methods, 10
- objective, 9
- stakeholders, 9

Inclusive elicitation

- car safety devices, 92
- challenges in traditional elicitation, 93, 94
- demographics, 92
- of user needs, 92

InclusiveMag, 22

Inclusive participatory action research (IPAR), 97

Inclusive requirements elicitation, 91, 98, 100

Inclusive software

- biases, 154
- definition, 153
- DT (*see* Design thinking (DT))
- end user's emotions, 162
- human-centric issues, 153
- modern software developments, 153
- software development cycle, 153
- software products, 154

Inclusivity, 61, 70, 80, 83, 296, 352, 470

INSPIRE

- autonomy, 451
- community partners, 452
- competence, 451
- demographic diversity, 454
- experiential learning, 450
- intensive training, 451
- lessons learned, 457–462
- principles of self-determination theory, 451
- program timeline, 452, 453
- projects and the solutions, 454–456
- recruitment and project selection, 452–454
- relatedness, 451
- research methods, 456, 457
- society-impactful projects, 450

sponsors, 452
 STEM, social impact, 152
 team formation, 454
 undergraduate and graduate
 students, 451
 workshops, 456
I
 Intensive training, 451
 Interdisciplinary approach, 429, 443, 444
 Intersectionality, 11, 12, 178, 180, 186
 Intersectional perspectives, software
 engineering
 dominant narratives
 courageous newcomer's
 story, 194, 195, 197, 198
 family man's story, 192, 193,
 195, 196
 rising star's story, 193, 194, 196–198
 intersectionality, 186
 narratives, 195, 196
 participants' social identities, 191
 research approach
 intersectional
 perspectives, 187–190
 interviews, 187
 narratives, 187, 190
 URGs, 186
 social identities, 186

J

JavaScript, 122, 123, 243, 244
 Job-related affective well-being scale
 (JAWS), 25, 123
 Jordan, 365
 development initiatives, 373
 GDP growth, 371
 geographic regions, 365
 in Middle East, 365

populations, 366
 software engineering practice, 370
**Jordan University of Science and
 Technology (JUST)**, 375
 Journey mapping, 158, 159, 162

K

Knowledge creation, 173
 Knowledge management (KM), 376

L

LadderBot, 97, 98
 Learning environment
 faculty attitudes and
 behaviors, 404
 gender and sexual diversity at
 internships, 405
 gender-diverse perspective in CSD
 courses, 404
 general perspectives about CSD
 courses, 404
 general perspectives about
 internships, 405
 peer attitudes in the classroom, 404
 perceptions about own learning
 abilities, 405

Lessons learned, 449

 community partners' collaboration,
 459, 460
 community problems, 457
 guidance empowers students to
 balance autonomy and motivates,
 458, 459
 mentorship and EDI training, 460–462
 real clients, 457
 training on soft skills, 457, 458

INDEX

LGBTQIA+ community, 278
LinkedIn Fairness Toolkit (LiFT), 145
Linux Foundation, 387, 489
LowCode approaches, 27
Low gender diversity, 232
 game jams, 279, 280
 hackathons, 279, 280

M

Machine learning software, 5, 145
Male-dominated industries, 344, 349
Male-dominated workforce, 24
Male-gendered industries, 348
Male-oriented domain, 435
Male protectionism, 176, 177, 180
Maternity wall, 57, 58
Measurements, 469, 470
Mentoring program,
 61, 194, 341, 352
Mentorship, 337, 338, 354
Mentorship-sponsorship divide, 394
Meritocracy, 41, 50, 178–180
Middle East and Northwest Africa
 (MENA), 367–370
Mining online channels, 94
Mining software repository data
 artifact-to-gender, 495, 496
 face-to-gender, 494, 495
 gender information, 491
 limitations, 496, 497
 name-to-gender, 491–494
 software engineering, 497
 source code/comments, 491
 wealth of data, 491
Mining user opinions online, 94, 95
Misgendering, 497
Multidimensional workforce, 508

Multidisciplinary teams, 26
Myth of racial democracy, 40, 50

N

Name-based gender inference, 233,
 234, 240
Name-to-gender tools, 491–494, 497
Namsor, 234, 236, 237, 240
Natural language process (NLP), 142,
 312, 313
Neurodivergent (ND) students
 accessible workplaces, 416
 accommodations, 414
 ASD, 416
 challenges in specific tasks, 414
 dyslexia, 416
 experiences, 415
 in higher education, 414, 415
 IT industry, 416
 ND individuals, 416
 and NT students, 414
 systematic literature review (SLR), 416
Neurodiversity
 ADHD, 413
 ASD, 413
 awareness of, 415
 dyslexia, 413
 in higher education, 414
 in industry, 415
 realms of medicine and
 psychology, 414
Neurotypical (NT), 160, 161, 414, 416
Neurotypical (NT) students, 414
Neutral language, 160, 282
NeverWorkInTheory, 518
NLP-based tool, 24
Non-binary discrimination, 77, 84

