

Doctoral Dissertation

Understanding Newcomer Activities Prior to Onboarding Open Source Software (OSS) Projects on GitHub

IFraz Rehman

Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Kenichi Matsumoto
Software Engineering Lab. (Division of Information Science)

Submitted on January 29, 2024

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

IFraz Rehman

Thesis Committee:

Supervisor Kenichi Matsumoto
(Professor, Division of Information Science)
Shoji Kasahara
(Professor, Division of Information Science)
Takashi Ishio
(Professor, Future University Hakodate)
Raula Gaikovina Kula
(Associate Professor, Division of Information Science)

Understanding Newcomer Activities Prior to Onboarding Open Source Software (OSS) Projects on GitHub*

IFraz Rehman

Abstract

Understanding newcomer activities prior onboarding Open Source Software (OSS) Projects on GitHub holds significance for researchers and practitioners seeking insights into their preparatory practices prior to onboarding. Firstly, I map related work on onboarding of developers 102 studies. I present key topics offering insights into current trends and gaps in developer onboarding, and motivate this thesis. To fill the gap, I conduct an empirical study that is broken into three parts: (i) identifying Newcomers through survey who are potential candidate to OSS projects, I find 171 Newcomer OSS-candidates (i.e., 85%) with no prior experience contributing OSS, and have (i.e., 82%) intention to later onboard to OSS projects. (ii) validate their pre-onboarding activities and characterize them through mixed method approach, finding shows Newcomer OSS-candidates like to target software-based repositories (i.e., 66%), their first contributions are mainly associated with development (commits) and maintenance (PRs), and are less likely to practice social coding. (iii) Finally, analyzing proportion of them who are onboard to OSS projects in GitHub, I find that Newcomer OSS-candidates eventually end up onboarding (i.e., 30% quantitative, 70% follow-up survey) an OSS project. Furthermore, they cite finding a way to start as the most challenging barrier to contribute. Suggestions for Newcomers should not be afraid to individually contribute to their own code, contribute adding new content or making documentation to upstream software repositories, or fork OSS projects before

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, January 29, 2024.

attempting to onboard. For OSS projects it might start with tasks to update the documentation, formatting or cleaning up code. For Researchers the majority of targeted repositories are software-based, this insight helps to understand the role of software based experimental, documentation, and web-based-application-libraries-and-frameworks repositories in platforms like GitHub, that should cater for developers.

Keywords:

Open Source Software, Onboarding, GitHub, Newcomers, Human Aspect, Software Engineering

Acknowledgements

I wish to extend my profound appreciation to my thesis advisor, Professor Kenichi Matsumoto, for affording me the privilege of conducting my doctoral research under his esteemed guidance at NAIST. His unwavering support and encouragement have been instrumental throughout my academic journey.

Furthermore, I am deeply grateful to Associate Professor Raula Gaikovina Kula, whose valuable assistance, guidance in experimental challenges, and insights into new research avenues have played a pivotal role in steering my research endeavors from their inception. His patience and continuous support have been indispensable since my very first day at NAIST.

I would also like to express my sincere gratitude to the members of my thesis committee, namely Professor Shoji Kasahara and Professor Takashi Ishio, for their expert feedback and constructive suggestions, which have significantly contributed to the enhancement of the quality of my thesis.

Special recognition is owed to my dear friends, Yusuf Sulisty Nugroho, Syful Islam, Wang Dong, and Bodin Chinthanet who have not only been trusted confidants but also akin to brothers, providing me with their wealth of experiences and valuable advice. Our shared moments at NAIST have been truly treasured.

I extend my appreciation to the camaraderie I have enjoyed with my fellow students and labmates at the Software Engineering laboratory, making my PhD studies a time of joy and intellectual growth. Additionally, I wish to express my gratitude to my family and friends in Pakistan for their unwavering love, support, and encouragement in my pursuit of a doctoral degree in Japan.

Lastly, I reserve my deepest gratitude for my devoted wife, whose unwavering moral support has been the bedrock of my doctoral journey. Her encouragement and unwavering understanding have provided the strength I needed. I count myself fortunate to have her steadfastly by my side.

In closing, I extend my sincere appreciation to the Ministry of Education, Culture, Sports, Science and Technology (MEXT) and the dedicated staff at NAIST, particularly the International Affairs Division, for their unwavering support in various capacities.

List of Publications

Journal paper

- **Newcomer OSS-Candidates: Characterizing Contributions of Novice Developers to GitHub**

Ifray Rehman, Dong Wang, Raula Gaikovina Kula, Takeshi Ishio, Kenichi Matsumoto. Empirical Software Engineering (EMSE), 2022, Volume 27, Pages 109.

List of International Presentations

- **Newcomer Candidate: Characterizing Contributions of a Novice Developer to GitHub**

Ifray Rehman, Dong Wang, Raula Gaikovina Kula, Takashi Ishio, Kenichi Matsumoto. 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, SA, Australia, 2020.

Contents

Abstract	ii
Acknowledgements	iii
List of publications	iv
Contents	iv
List of Figures	viii
List of Tables	viii
1 Introduction	1
1 Problem Statement	2
1.1 The Onboarding Problem	2
2 Contributions	3
3 Thesis Structure	5
2 Background	7
1 The Evolution of Open Source Software (OSS)	7
2 The Role of Newcomers in OSS	8
3 The Impact of Social Coding Platform	9
4 Research Gap and Thesis Focus	10
5 Chapter Summary	11

3	Systematically Map Related Work	12
1	Introduction	12
1.1	Chapter Organization	13
2	The Systematic Mapping Process	14
3	Results: Maps of Onboarding Research	25
4	Threats To Validity	28
5	Related study	30
5.1	Studies on Human Aspect of Software Engineering	30
5.2	How are project-specific forums utilized? A study of participation, content, and sentiment in the Eclipse ecosystem	32
6	Conclusion	33
4	Newcomer OSS-Candidates: Characterizing Contributions of Novice Developers to GitHub	36
1	Introduction	36
2	Identifying Newcomer OSS-Candidates through Survey	40
3	Validating Pre-Onboarding Activities and Characterize them through Mixed Method Approach	42
3.1	(RQ1) What kinds of repositories does a Newcomer OSS-Candidate target?	42
3.2	(RQ2) What are the kinds of first Contributions that come from Newcomer OSS-Candidates?	47
3.3	(RQ3) To what extent do Newcomer OSS-Candidates practice social coding with their first contributions? * Social Coding: in Terms of Multiple Authorship	50
4	Analyzing Proportion of Onboarding to OSS Projects in GitHub	52
4.1	(RQ4) What is the proportion of Newcomer OSS-Candidates that eventually onboard an OSS project?	53
5	Discussions	56
5.1	Lessons learned	56
5.2	Implications (Expectations vs. Actual Results)	57
6	Threats to Validity	59
7	Related Work	60
7.1	Studies on Onboarding Motivators:	60

7.2	Studies on Onboarding to Organizations	62
7.3	Studies on the Onboarding Process:	63
7.4	Studies on Social Coding on GitHub	65
7.5	Studies on the barriers to Onboarding:	66
8	Conclusion	68
5	Conclusion	69
1	Contributions	70

List of Figures

1.1	An overview of the scope of the thesis.	5
3.1	Mapping study process.	15
3.2	Defined terms used in the search strings	16
3.3	Distribution of Paper Publication	21
3.4	Distribution of Research Types	22
3.5	Example of a forum thread.	31
3.6	Frequency of posts per user status. The maximum number of posts for each type of users is used to define the threshold of post-based membership. The threshold for Juniors and Members are 29 and 106, respectively. Although Seniors have posted more than 28 thousand posts, I limit up to 600 in the figure.	33
4.1	Frequency for contributed repository kinds with Fork and Upstream. Experimental and Documentation are the most frequently targeted software repository kinds, i.e., 24% and 21%, respectively.	46
4.2	An example of how I define developers practice social coding, where more than one author contributes to the git.gemspec file.	50
4.3	Identify social coding in terms of whether a contribution is modified by a single author or multiple authors.	51
4.4	Qualitative analysis using a follow-up survey to acquire the perception of Newcomer OSS-Candidates.	54

List of Tables

3.1	Caption for LOF	16
3.2	Papers statistics during the filtration and screening phases.	19
3.3	Exclusion and Inclusion Criteria.	19
3.4	Topics.	24
3.5	Result of areas and topics.	26
3.6	Outputs of the pre-processing of the forum dataset	32
4.1	Survey Questions sent to potential respondents	41
4.2	Two questions in survey	42
4.3	Proportion of software and non-software repositories targeted by Newcomer OSS-Candidates. Around 66% of Newcomer OSS-Candidates tar- get Software repositories.	44
4.4	Frequency for Contribution’s Kinds of Newcomer OSS-Candidates. In the first commits, 43% of Newcomer OSS-Candidates are typi- cally engaged in repository initializing activities, and 60% are en- gaged in the management activities of the PRs.	49
4.5	Frequency of social and non-social contributions from Newcomer OSS-Candidates in terms of single/multiple authorship. After join- ing GitHub, 73% and 59% of Newcomer OSS-Candidates have non- social based contributions in their first commits and PRs.	52
4.6	Frequency of Newcomer OSS-Candidates that started the onboard- ing process for OSS repositories.	55
5.1	Guidelines for (N)Newcomers and (PM)Project Maintainers	71

1 | Introduction

Over the course of several years, software companies worldwide have actively interacted with Open Source Software (OSS) projects and their corresponding communities. The motivation for participating in, support, or even the initiation of OSS projects often arises from the aim to curtail development expenditures and foster heightened levels of innovation [44]. OSS development exhibits substantial parallels with the realm of Global Software Development (GSD), sharing numerous characteristics. Typically, OSS projects operate in a highly decentralized manner, encompassing participants hailing from diverse geographical regions and cultural contexts.

Onboarding within the realm of Software Engineering (SE) is recognized as a critical practice, significantly influencing both open source software (OSS) and proprietary software projects. Over the past decade, advancements in onboarding tools and methodologies have facilitated more efficient integration processes, gaining widespread acceptance in both OSS [33, 91] and organizational sectors [8, 105].