Non-binary trans-people, 277
 Non-heterosexual, 400, 402
 Non-inclusive software, 154
 Non-stereotypical talent spectrum, 260

O

Online communities
 BLV software developers, 511–514
 Online crowd-based elicitation, 94
 Online ecommerce, 22
 Online feedback, 94–100
 Onshoring GSE, 376, 377
 OpenDyslexic, 419
 Open source DEI
 All In, 386
 All In for Maintainers, 394, 395
 All In for Students, 392–394
 better understanding of strengths,
 limitations, and challenges, 388
 challenges and benefits, 386
 decision-making, 387
 diversity problem, 386
 GitHub, 386, 387
 implementation, 392
 intervention should at community
 level, 388, 389
 invest in our ecosystem, 390, 391
 Linux Foundation, 387
 maintainers need help, 389, 390
 more inclusive, 388
 opportunities for everyone in chain, 391
 principles, 385
 research-based findings, 392
 social sector organizations, 386
 Open source projects, 63, 216, 245, 285,
 386, 487
 Open source software (OSS)

 economic value, 232
 gender representation, 232
 low gender diversity, 232
 Open source software sustainability, 232
 OpenStack, 64
 Organizational decision-making, 211

P, Q

Pair programming, 9, 123, 130, 170
 Peer parity, 56, 63
 Person's ethnicity, 37
 Perspective API, 312, 313, 315
 Physical/cognitive disabilities, 95
 PlatformIO, 241, 242, 245
 Playbacks, 159
 Population
 software engineering practice, 371
 Professional insecurities, 176, 177
 Program-L, 511
 Project Breakthrough, 393
 Public and governmental authorities, 378
 Pull request, 47, 323, 324, 344

R

Racism, 40–42, 48–50, 178, 190
 Recruitment methods, 25
 Repository mining, 488, 491, 496
 Requirements elicitation, 91, 93,
 95, 96, 100
 Restrooms, 71, 403, 406, 407, 409

S

Scholarship, 337, 339, 393
 Science, Technology, Engineering, and
 Math (STEM), 177, 400, 450

INDEX

- Search databases, 42, 43
Search string, 42, 340
Self-determination theory, 449
 principles of, 451
Self-efficacy, 198, 266, 348, 353, 475, 476
Self-identify, 208, 220, 221, 444, 490
Sexism and prejudice, 176, 180
Sexual orientation, 39, 170, 278, 400,
 405, 450
Smart analysis tools, 97
Smart home technology, 18
Social identities, 178, 186–188, 191,
 195, 196
Social media sites, 498
Social sector organizations, 386
Social vulnerability, 180, 181
Society-impactful problems, 450, 451
Society-impactful projects, 450
Socio-technical grounded theory
 (STGT), 26
Soft skills, 194, 281, 285, 442, 457, 458
Software data, 6
Software developer inclusion
 code review process, 217
 collaborative development
 activities, 217
 design research process, internal
 tools, 217
 documentation, engineering tools, 212
 experiences, collaborative
 development activities, 212,
 213, 217
 inequities, code review, 208
 level of comfort reaching out off-team,
 214, 215
 open-ended question, 210
 open information-sharing, 211
 organizational decision-making, 211
product design processes, 211
race/ethnicity categories, 208
refinement and analysis, 222
sample, 220, 221
sampling approach, 208
Stack Overflow *vs.* GitHub, 216, 218
survey, 208, 209
survey design, 219
survey frontmatter, 221
survey questions, 222–227
transparency, 211
Software development, 3, 5–9, 92, 125,
 140–142, 276, 332
Software engineering (SE), 169, 497
 human aspects, 17
 inspire other fields, 11
 in South America, 399
 unique opportunity, 5, 6
Software Engineering Body of Knowledge
 (SWEBOK), 374
Software engineering education and
 research
 in African Nations, 374
 EDI, 373
 in the Middle East, 374, 375
 UNU-CS, 373
 UNU-IIST, 373
Software engineering policy
 EDI, 377, 378
 onshoring GSE, 376, 377
 policy makers and visionaries, 376
 regions, 376
Software engineering practice, 366
 demographics, 370, 371
 GDP, 371
 in Ghana, 370
 global footprint, 372, 373
 Human Capital, 372

in Jordan, 370
 population, 371
 Software engineering (SE) process, 106
 Software-intensive products, 365, 378
 Software product, 91–95, 154, 157
 Software systems, 7
 for fairness, 8
 Stack Overflow, 27, 216, 219,
 350, 494–496
 STEM university, 375
 Stereotypes, 10, 40, 50, 173–175, 190,
 259, 433
 Stereotypical talent spectrum, 260
 Strange Loop, 518
 Study's population, 431
 Sustainable solutions, 11
 Systematic literature mapping, 43
 Systematic literature review (SLR), 340,
 400, 416
 Systematic mapping, 38, 41, 42, 339, 354