On the other side, GitHub, renowned as the premier open-source version control platform, hosts a vast array of over 3 hundred million repositories and more than a hundred million developers¹. It functions as a social coding platform for both software and non-software projects. The inclination of individuals to establish new teams or transition to existing ones is significantly enhanced by the presence of technical standards and platforms. GitHub, in particular, serves as a prevalent and widely embraced distributed platform for code sharing and version control. Developers commonly possess a high level of familiarity with systems like git, which is supported by GitHub, and the collaborative processes it enables,

¹<https://octoverse.github.com/>

thereby streamlining the process of contributing to, initiating, or transitioning to team projects on this platform. Nevertheless, GitHub extends its facilitative role in team formation and migration beyond technical aspects, thanks to its incorporation of *social coding* features. These features enable developers to monitor and assess each other's activities, thereby forming comprehensive impressions of their social and technical competencies and conduct².

However, literature underscores the criticality of continually incorporating newcomers into GitHub OSS projects and onboard them to the development process for the success of these ventures. The thesis "Understanding newcomer activities prior to onboarding Open Source Software (OSS) Projects on GitHub", holds significance for researchers and practitioners seeking insights into newcomers' preparatory practices before onboarding. The initial part of the thesis embarks on a systematically map related work on onboarding of developers 102 studies. I present key topics offering insights into current trends and gaps in developer onboarding and motivate this thesis. In the second part, to fill the gap, I conduct the research adopting an empirical approach, that is broken into three parts: (i) identifying Newcomers through survey who are a potential candidate for OSS projects, (ii) validating their activities before onboarding and characterize them through mixed method approach, and finally (iii) analyzing their proportion who eventually onboard to OSS projects in GitHub. The collective goal of these studies is to offer a comprehensive understanding of pre-onboarding practices within OSS communities. This entails illuminating prevailing trends, identifying research gaps, and shedding light on potential areas for future research and enhancements in this vital domain.

1 Problem Statement

1.1 The Onboarding Problem

Incorporating newcomers into a new software development project often presents them with various types of barriers, as detailed in previous research [4]. *Technical barriers* newcomers frequently face issues arising from the high complexity of

²<https://github.com/about>

the systems under development [4, 94]. This complexity is exacerbated by their lack of prior knowledge in the specific domain where the development activities are taking place [94]. *Process barriers* in this category, newcomers encounter difficulties in gaining a comprehensive understanding of the software they are tasked with contributing to, as well as in determining where to initiate their work [94, 107]. *Interpersonal barriers*, these barriers manifest as communication challenges when newcomers are integrated into a diverse team, where individuals with distinct objectives, varied cultural backgrounds, and differing interpersonal skills collaborate [4].

These barriers exert a notable influence not only on newcomers but also on various other stakeholders engaged in the software development process. Newcomers frequently contend with feelings of frustration [76, 89]. Senior developers, in their newly assigned mentoring roles, often encounter difficulties and challenges [4]. Consequently, project managers must grapple with the ramifications of reduced productivity and the allocation of senior developers' time to mentor newcomers, leading to significant trade-offs and resource allocation concerns [76]. Despite the above attempts, it remains unclear (i) to find the current trends and gaps in key topics offering insights into developers' onboarding, a systematic thorough review of related work, and (ii) identifying newcomers OSS-candidates, characterizing pre-onboarding activities, and analyzing onboarding proportion to GitHub OSS projects. Therefore, I state this thesis as follows:

Thesis Statement: An effective onboarding for newcomers requires a proper understanding of practice activities before onboarding OSS projects. If the needed information is at hand, it will reduce newcomers' and project maintainers' barriers and further improve onboarding efficiency. This thesis presumes that Newcomer OSS-candidates practice before they onboard to open-source software projects on GitHub.

2 Contributions

This thesis makes several significant contributions to the human aspect of Software Engineering, particularly in the context of understanding newcomers' prior

activities before join to Open Source Software (OSS) projects on GitHub. These contributions are not only novel but also address critical gaps in the existing literature, offering practical insights for both researchers and practitioners. The key contributions are as follows:

- **Literature Review.** This thesis presents a comprehensive systematic review of the existing literature on developers' onboarding in Software Engineering. This review is significant as it consolidates key topics and identifies gaps in current research, providing a foundation for further studies in this area.
- **Empirical Study.** An in-depth empirical study is conducted to characterize the contributions of novice developers to GitHub. This study is original in its approach and offers a detailed analysis of the kinds of repositories and activities by newcomers, which were previously unexplored. The results provide valuable insights into the practices and challenges faced by new developers in OSS projects.
- **Term Newcomer OSS-Candidate.** This thesis introduces and defines the term 'Newcomer OSS-Candidate', a concept not previously defined in academic literature. This term helps in precisely identifying and studying a specific group of developers who are in the preliminary stages and have the potential to contribute to OSS projects.
- **Characterization of Newcomers' Activities and Repository Kinds.** The research provides a novel characterization of activities and repositories kinds by Newcomer OSS-Candidates. This contribution is significant as it sheds light on the initial steps taken by newcomers and the nature of their contributions, which can inform better onboarding strategies.
- **Development of Replication Package.** To support and encourage further research in this area, a replication package has been created and made available. This package includes detailed data and methods used in the empirical study of Newcomer OSS-candidates. It can be accessed at at: <https://github.com/NAIST-SE/NewcomerCandidate>. This package not

only enhances the transparency and reproducibility of the current research but also provides resources for future studies to build upon.

In summary, these contributions represent a substantial advancement in understanding the joining process of newcomers to OSS projects. They provide a deeper insight into the pre-onboarding activities, motivations, and challenges faced by newcomers, which is crucial for developing effective strategies for integrating newcomers into OSS projects.

3 Thesis Structure

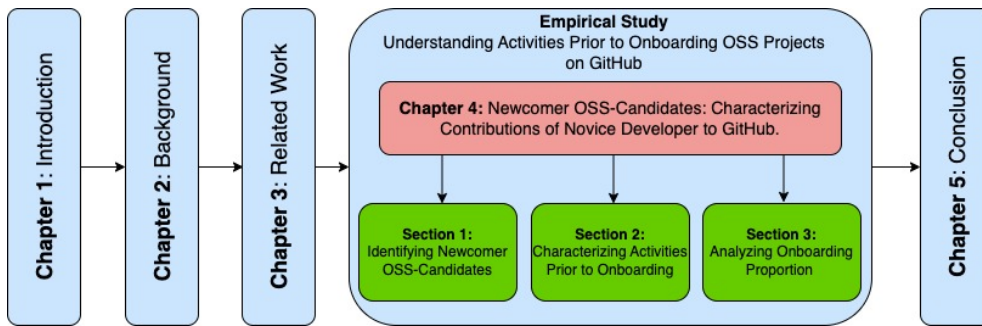


Figure 1.1. An overview of the scope of the thesis.

In this section, I describe an outline of this thesis. Figure 1.1 illustrates the structure of the thesis. The details of the rest of this thesis are listed as follows:

- **Chapter 2** presents the background of this thesis which comprises (i) The Evolution of OSS, (ii) The Role of Newcomer in OSS, (iii) The impact of Social coding Platform, and (iv) Research gap and thesis focus.
- **Chapter 3** presents a related work on key topics addressed on Developers' Onboarding in Software Engineering.
- **Chapter 4** presents an empirical study about Newcomer OSS-candidates: characterizing contributions of novice developer to GitHub which is divided in three sections: (i) Identifying Newcomer OSS-Candidates (ii) Validating and Characterizing their activities prior to onboarding, and (iii) analyzing the proportion who onboard to GitHub OSS projects

- **Chapter 5** finally, concludes the all studies in this thesis.

2 | Background

The purpose of this chapter is to describe the background of this thesis. Section 1 introduces the evolution of Open Source Software (OSS). Section 2 introduces the role of newcomers in OSS. Section 3 introduces the impact of social coding platforms. Section 4 introduces the research gap and thesis focus. Finally, Section 5 describes the summary of the background.

1 The Evolution of Open Source Software (OSS)

The development model of Open Source Software (OSS) is not a recent innovation but has historical roots dating back to the 1990s. During this period, notable software endeavors, including The Linux Project, embraced the practice of engaging users to identify and rectify software defects, as well as contribute to code improvement. This approach emerged as a pragmatic means of identifying and addressing software issues efficiently. The early emergence of OSS aligns with the advent of ARPAnet, one of the earliest computer networks, where source code was informally exchanged within the hacker community. A prominent advocate for the concept of open software during this era was Richard Stallman, the founder of the Free Software Foundation. Stallman passionately argued for the "moral" imperative of open software, advocating against the prevailing proprietary ownership of code [29, 79]. Throughout the 1990s, the evolution of OSS involved the dissemination of code through a mechanism known as a "tarball," which was essentially a compressed .tar file used for code distribution. This method of sharing code was commonly facilitated through email communication [30].

Other side, Lehman and his research team have established a substantial and

widely recognized body of research concerning the evolutionary dynamics of large, enduring software systems, as evidenced by their notable contributions in publications such as [52, 53, 54], and [103]. Central to Lehman's work are the principles encapsulated within Lehman's laws of software evolution [53], which have been derived from empirical investigations into various extensive software systems. These laws posit that as software systems expand in scope and complexity, the task of introducing new code into these systems becomes progressively challenging, necessitating deliberate efforts to restructure the overall design. Furthermore, the empirical findings of Turski, who conducted statistical analyses based on these case studies, have shed light on the growth patterns exhibited by such software systems [53, 103]. According to Turski's analyses, the growth of these systems, as measured by indicators such as the number of source modules and the frequency of module modifications, typically follows a sub-linear trajectory. In simpler terms, as software systems increase in size and complexity, the pace of growth tends to decelerate, reflecting the diminishing rate of change and adaptation as the system matures.

2 The Role of Newcomers in OSS

In the context of Open Source Software (OSS) development, a "newcomer" refers to a developer who is attempting to make their initial code contributions to a particular OSS project [92]. The process of onboarding newcomers into OSS projects can often present challenges characterized by unfamiliar and potentially unwelcoming environments. As posited by Fogel [36], the initial experience a newcomer encounters within an OSS project significantly influences their decision to continue engaging with it; unfavorable first impressions may deter them from further involvement. This situation holds particular relevance for individuals enrolled in software engineering courses, particularly students who are at the nascent stages of their development and skill acquisition in the field. Newcomers require effective orientation and guidance to navigate the intricacies of the project and to successfully make meaningful contributions [96]. The process of motivating, engaging, and retaining new developers within an OSS project holds pivotal importance in maintaining a vibrant and sustainable OSS community [78].

Drawing an analogy, Dagenais et al. [25] liken newcomers to explorers embarking on a journey through a potentially challenging and unfamiliar terrain, where they must rely on self-guidance to navigate the tasks and obstacles inherent in the OSS environment.

On the other side, the involvement of newcomers in Open Source Software (OSS) projects is typically marked by enthusiastic participation; nevertheless, their integration into these communities is impeded by a range of social and technical obstacles, as documented in [41, 63, 93], and [95]. Notably, newcomer onboarding within OSS communities is often perceived as challenging primarily due to the substantial technical prerequisites that must be met. However, research findings indicate that social impediments to onboarding are not only more prevalent but also more severe than their technical counterparts [95]. In contrast to conventional software development practices, effective communication emerges as a critical social obstacle for newcomers in OSS communities. This challenge arises from the limited opportunities for team members to engage in face-to-face interactions, compounded by differences in time zones and cultural backgrounds [70, 100].

3 The Impact of Social Coding Platform

GitHub, functioning as a prominent social coding platform, occupies a central role within the domain of Open Source Software (OSS). It serves as a widely adopted version control platform that functions as a repository for hosting software projects, encompassing features such as pull request management, issue tracking, and workflow facilitation, as documented in [57, 77]. At the juncture of this study, GitHub boasted an impressive user base, hosting in excess of 330 million repositories while catering to the needs of over 100 million developers¹. GitHub's significance extends beyond its technical utility, as it incorporates *social coding* features that play a pivotal role in fostering team dynamics and facilitating project migration. These social coding features empower developers to engage in the monitoring and evaluation of each other's actions, thereby enabling the formation of comprehensive assessments concerning both their technical profi-

¹<https://github.com/about>

ciencies and their interpersonal conduct. Notably, GitHub preserves and exposes the actions undertaken by developers, including activities such as bug reporting, pull request submissions, and commenting. These activities are not merely archived but can be readily tracked by fellow users, and accessible through user-designated "home" pages. Consequently, these records serve as valuable sources of information that not only allow for the evaluation of an individual's technical expertise but also offer insights into their social acumen and inclinations. It is worth noting that one of the chief advantages of a social coding platform like GitHub lies in its capacity to aggregate contributions to software projects from within the community, as elucidated in [21]. The continuous influx of newcomers into these projects, along with their active participation in the development process, emerges as a pivotal factor underpinning the success of these endeavors, as underscored by [71].

4 Research Gap and Thesis Focus

Research Gap and Motivation: Despite the known increase in the number of potential OSS contributors at social coding platforms like GitHub, there is a notable lack of understanding regarding the activities and experiences of these individuals before they join OSS projects. This gap is crucial because pre-onboarding experiences can significantly shape a newcomer's ability to contribute effectively to OSS projects. Filling this research gap is essential for the sustained success and growth of OSS projects. Understanding the pre-onboarding activities of potential contributors is key to developing more effective onboarding processes and retention strategies, ultimately contributing to the health and longevity of OSS communities.

Relevance to OSS Success:

- **Skills and Knowledge Assessment:** Understanding the prior experiences of new OSS contributors could help in assessing the skills and knowledge they bring, which is vital for effective integration into projects.

- Tailored Onboarding Processes: Knowledge of pre-onboarding activities could inform the development of more personalized and efficient onboarding processes, catering to the varied backgrounds of new developers.
- Retention Strategies: Insights into the motivations and prior engagements of newcomers can aid in devising strategies to not only attract but also retain these contributors in OSS projects.

5 Chapter Summary

The chapter provides a comprehensive background concludes by emphasizing the significance of research in understanding newcomer activities before onboarding to Open Source Software (OSS) projects on GitHub. This study is crucial for practitioners and researchers as it provides insights into the preparatory practices of newcomers. It involves a systematic mapping of related work and an empirical study divided into three parts: identifying newcomers, validating and characterizing their pre-onboarding activities, and analyzing the proportion who onboard to GitHub OSS projects. This comprehensive approach illuminates pre-onboarding practices within OSS communities, identifies research gaps, and offers insights for future research and improvements in the onboarding process. The goal is to enhance the understanding of OSS communities and inform more effective onboarding and retention strategies.

3 | Systematically Map Related Work

The proliferation of onboarding tools has led to increased data availability and has catalyzed a substantial body of research in the field of onboarding over the past decade. This chapter initiates a systematic mapping study aimed at scrutinizing the aspects of key topics addressed in research pertaining to developers' onboarding in Software Engineering (SE). The primary objective of this mapping study is to provide valuable insights for both researchers and practitioners, facilitating their comprehension of the challenges inherent in onboarding while enabling them to stay abreast of the latest practices and state-of-the-art developments in the field.

1 Introduction

Onboarding refers to the process by which developers and employees become part of an Open Source Project (OSS) or an organization. This subject has a rich history in organizational studies, dating back to the 1970s [8, 105], and has garnered substantial attention in the context of OSS development projects [33, 91]. In the context of onboarding to OSS projects, community-based OSS initiatives rely heavily on a steady stream of new developers for sustainability and growth [37]. Consequently, researchers have extensively studied the motivations [83, 109] and barriers [4, 96] faced by newcomers to enhance their recruitment and ease onboarding challenges. This has led to a comprehensive body of research investigating the motivations [62, 82] of OSS developers and the factors that make projects appealing by overcoming the barriers [95, 111] for developers.

On the other side, in the field of software development organizations, the literature generally does not differentiate between development methodologies, such as agile or non-agile [25, 84]. However, some research focuses specifically on onboarding within agile software development teams: For instance, Buchan et al. [15] include investigations into effective onboarding techniques for agile teams, Britto et al. [14] analysis of onboarding in globally distributed legacy projects without emphasizing agile peculiarities, Dagenais et al. [25] identification of orientation aids for newcomer integration in agile projects, and Barroca et al. [6] observations that effective integration of newcomers is crucial for the long-term sustainability of agile practices.

This paper collects topics addressed for onboarding studies, with the end goal to emphasize the significant evolution of onboarding practices in Software Engineering (SE), particularly in open source software projects and software development organizations. It outlines the advancements in onboarding tools and methodologies over the past decade and their role in streamlining the integration process. The scope of the systematic study revolves around a research question: aims to provide insights to cover the (*RQ1*) topics addressed in SE, by analyzing 102 premium studies. It seeks to offer insights into current trends, gaps, challenges, and validity threats in these studies, contributing to a deeper understanding of current trends and gaps practices in developer's onboarding.

Throughout the collection of 459 papers from the high-impact SE venues, I generate statistics for the 102 collected papers including 46 conferences and 56 journals. Results, concerning about topics addressed, I identify that the majority of studies have leveraged data on contributions to gain a deeper understanding of people (38 studies) and systems (21 studies) within the comprehension of the software engineering process. Conversely, the characterization of contribution assessments has been less frequently addressed, with an emphasis on the evaluation of team or developer performance when such characterization occurs.

1.1 Chapter Organization

The remainder of this chapter is organized as follows. Section 2 presents the systematic mapping process, including the research question, search conduction, and screening process. Section 3 shows the results of the systematic mapping

study. Section 4 explains the threats to the validity of the research. Finally, I draw our conclusion from this related work in Section 5.

2 The Systematic Mapping Process

The methodology employed in this study draws inspiration from the work of Hamer et al. [40] and bears resemblance to the systematic mapping study conducted by Petersen et al. [74]. In essence, the systematic mapping study comprises a series of procedural steps, including the formulation of research inquiries, the systematic retrieval of relevant academic papers, the screening process, keyword assignment for mapping purposes, and the extraction of pertinent data.

In shaping the aim of this research, I have also integrated the Goal Question Metric (GQM) model introduced by Caldiera and Rombach [17]. The purpose of this study is to examine the topics of research, the potential validity threats, and the challenges encountered, all from the perspective of researchers within the realm of software engineering studies. This investigation was carried out through a systematic mapping study, adhering to established guidelines in software engineering [74, 75]. An overview of this mapping study’s process is depicted in 3.1, documentation of each stage, encompassing the preliminary selected papers and the results from each phase of the screening process. This methodology is divided into three primary stages: formulating the research questions, executing the search conduct, and evaluating the screening of papers.

Research Questions

To define the scope of the mapping study in the respect of topics addressed, I formulate the research question:

1. (*RQ1*): What topics have been addressed in SE onboarding studies? The motivation for this research question is to uncover the fields where research on developer contributions has been conducted, with the goal of comprehending the topics explored by these studies. This knowledge can provide context to the researched areas and serve as a motivation for future work.