T

Teaching assistants (TAs), 403, 404,
 407, 450
 Text-based gender inference, 233
 The Federated Africa and Middle East
 Conference on Software
 Engineering (FAMECSE), 375
 Thematic analysis, 108, 153, 171, 457
 Themis, 145
 The United Nations University in Macau
 created the International Institute
 for Software Technology (UNU-
 IIST), 373
 The United Nations University Institute on
 Computing and Society
 (UNU-CS), 373

Toxicity, 310, 312–318
 Toxicity API, 313, 314
 Traditional elicitation, 91, 93–94,
 96, 97, 159
 Trailing spouse, 63
 Transgender, 178, 208, 276–278, 401, 490

U

UI developer job roles, 121
 UI development, 123, 130
 and collaboration, 124
 and sighted developers, 126–129
 in software development teams, 125
 Under-represented end user human
 aspects, 27
 United Nations Development
 Programme's Gender Equality
 Seal, 400
 User-centered design (UCD), 93,
 96, 97, 100
 User interface (UI) development, 121
 Utilitarianism, 139

V

Validated, 469, 471, 478–481
 Virtual spaces, 407
 Virtue ethics, 139, 140
 Visual artifacts, 125
 Visual impairments
 code readability and navigation, 127
 design and development roles, 126
 existing projects, 128
 mouse interactions, 126
 programming-related resources, 123
 roles of programmers, 130
 sharing their code contributions, 129

INDEX

Visual impairments (*cont.*)

- UI development for mobile platforms, 122
- UI development jobs, 130
- UI frameworks and libraries, 123

W, X, Y, Z

Web development, 245, 258, 262, 263, 419

Woman-man binary, 277

Women in computing (WiC), 198, 400, 401, 433

Women in open source infrastructure

- automatic gender inference tools, 233, 234

future studies, 246

gender distribution

- core contributors, 243

- different ecosystems, 240–242

- prior studies, ecosystems, 238

methods

- data processing pipeline, 238, 239

- gender inference, 240

prior studies, gender distribution

- data sources and methods, 235

- mining software repositories, 238

- survey, 234

women's participation, package

- managers, 243

Women in tech

actions to mitigate challenges

- baseline actions, 60

- cheer those who are centered, 62

- code of conduct, 62

- de-stereotype the tech, 61

- empower, 60

- leadership roles, 59

- as mentors for other women, 61

promote awareness, 59

promote inclusive language, 61

recruitment process, 60

showcasing, 60

supporting parenthood, 62

women-only groups and

activities, 61

types of challenges, 56

- glass ceiling, 57

- impostor syndrome, 57

- lack of peer parity, 56

- maternity wall, 57

- prove-it-again, 57

- toxic culture, 56

- work-life balance, 58

Women mentorship, OSS projects

- approaches, 353

- barriers, 342

- challenges, 343

- formal mentorship programs, 339

- formal mentorship, support women

- need, 350

- gender bias, 338

- gender-specific challenges, 339

- inclusion criteria, 341

- informal mentorship, 339

- lack of confidence, 338

- literature, 354

- mentoring programs, 341

- mentorship practices *vs.* D&I, 338

- methodology, 339–341

- onboarding and retention, open

- source communities, 338

- other disciplinary contexts, 342

- professional contexts, 341

- professional/OSS contexts challenges

- adapt to male standard, behavior,

- 345, 346

- competing priorities in the home, 344, 345
inappropriate behavior, mentors/
perceptions of impropriety, 346, 347
lack of women in top positions, 349
less likely to be included/less likely
to receive mentorship, 347, 348
male-gendered environments
disadvantage women,
mentorship, 348
mentors reluctant to mentor
women, 347
seen and/or see themselves as less
capable, 344
SE literature, 353
Stack Overflow community, 339
strategies, women mentees in OSS
projects
connect women with online
support communities, 353
create more equitable working
environments, women, 353
inclusivity-aware mentorship
training, 352
mentorship goals, 351
monitor progress, exit
dysfunctional mentor-mentee
relationship, 352
multiple mentors, 351
recognize and reward
mentorship, 352
women mentees benefit, women
mentors, 350
training mentors, 339
vexatious labor, 338
Women-only game jams, 279
Workforce diversity, 185, 198
Workplace harassment, 174
World Bank Human Capital Index
(HCI), 371