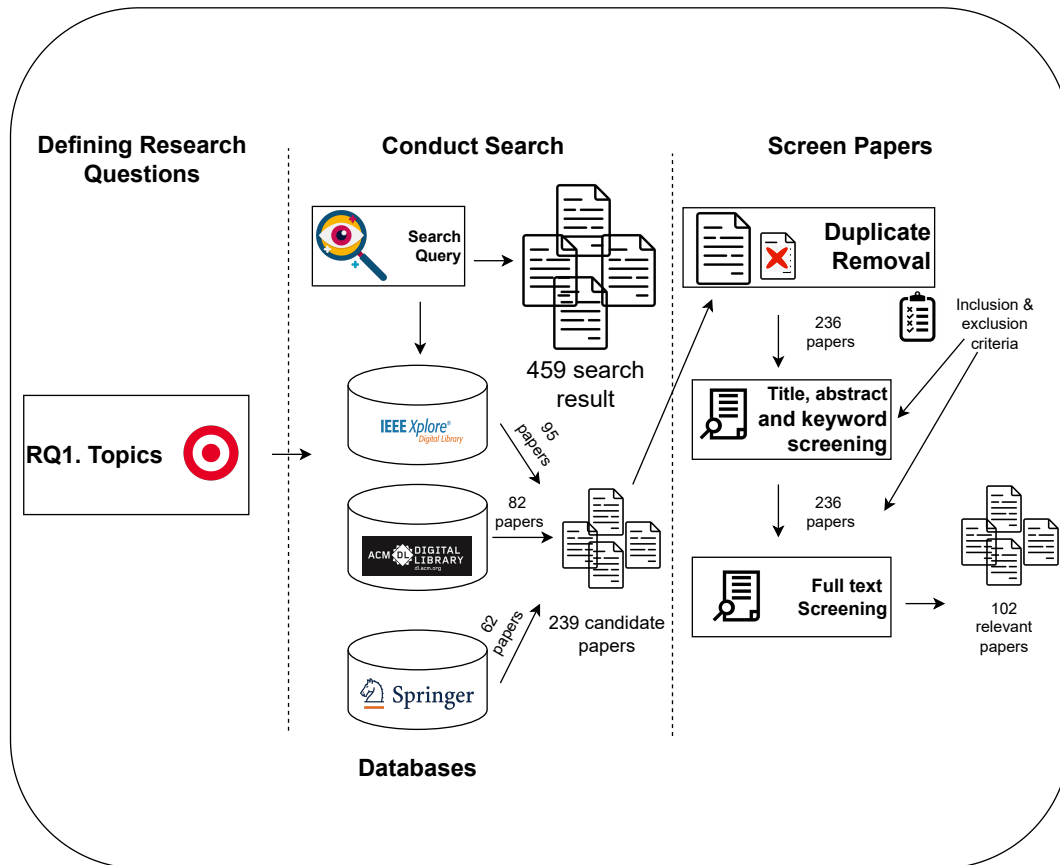


Figure 3.1. Mapping study process.

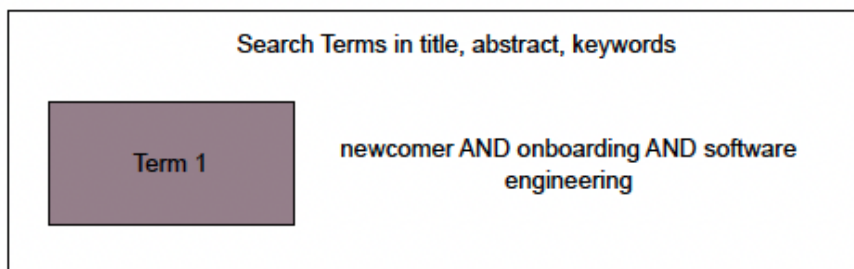


Figure 3.2. Defined terms used in the search strings

Table 3.1. Corpus of venues (conferences and journals) studied in this paper. Note that ICSM now is called ICSME; and WCRE and CSMR are fused into SANER.

Journal	Name	Impact factor	Established
TSE	IEEE Transactions on Software Engineering	7.4	1991
EMSE	Empirical Software Engineering	4.1	1996
IST	Information and Software Technology	3.9	1992
TOSEM	Transactions on Software Engineering and Methodology	3.6	1992
S/A	IEEE Access	3.5	1983
JSS	The Journal of Systems and Software	3.5	1991
SPE	Software: Practice and Experience	3.5	1991
ASEJ	Automated Software Engineering	3.4	1994
S/W	IEEE Software	3.3	1991
SOSYM	Software and System Modeling	2.0	2002

Conference	Name	h5-index	Established
ICSE	International Conference on Software Engineering	85	1994
FSE	ACM SIGSOFT Symposium on the Foundations of Software Engineering	60	1993
PL	Proceedings of the ACM on Programming Languages	56	2017
ASE	IEEE/ACM International Conference on Automated Software Engineering	50	1994
PLDI	ACM SIGPLAN Conference on Programming Language Design and Implementation	50	1979
MSR	Mining Software Repositories	46	2004
ISSTA	International Symposium on Software Testing and Analysis	40	1989
SANER	IEEE International Conference on Software Analysis, Evolution and Re-engineering	39	2014
ICSME	IEEE International Conference on Software Maintenance and Evolution	35	1994
TACAS	International Conference on Tools and Algorithms for the Constructions and Analysis of Systems	34	1998
PPOPP	ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming	33	1988
COMPSAC	IEEE Annual Computer Software and Applications Conference	32	1977

Conduct Search

In formulating our search string, I adhere to the stringent criteria outlined by A. Kitchenham [1], which include: (i) employing a defined search strategy, (ii) using a specified search string created from a list of synonyms connected by ANDs, (iii) utilizing a wide range of search sources, (iv) meticulously documenting the search process, and (v) ensuring that at least two researchers verify the selection of papers. This approach is exemplified in 3.2, which displays our specific search string. For the first term of the search string, as illustrated in 3.2, I incorporate widely recognized terms related to Onboarding, such as 'onboarding', 'newcomer', and 'software engineering'. This is to ensure the search yields results pertinent to the onboarding process within the domain of software engineering.

In alignment with the research question and to ascertain papers of exceptional quality while comprehending the cutting-edge advancements in the field, the search was deliberately focused on literature published in prestigious journals and conferences within the software engineering discipline, spanning the period from 2013 to 2023. Drawing inspiration from Hamer et al. [40] findings, I choose 2013 as the initial point for selection. This decision is based on Hamer et al.'s observation of a consistent increase in the volume of research pertaining to studies on developers' onboarding, a trend that has been evident since 1989.

In the selection of publication venues, the approach mirrored the mapping study by Mathew et al. [60]. I sourced papers from 12 conferences known for their significant h5-index and 10 journals recognized for their high impact factors. The h5-index, derived from Google Scholar, represents the h-index for articles published over the past 10 complete years. The collection included works from prominent conferences such as the International Conference on Software Engineering (ICSE), Mining Software Repositories (MSR), Symposium on the Foundations of Software Engineering (ESEC/FSE), and the International Conference on Software Analysis, Evolution and Re-engineering (SANER). In terms of journals with high impact factors, the selection encompassed publications like Transactions on Software Engineering (TSE), Empirical Software Engineering (EMSE), Information and Software Technology (IST), and Transactions on Software Engineering and Methodology (TOSEM). The methodology for obtaining the h5-index uti-

lizes Guide2Research¹ as the primary resource. The Impact Factor, a numerical index, is employed to assess the influence of scientific journals, with data sourced from Clarivate Analytics². Conferences boasting higher h5-indexes, along with journals possessing elevated impact factors, are considered to hold greater inherent prestige within their respective academic fields. A comprehensive summary of these paper collection sources is presented in Table 3.1.

In an effort to mitigate selection bias, the initial selection encompassed an extensive array of digital repositories, including IEEE Xplore, Science Direct, ACM Digital Library, and SpringerLink databases. These databases are frequently utilized in secondary studies within the field of software engineering [2]. I conducted the main search in October 2023. The search query retrieved 459 search results over the past decade from 2013 to 2023 (66 from IEEEExplore, 104 from ACM Digital Library, 39 from SpringerLink, 12 from Science Direct, and 15 from other digital sources) from the mentioned main four sources.

The evaluation of the primary papers' quality serves as an additional criterion for exclusion in the study [46]. In the quality assessment process, I selectively included papers from esteemed venues, under the presumption of their high quality and notable recognition within the software engineering (SE) community. Furthermore, to focus solely on technical contributions, the subsequent data processing excluded shorter papers, workshop contributions, books, theses, editorials, tutorials, panels, poster sessions, prefaces, and opinion pieces (specifically those 4 pages or fewer in length). It is acknowledged that inclusion and exclusion criteria to this approach exist which is addressed in greater detail in next Section. Following the search conduct process, I identified an initial set of 236 papers, as detailed in Table 3.2.

Screen Process

The selected studies were meticulously screened to ascertain their relevance, employing a defined set of criteria for both exclusion and inclusion. This process involved a manual review of each paper's title and abstract. The exclusion criteria established were as follows: *E1*: the Paper is not written in English, *E2*:

¹<https://www.guide2research.com/>

²<https://clarivate.com/webofsciencegroup/essays/impact-factor/>

Table 3.2. Papers statistics during the filtration and screening phases.

		# of Papers
Conduct Search		
	Search String Result	459
<i>All Papers</i>		236
Screening of Papers		
	Conference paper	46
	Journal paper	56
<i>Total Papers</i>		102

Table 3.3. Exclusion and Inclusion Criteria.

ID	Type	Criteria
E1	Exclusion	Not written in english
E2	Exclusion	Unrelated to onboarding software engineering
E3	Exclusion	Are not primary study
E4	Exclusion	Outside of time-frame
I1	Inclusion	Study developers' onboarding in software engineering
I2	Inclusion	Peer- reviewed Studies
I3	Inclusion	Written in english with full text available.

the paper is unrelated to onboarding in software engineering, $E3$: Non-primary research paper, $E4$: the Paper is outside of the designated study period. Concurrently, the inclusion criteria were outlined to encompass the scope, objectives, and assessments of the studies. The following papers are included that met these Criteria: $I1$: the Paper should be focused on onboarding, software engineering, and developers, $I2$: the Paper must be peer-reviewed, $I3$: the Paper needs to be in English and have the full text accessible. The criteria array, as presented in Table 3.3, was developed through a collaborative and iterative process. The evaluation formula applied for these criteria is expressed as: $NOT (E1 OR E2 OR E3 OR E4) AND (I1 AND I2 AND I3)$. The sequence in which these criteria are applied is first $E1$, followed by $E2$, $E3$, $E4$, and then $I1$, $I2$, and $I3$. Should any criterion not be satisfied, the evaluation of subsequent criteria is discontinued.

To reduce bias, this manual paper selection was conducted by the first and the second authors. In the initial phase of screening, a prevalent cause for the exclusion of works was the use of the term "contribution" in contexts unrelated to software contribution. Specifically, this pertained to instances where studies claimed contributions to the broader literature. Additionally, research focusing on the development of software products not pertinent to software engineering and onboarding practices was also frequently eliminated. For instance, studies introducing or employing software tools for applications in the medical or simulation domains were deemed outside the scope of this review. Furthermore, during the preliminary selection process, a significant number of papers were excluded for being secondary or tertiary sources rather than primary research works. However, any study deemed potentially relevant by either of the authors was carried forward to the subsequent stage of evaluation. In this latter phase, the primary reason for the disqualification of studies was their focus on evaluating contributions rather than investigating software engineering projects in depth.

The screening procedure culminated in the selection of 102 papers from an initial pool of 236. This curated collection comprises 46 papers from leading conferences and 56 from high-impact journals, as detailed in Table 3.2. The distribution of these papers, differentiated by conference and journal sources over the period under study, is illustrated in Figure 3.3. The data illustrated in the figure indicates a progressive increase in research publications related to Onboarding in Software Engineering (SE) over the most recent three-year period. This upward trajectory is evidenced by the publication of three papers in 2019, four in 2020, and a notable surge to nine papers in 2021. Concurrently, there has been a discernible rise in the number of papers submitted to journals in this field, particularly from 2020 onwards. This is exemplified by the submission of eight papers to various journals in the year 2021 and 2022, suggesting a growing academic interest and research activity in onboarding within the SE domain.

In an effort to more comprehensively understand the nature of research within the field, I conducted a manual classification of the types of research papers, aligning with the methodology outlined by Bernard [11]. This classification system segmented the papers into four distinct categories: Mixed-Method, Qualitative, Quantitative, and Survey or Interviews. The Mixed-Method category encapsu-

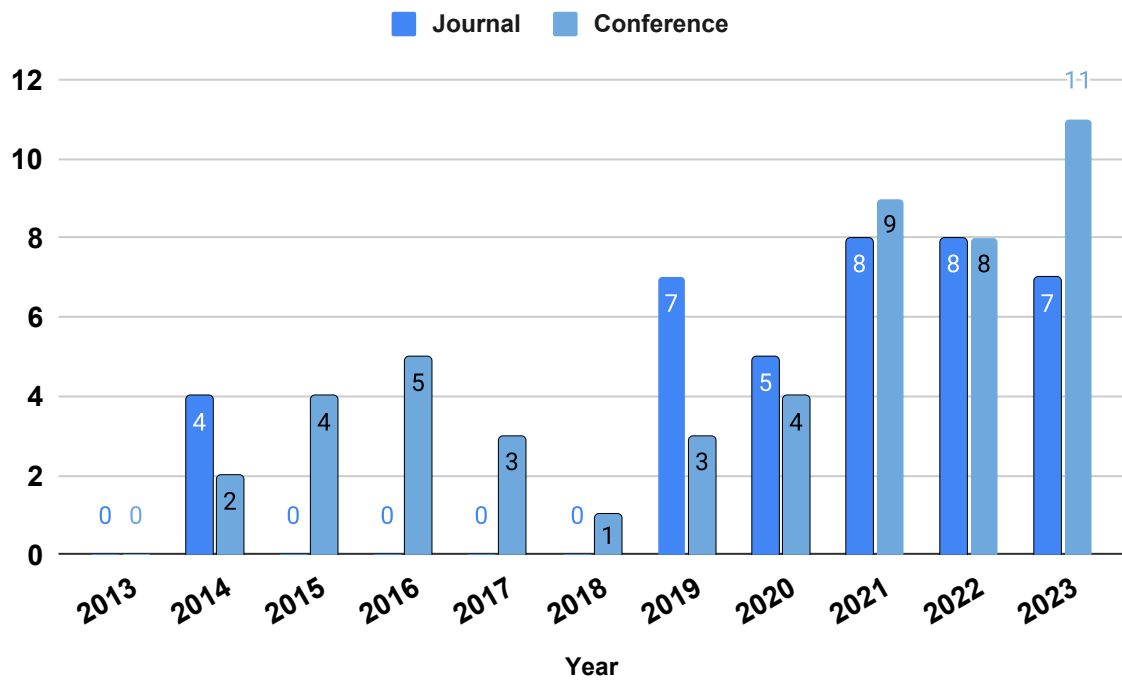


Figure 3.3. Distribution of Paper Publication

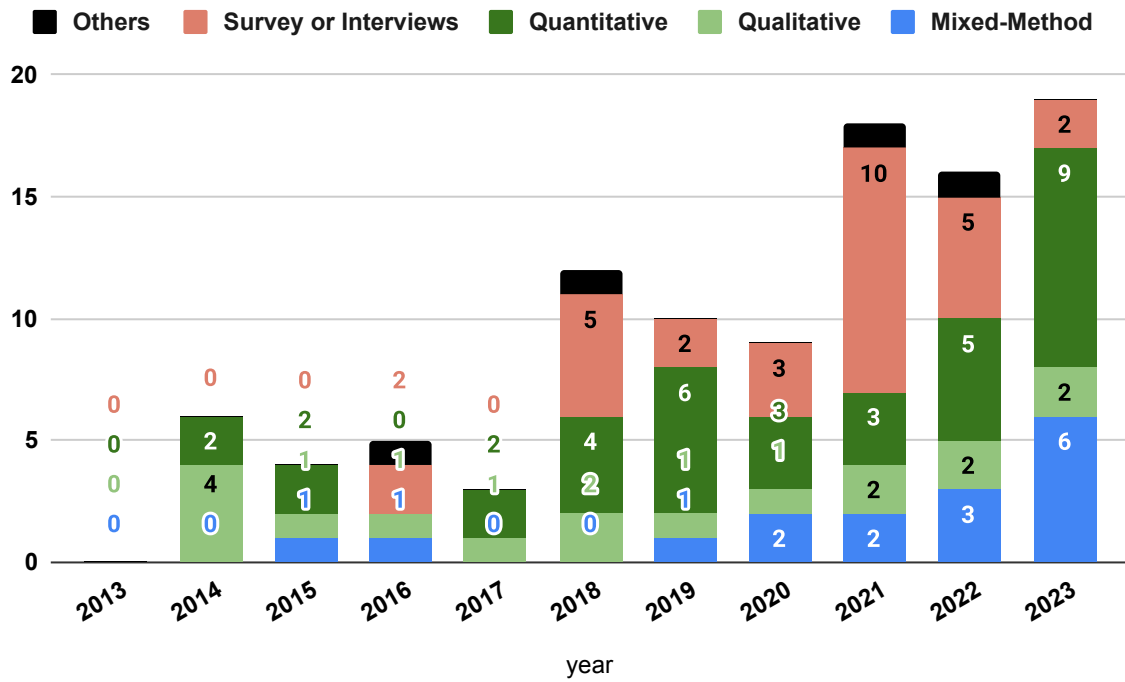


Figure 3.4. Distribution of Research Types

lates papers employing both quantitative and qualitative/survey methodologies. The Survey or Interviews category comprises studies utilizing surveys, interviews, and user/control studies. Papers that did not conform to these specified types were categorized under Others. The process of classifying the research paper types was executed in two stages. Initially, two authors undertook the first round of classification. Subsequently, in the second round, a third author with extensive research experience joined the effort to validate the categorization of each paper. The distribution of these research types across 102 papers, within the timeframe of the study, is depicted in Figure 3.4. Notably, there has been a recent surge in the popularity of Survey or Interviews based papers, with ten papers being published in the year 2021, indicating a shifting trend in research methodologies.

Keywording of Relevant Papers

The classification process of relevant studies entailed an in-depth examination of each paper, which involved not only a detailed analysis of the abstract but, in certain instances, necessitated a thorough review of the entire paper.

Analysis Approach for Topics Addressed (RQ1). In order to ascertain the scope of research topics addressed, I categorized the topics of the studies under investigation. This categorization was influenced by the framework proposed by [40], allowing for each paper to be classified into multiple topic categories. Table 3.4 delineates the 25 distinct sub-areas identified, which are further organized into four primary areas: Comprehend, Train, Construct, and Characterize. The methodology for this classification relies on the title, keywords, and abstract of the studies for initial categorization. This task was performed by two co-authors, who collaborated in the first round of classification, convening around a round table to assign each study to the predefined main areas. Subsequently, in the second round, a more granular classification was conducted, identifying further sub-areas within these main categories to enhance the depth of understanding.

Table 3.4. Topics.

Main-area & Sub-area	Topic
-Comprehend	—-Understanding Software Engineering Phenomena
People	<i>Organizational management</i> <i>Newcomers</i> <i>Collaboration or teamwork</i> <i>Communication</i> <i>Community management</i> <i>Personality</i> <i>Expertise or experience</i> <i>Roles</i> <i>Motivation</i> <i>Contribution inequality</i> <i>Disengagement</i>
Artifacts	<i>Contribution Patterns</i> <i>Communication</i>
Systems	<i>Project health</i> <i>Project success</i> <i>Ecosystem and project characterization</i> <i>Project diversity</i> <i>Migrations</i>
-Train	—-Teaching and training students <i>Education assessment</i> <i>Teaching courses</i>
-Construct Models	—-Creating models and artifacts <i>Recommendations systems</i> <i>Prediction Models</i>
Artifacts	<i>Tools</i> <i>Bots</i>
-Characterize	—-Describing contributions assessments <i>Team or developer performance</i>

3 Results: Maps of Onboarding Research

The results will answer the research question, with the table of the categories of the papers.

(RQ1): What topics have been addressed in SE onboarding studies?

In order to delineate the nature of topics addressed in the studies, I engaged in a classification of these elements. Table 3.5 provides a statistical overview, including the number of studies for each category (indicated within parentheses), and outlines potential opportunities for advancement in the field. The main categories in this classification are highlighted in **bold**, while the sub-categories are distinguished in *italics*. In the process of identifying each study, the designation 'CXXX' was employed, wherein 'C' signifies a contribution work and 'XXX' represents a unique three-digit identifier. The comprehensive list of the selected works can be found in Appendix A 1.

To ascertain the research domains explored, this study employed a systematic categorization of the topics under investigation. The categorization was derived from a careful examination of the titles, keywords, and abstracts of the papers. This approach allowed for the classification of each paper into one or more topic areas. As delineated in Table 3.5, a total of 25 distinct topics were identified and subsequently grouped into four principal areas of interest: **comprehend**, **train**, **construct**, and **characterize**. The table depicted in the provided categorizes studies within the domain of Software Engineering into main areas and sub-areas, each linked to specific research topics and associated studies.

The topic most frequently addressed in the literature, as reflected by 102 papers, was utilizing the information of the contribution assessment to **comprehend** to gain insights into software engineering phenomena. Within this area, a compilation of 18 distinct topics emerged, which were further classified into three subcategories: the comprehension of people, artifacts, and systems. The **comprehend** main area, with a total of 68 studies, primarily focuses on Understanding Software Engineering Phenomena. Within this, the first most studied

Table 3.5. Result of areas and topics.

Main-area & Subarea	Topic	Studies
-Comprehend(68)	—Understanding Software Engineering Phenomena	
People(38)	<i>Organizational management(4)</i> <i>Newcomers(14)</i> <i>Collaboration or teamwork(3)</i> <i>Communication(1)</i> <i>Community management(1)</i> <i>Personality(3)</i> <i>Expertise or experience(2)</i> <i>Roles(2)</i> <i>Motivation(4)</i> <i>Contribution inequality(3)</i> <i>Disengagement(1)</i>	[C004, C048, C054, C064] [C006, C007, C008, C010, C030, C047, C053, C058, C060, C071, C072, C077, C083, C099] [C015, C055, C074] [C019] [C022] [C024, C089, C092] [C032, C049] [C037, C070] [C038, C059, C084, C096] [C045, C046, C081] [C082]
Artifacts(9)	<i>Contribution Patterns(5)</i> <i>Communication(4)</i>	[C009, C018, C020, C034, C102] [C016, C026, C040, C043]
Systems(21)	<i>Project health(14)</i> <i>Project success(3)</i> <i>Ecosystem and project characterization(1)</i> <i>Project diversity(2)</i> <i>Migrations(1)</i>	[C017, C033, C035, C036, C039, C041, C042, C051, C052, C056, C065, C091, C095, C097] [C021, C057, C101] [C029] [C086, C088] [C093]
-Train(4)	—Teaching and training students	
	<i>Education assessment(3)</i> <i>Teaching courses(1)</i>	[C062, C068, C085] [C005]
-Construct(29)	—Creating models and artifacts	
Models(13)	<i>Recommendations systems(3)</i> <i>Prediction Models(10)</i>	[C011, C067, C078] [C001, C002, C012, C013, C014, C028, C031, C079, C087, C098]
Artifacts(16)	<i>Tools(11)</i> <i>Bots(5)</i>	[C023, C044, C050, C061, C063, C069, C073, C075, C076, C090, C100] [C025, C027, C066, C080, C094]
-Characterize(1)	—Describing contributions assessments	
	<i>Team or developer performance(1)</i>	[C003]

subarea is **People** with the most substantial sub-area, comprising 38 studies and covering topics from *Organizational management* to *disengagement*. The most mentioned topic in the subarea is how to assist newcomers with onboarding (14 studies). Aspects considered within this category include mentoring, barriers, and guidelines for onboarding. Noteworthy concentrations within this sub-area also include *organizational management* (4 studies) and *motivation* (4 studies) of developers to contribute to the project. Another studied topic was the *collaboration or teamwork, personality, and contribution inequality* of developers in projects (3 studies of each). Other less-mentioned topics include the developers' *expertise or experience* (2 studies), their *roles dynamics* (2 studies), *communication* (1 study), *disengagement* (1 study), and *community management* (1 study).

The second most studied subarea with 21 studies and 5 topics, is comprehending software **systems or projects**. Within this subarea, the most mentioned topic is the study of readme files of projects, onboarding programs, or sustainability factors of *project health* (14 studies). Other less-mentioned topics are *project success* (3 studies), *project diversity* (2 studies), *project characterization* (1 study), and *project migrations* (1 study).

The least studied subarea, with 9 studies and 2 topics, is comprehending **artifacts**. The most studied topics were *contribution patterns* (5 studies) and *communication* (4 studies). *contribution patterns* studied the developer's activities in the open-source such as pull request contributions, commit, and target projects. Meanwhile, *communication* analyzed participating in meetings, creating comments, replying to threads, interacting, and answering emails.

The second most mentioned main topic, found in 29 papers and 2 topics, is the utilization of contribution data to **construct** models, tools, recommendation systems, and bots. This category has two main subthemes: artifacts (16 studies) and models (13 studies). The most mentioned topic was constructing *tools* or programming environments (11 studies). These tools to capture toxicity capture, issue labeling, task assignment in OSS. Thus, this topic is within the artifact subtheme. The second most mentioned topic was *prediction models* to help the project handlers with long-time contributors identification (10 studies). Other mentioned topics within this category include the creation of *bots* (5 studies), and the construction of a *recommendations system* (3 studies).

The tertiary focus in the corpus of research, encompassing 4 studies across 2 topics, centered on the pedagogical strategies for **training and teaching** students in the field of software engineering. The predominant topic within this domain involved the *evaluation and assessment of educational projects in software engineering*, as documented in 3 studies. These investigations proposed, examined, and compared various methodologies for assessing the contributions of students and teams within Open Source Software (OSS) projects, with methodologies ranging from summer code programs to educational strategies and training tools. The secondary topic within this educational arena, featured in a single study is *teaching course*, including the adoption of novel technologies and mentoring processes. This particular study provided insights into the enhancement of student engagement and the facilitation of their integration into OSS projects.

Finally, the least mention area of study, with only 1 study is the **Characterize** main area consists of a single study that discusses describing contributions assessments in terms of (1 study) *Team or developer performance*.

Answering RQ1: Four principal thematic areas have been identified within the realm of software engineering research: the comprehension of software engineering phenomena, the training of students, the construction of models or artifacts, and the characterization of contribution assessments. The majority of studies have leveraged data on contributions to gain a deeper understanding of people and systems within the comprehension of software engineering phenomena, and to develop artifacts, with particular attention to the integration of newcomers, the evaluation of project health, and the creation of tools. Conversely, the characterization of contribution assessments has been less frequently addressed, with an emphasis on the evaluation of team or developer performance when such characterization occurs.

4 Threats To Validity

I now discuss threats to the validity of the mapping study.

External validity. External validity pertains to the generalizability of findings. This mapping study’s outcomes are interpreted within the context of the Software Engineering (SE) domain, and the legitimacy of the derived conclusions is confined exclusively to the SE sphere. The external validity threats are thus not applicable.

Construct validity. Construct validity addresses the extent to which the measurement techniques accurately represent the subject of the study. The risk of misclassification in areas such as contribution, methodology, and topic identification arises from the subjective nature inherent in the coding methodology during the qualitative analysis. To reduce this risk, a collaborative approach was employed where two co-authors engaged in a round-table discussion for the classification process. In instances of disagreement, a thorough review and discussion of the complete content of the papers were conducted until a consensus was achieved.

Internal validity. Internal validity concerns the credibility of inferences about cause-and-effect relationships. In this study, I identify three potential threats to internal validity. The primary threat pertains to the selection of papers during the screening phase. The substantial volume of results generated by the search string necessitated an initial filtering stage, where the first author screened and excluded papers based solely on their titles and abstracts. This approach introduces a potential bias in the selection process. However, I maintain confidence in mitigating this threat, considering the first author’s background as an experienced researcher in software engineering, equipped with domain-specific knowledge. The second threat stems from the selection of venues. The mapping study included 22 leading venues, selected based on their online citation indices, feedback from the software engineering community and google scholar ranking, in alignment with the approach used by Mathew et al. [60]. It is acknowledged that some venues may inevitably be omitted from this study. Nevertheless, it is the contention that these 22 top venues adequately encapsulate the best practices in SE research. The third potential internal threat concerns the terms employed in the search string. There is a possibility that the search string may not encompass all relevant terms. To mitigate this risk, an initial round of manual examination

was conducted on eleven papers related to Software Engineering (SE). This process was aimed at grouping potential term candidates. Following this preliminary assessment, I have gained confidence in the adequacy of the current search terms used in the study.

Conclusion validity. Conclusion validity refers to the extent to which the inferences drawn about correlations within the data are justifiable. In the datasets, there exists a concern regarding the accuracy of the groupings. The absence of comparable studies with analogous results precludes the possibility of corroborating the findings. To address this issue, I have adhered to systematic guidelines to underpin the validity of the outcomes. Since the scope of the results is confined to developer contributions within the field of software engineering, concerns about generalizability are not relevant, as the conclusions are specific to the domain studied. Additionally, there were a constraints on the time frame of the studies included in the research.

5 Related study

5.1 Studies on Human Aspect of Software Engineering

Contributors. Software development could not be separated from users' participation in a forum. Their contributions are not always related to writing code. A number of studies on the contributors in communication channels, has shown that experienced contributors and newcomers play an important role in developing software.

Senior contributors. Recent studies show that every individual has an opportunity to become a valuable contributor. Mockus [65] built a model to analyze the users' chances of becoming a senior contributor depend on her competence, passion, and first-time contribution opportunity. Zhou and Mockus [110] found that the participation of new members in the issue tracking system environment might impact their status of becoming a long-term contributor.

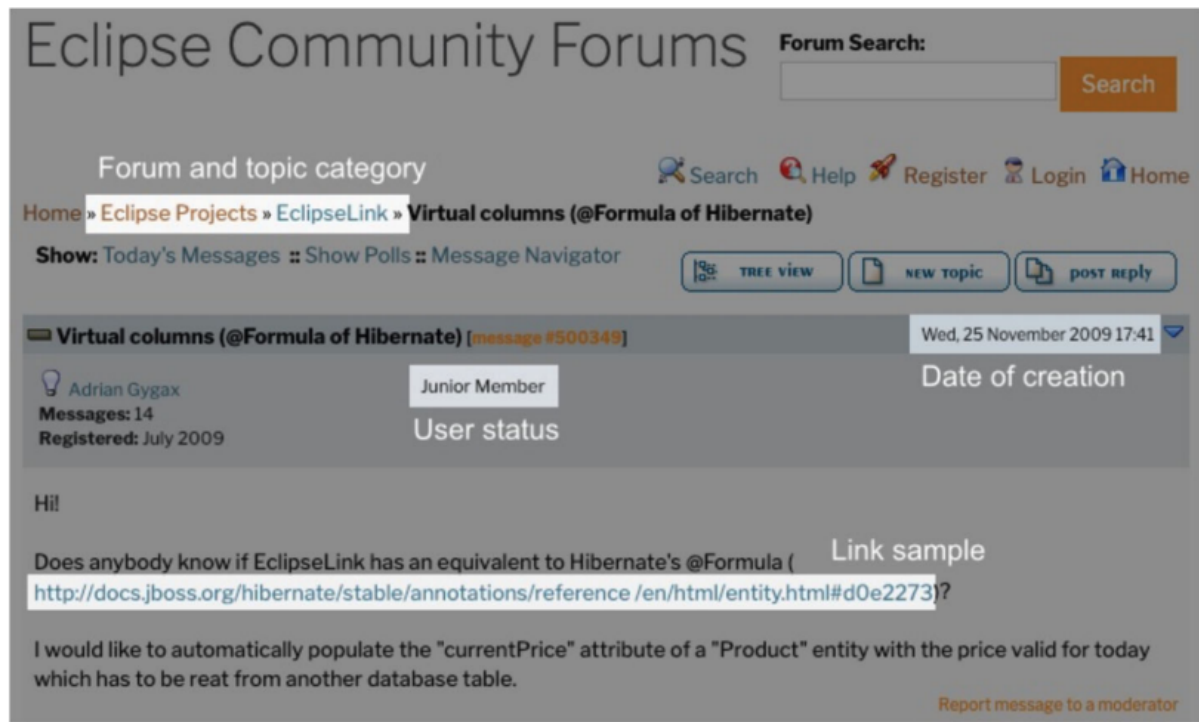


Figure 3.5. Example of a forum thread.

Junior contributors. Despite most of the valuable information for improvement to software quality comes from the experienced members in a software project [55], software developers should not underestimate the newcomers' contributions in a discussion forum. Steinmacher et al. [93] identified that the lack of social interaction with the community, having unanswered questions or receiving delayed answers, and their technical experience backgrounds are some difficulties that new members faced when they make contributions to an open source software project. Middleton et al. [64] also studied the contribution characteristics of new members in OSS projects. The authors identified that the participation forms of the new members, such as pull requests and how they comment in the discussion influence the decision to join OSS teams.

Table 3.6. Outputs of the pre-processing of the forum dataset

Step	# Threads
Step 1: Raw extraction	1,097,174
Step 2: Remove duplication	832,058
Step 3: Separation:	
(i) Threads by webmaster	542,997
(ii) Threads by non-webmaster users	289,061

5.2 How are project-specific forums utilized? A study of participation, content, and sentiment in the Eclipse ecosystem

Membership Classification. In this study, I describe the techniques to classify the membership of users for each posted message. Unlike the other question and answer online forums such as Stack Overflow, in the Eclipse community forum, all registered users are assigned into three statuses of membership, that are, (1) Junior, (2) Member, and (3) Senior. These user statuses are included in the collected data resulting from Step 3 in Table 3.6 which can be seen in every post of a user, as shown in Fig. 3.5. The status of each user may change from the lowest level (i.e. Junior) into the highest one (i.e. Senior) depending on the contributions of the user in the community. However, in the forum, I could not differentiate which posts were posted by users when they were a Junior, Member or Senior. This is because once the status of a user has changed, it will replace the old status in all posts of a user with the latest status, including their first posts. Furthermore, I also did not find any information about the time when the status of a user changed.

To define the member status of each registered user, I attempted to calculate the total number of posts of every user. From this amount of posts, I summarized the quantity of posts per author based on the user identity number. The total number of posts per author varies, from less than ten to more than one thousand posts. In this step, I found the maximum number of posts of each user if I consider the latest status of users as collected in the dataset, as shown in Fig. 3.6. The maximum number of posts by Juniors and Members are 29 and 106 respectively,

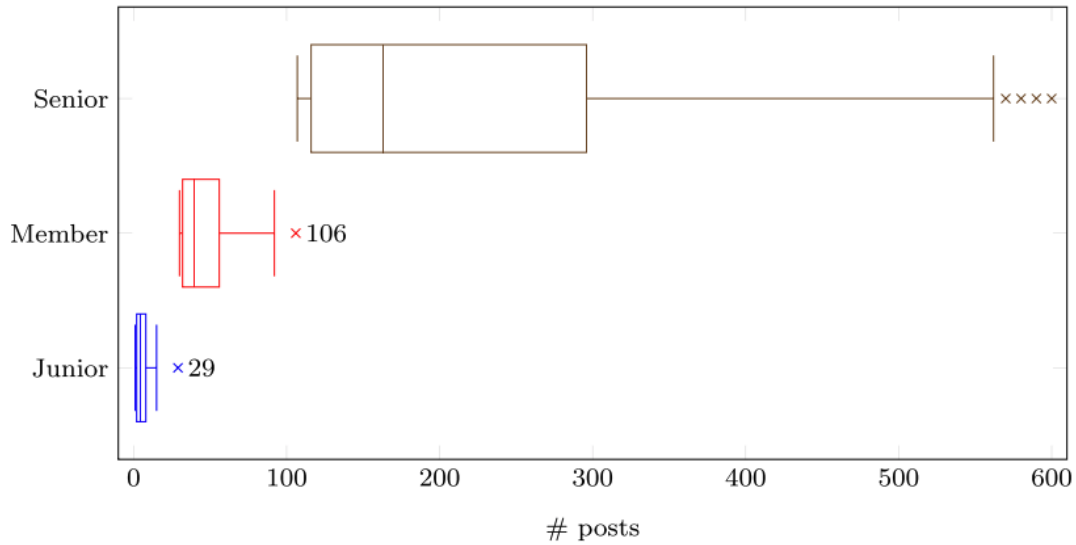


Figure 3.6. Frequency of posts per user status. The maximum number of posts for each type of users is used to define the threshold of post-based membership. The threshold for Juniors and Members are 29 and 106, respectively. Although Seniors have posted more than 28 thousand posts, I limit up to 600 in the figure.

while the maximum number of posts by Seniors is 28,476. Based on this finding, I used these maximum numbers as the thresholds to differentiate the user status for each post based on the sequence number of a post. The sequence number of a post depends on its creation date in order. The earliest post is assigned as the first post, then followed by the other posts ordered by date of creation.

6 Conclusion

Onboarding within the realm of Software Engineering (SE) is recognized as a critical practice, significantly influencing both open source software (OSS) and proprietary software projects. Over the past decade, advancements in onboarding tools and methodologies have facilitated more efficient integration processes, gaining widespread acceptance in both OSS and non-OSS sectors. This work presents what I believe to be the inaugural systematic mapping study in this area, aimed at delineating the landscape of topics addressed across 102 software

engineering studies that are published in premium conferences and journals. The investigation encompasses a comprehensive analysis of research topics, contexts examined, potential validity threats, and evaluation challenges encountered in these studies. The visualization comprises concerning topics addressed, I identify that the majority of studies have leveraged data on contributions to gain a deeper understanding of people (38 studies) and systems (21 studies) within comprehension of the software engineering process. Conversely, the characterization of contribution assessments has been less frequently addressed, with an emphasis on the evaluation of team or developer performance when such characterization occurs. The study reveals that the domain of developer onboarding is extensive and characterized by a diversity of studies. However, there remains ample scope for further research to conduct studies on the following gaps:

- **Limited Understanding of Newcomer pre-onboarding activities.** While there is substantial literature on newcomers after onboarding OSS but there is a notable gap in understanding how to retain newcomers on the social coding platform by analyzing and characterizing their activities before onboarding. This is crucial for the sustained success and vibrancy of potential OSS communities.
- **Impact of Newcomer Social Dynamics.** The role of social dynamics and collaborative nature relationships among newcomers towards OSS joining process is underexplored. Understanding how social factors influence a newcomer's integration into OSS projects could provide valuable insights.
- **Comparative Studies Across Different OSS Platforms.** Comparative studies between different OSS platforms (e.g., GitHub vs. GitLab) in the context of onboarding OSS are scarce. Such comparative analysis could reveal unique challenges and opportunities inherent to different platforms.
- **Longitudinal Studies.** There is a need for more longitudinal studies to understand how onboarding processes evolve over time and how changes in OSS communities impact these processes.

By addressing these gaps, future research can contribute significantly to the body of knowledge in OSS onboarding and help in developing more effective and

inclusive strategies for integrating newcomers into OSS projects.

4 | Newcomer OSS-Candidates: Characterizing Contributions of Novice Developers to GitHub

In the previous chapter, through the mapping study, the literature reviews reveal the key topics offering insights into current trends and gaps in developer onboarding and motivate this thesis. Hence, regarding filling out the gaps concerning the limited understanding of newcomer pre-onboarding activities and the impact of their social dynamics, in this chapter, I endeavor to conduct an empirical study that is broken into three parts to gain a deeper understanding of (i) identifying Newcomer-OSS Candidates, (ii) validating their pre-onboarding activities, and characterizing them through mixed methods approach, and finally (iii) analyzing the proportion of them who are eventually onboard to OSS projects in GitHub.

1 Introduction

The success of Open Source Software (OSS) has always been based on the continuous influx of newcomers and their active involvement [72]. Previous studies have shown evidence that many contemporary projects are at risk of failure, with one of the reasons being the inability to attract and retain newcomers [34, 104]. For example, Coelho and Valente [20] proposed two strategies that include newcomers which aim to transfer the project to new maintainers and to accept new core developers. In another study, Steinmacher et al. [92] presented a model that

analyzes the influential forces to newcomers being drawn or pushed away from a project.

In contrast, the rise of social coding platforms has led to an explosion of potential developers. GitHub reported¹ around 10 million-plus new users in 2020 and allows over 40 million developers to showcase their skills to the world’s largest community (44 million upstream repositories). With this upsurge in user activity, However, the extent to which these developers’ activities prior to onboarding to OSS projects is unknown. Despite the known increase in the number of potential OSS contributors, there is a notable lack of understanding regarding the activities and experiences of these individuals before they join OSS projects. This gap is crucial because pre-onboarding experiences can significantly shape a newcomer’s ability to contribute effectively to OSS projects.

There are several motivations need to understanding the activities of novice developers before they join OSS projects. For instance, *skills and knowledge assessment*: understanding the prior experiences of new OSS contributors could help in assessing the skills and knowledge they bring, which is vital for effective integration into projects. Another reason *tailored onboarding processes*: knowledge of pre-onboarding activities could inform the development of more personalized and efficient onboarding processes, catering to the varied backgrounds of new developers. Other motivation is *retention strategies*: insights into the motivations and prior engagements of newcomers can aid in devising strategies to not only attract but also retain these contributors in OSS projects.

The term newcomer has usually been used in a loose way in literature [92]. Inspired by the incubation of OSS projects on GitHub, I coin the term “Newcomer OSS-Candidate”, *who is not yet a newcomer, but has potential to become one*. Concretely, I define a Newcomer OSS-Candidate as a developer that satisfies these three criteria: 1) is a developer that does not have any prior experience contributing to an OSS project, 2) is a new user to a social coding platform, and 3) has the intention to onboard an OSS project hosted on a social coding platform. Although there is a complete body of work that has studied the barriers and struggles of newcomers [90, 92], none has explored the activities prior to onboarding to OSS projects of Newcomer OSS-Candidates. Most of the work

¹Statistics from <https://octoverse.github.com> accessed January 2020

revolves around newcomers who have already onboarded to OSS projects.

Hence, this empirical study which is broken into three parts using GitHub as a case platform to gain a deeper understanding of (i) identifying Newcomer-OSS Candidates, (ii) validating their pre-onboarding activities and characterizing them through a mixed method approach, and finally (iii) analyzing the proportion of them who are eventually onboard to OSS projects in GitHub. I studied 171 Newcomer OSS-Candidates and their GitHub repositories, first commits and Pull Requests (PRs) guided by four research questions:

- **(RQ1.) What kinds of repositories does a Newcomer OSS-Candidate target?**

Motivation: Kalliamvakou et al. [45] showed that most repositories hosted on GitHub are non-software. However, since Newcomer OSS-Candidates have the intention to later onboard a software project, I would like to test the assumption that *(H1) Newcomer OSS-Candidates are more likely to target software repositories*. Since GitHub users can either create their own upstream repositories or fork existing repositories, I compare these two kinds of repositories.

Result: I observe that 66% of Newcomer OSS-Candidates target software based repositories. The statistical test indicates that hypothesis *H1* is established. Furthermore, Experimental and Documentation are the most frequently targeted software repository kinds for fork and upstream repositories, i.e., 24% and 21%, respectively.

- **(RQ2.) What are the kinds of first contributions that come from Newcomer OSS-Candidates?**

Motivation: Hattori and Lanza [42] showed that OSS projects constantly add new content to software (i.e., development) more frequently than maintaining existing code. Hence, for this RQ, my motivation is to understand whether or not Newcomer OSS-Candidates are more likely to add new content or maintain the repository. Hence, by studying these two types of con-

tributions, I test the hypothesis that *(H2) Contributions to GitHub repositories from Newcomer OSS-Candidates are more likely to do development activities*. I analyze two kinds of GitHub contributions, either a direct contribution through a commit, or a submitted Pull Request (PR).

Result: For the first commit contributions, I find that 74% of contributions from Newcomer OSS-Candidates are related to development activities. For the first PR contributions, results show that 60% of contributions are associated with management activities. The statistical tests confirm that hypothesis *H2* is established in first commit contributions, while is not established in first PR contributions.

- **(RQ3.) To what extent do Newcomer OSS-Candidates practice social coding with their first contributions?**

Motivation: Since GitHub is a social coding platform, I would like to explore the extent to which a Newcomer OSS-Candidate is likely to make a social contribution as their first contribution. Specifically, I analyze whether or not a Newcomer OSS-Candidate shares code, which is measured by single or multiple authorship on a file. Hence, similar to RQ3, I explore the commit and PR contributions to test the hypothesis *(H3) Newcomer OSS-Candidates are more likely to contribute to a file with multiple authorship*.

Result: Results show that after joining GitHub, a majority of Newcomer OSS-Candidates (i.e., 73% of first commits and 59% of PRs) do not share code with other authors. Moreover, the statistical tests validate that hypothesis *H3* is not established for both first commit and first PR contributions.

- **(RQ4.) What is the proportion of Newcomer OSS-Candidates that eventually onboard an OSS project?**

Motivation: In accordance with my definition, I explore the extent to which these Newcomer OSS-Candidates eventually onboard an OSS project.

I would like to explore the proportion of Newcomer OSS-Candidates who eventually onboard an OSS project. Additionally, I validate what kinds of barriers that Newcomer OSS-Candidates face when onboarding OSS repositories.

Result: Quantitative analysis shows that 30% of Newcomer OSS-Candidates eventually onboarded engineered OSS repositories. Complementary, a follow-up user survey shows that 70% of studied participants ended up making contributions to an OSS repository. Newcomer OSS-Candidates strongly agreed that they face the barrier of *finding a way to start*, while *social interaction* received the most mixed responses as a barrier.

The remainder of this paper is organized as follows: Section 2 describes the identification procedure for Newcomer OSS-Candidates through survey. Section 3 reports the validating pre-onboarding activities and characterizes them through the mixed method approach of my empirical study, while Section 4 analyzes the proportion of onboarding to OSS projects in GitHub, Section 5 discusses the lessons learned and study implications. Section 6 discloses the threats to validity, Section 7 presents related work and finally, I conclude the paper in Section 8. To facilitate replication and future work in the area, I have prepared a replication package, which includes the studied 171 Newcomer OSS candidates' repositories, manually labeled datasets, the scripts for the quantitative analyses, and the survey materials. The package is available online at <https://github.com/NAIST-SE/NewcomerCandidate>.

2 Identifying Newcomer OSS-Candidates through Survey

Approach. In this section, I describe the process of identifying Newcomer OSS-Candidates. I used the first-contribution community² in GitHub as data source for collecting Newcomer OSS-Candidates. The community is an initiative established

²<https://github.com/firstcontributions/first-contributions/blob/master/Contributors.md>

Table 4.1. Survey Questions sent to potential respondents

Survey Questions for Newcomer OSS-Candidate
Q1) What is your motivation to make a contribution to GitHub? (a) Learning to Code. (b) Assignment or Experiment Project. (c) Intend to contribute to an Open Source. (d) Use to showcase my programming skills. (e) Others.
Q2) Did you have prior experience contributing to an OSS before GitHub? (Yes/No)

to help beginners make their first contributions on GitHub and currently has over 5,000 plus contributors, over 39.7 thousand forks, and over 21 thousand stars as of October 2021. To extract the survey respondent candidates, I used command `"git log -pretty=format:%ae"`³ on Contributors.md file provided by the community and were able to get 17,507 respondent candidates. I sent an online survey invitation⁴ to reach up to 4,000 respondent candidates through email and a slack channel.⁵ Survey was open from March 3, 2020 to March 31, 2020 (around a four-week period). I received 208 responses, allowing us to mine their repositories and contributions by providing their GitHub IDs. In the survey, I validate the definition of Newcomer OSS-Candidate by asking two questions. The two questions are presented in Table 4.1. Besides, respondents were also asked about their interests, and their perception rank of their programming skills.



171 Identified Newcomer OSS-Candidates. Table 4.2 presents the survey answers that are related to prior OSS experience of respondents and their motivations to contribute. Table 4.2b shows that 82% of respondents (i.e., 171 responses) intend to contribute to an OSS project. Furthermore, these respondents claim that they have not had any prior OSS experience. Henceforth, I define a *Newcomer OSS-Candidate* as a developer that does not have any prior

³<https://git-scm.com/docs/pretty-formats>






⁴<https://tinyurl.com/r7acxvn>

⁵<https://firstcontributions.slack.com/>

Table 4.2. Two questions in survey

Have you had any prior OSS experience?	Percent	
No	85%	
Yes	15%	

(a) Answers to Q1 of the survey

What is the motivation to contribute?	Percent	
(a) Learning to Code.	58%	
(b) Assignment or Experiment Project.	21%	
(c) Intend to contribute to an Open Source.	82%	
(d) Use to showcase my programming skills.	42%	
(e) Others	5%	

(b) Answers to Q2 of the survey

experience contributing to an OSS project, is a new user to a social coding platform, and has the intention to onboard an OSS project hosted on a social coding platform. According to the definition of Newcomer OSS-Candidate, I used these 171 participants to further track their repositories and contributions for subsequent analyses.

3 Validating Pre-Onboarding Activities and Characterize them through Mixed Method Approach

To answer research questions, each research question comprises the approach and their results.

3.1 (RQ1) What kinds of repositories does a Newcomer OSS-Candidate target?

Approach. To answer RQ1, I first construct the *(D1) Newcomer OSS-Candidate Repository Dataset*, which is a mapping of selected Newcomer OSS-Candidate infor-

mation (as described in Section 2) with their GitHub repository contributions. Using the GitHub REST API (GitHub, 2020) and the credentials of the 171 survey participants, I retrieved 2,392 unique contributed repositories, consisting of 936 fork⁶ and 1,456 upstream⁷ repositories. Under the guidance of Borges et al. [12], Kalliamvakou et al. [45], I classify the repositories into software and non-software. The definitions of software and non-software repositories are described below:

- *(Software) Application Software:* systems that provide functionalities to end-users, like browsers and text editors.
- *(Software) System Software:* systems that provide services and infrastructure to other systems, like operating systems, middleware, servers, and databases.
- *(Software) Web libraries and frameworks.*
- *(Software) Non-web libraries and frameworks.*
- *(Software) Software tools:* systems that support software development tasks, like IDEs, package managers, and compilers.
- *(Software) Documentation:* repositories with documentation, tutorials, source code examples.
- *(Software) Experimental:* repositories include demos, samples, test code, and tutorial examples.
- *(Non-Software) Storage:* category includes repositories documents and files for personal use, such as presentation slides, resumes, e-books, music files etc.
- *(Non-Software) Academic:* class and university research projects come under this category.
- *(Non-Software) Web:* under this category I classify websites and blogs.

⁶<https://docs.github.com/en/get-started/quickstart/fork-a-repo>

⁷<https://docs.github.com/en/get-started/quickstart/github-glossary#upstream>

Table 4.3. Proportion of software and non-software repositories targeted by Newcomer OSS-Candidates. Around 66% of Newcomer OSS-Candidates target Software repositories.

Category	Percent (%)	Fork & Upstream (%)
Software	66	Upstream (52)
		Fork (48)
Non-Software	24	Upstream (55)
		Fork (45)
Others	10	-
		-

- *(Others) No longer accessible/Empty*: repositories that gave 404 error, containing only a license file, a gitignore file, a README file, or no files at all were placed under this category.

I use a qualitative method to manually classify the different kinds of repositories. Following the protocol, with a confidence level of 95% and a confidence interval of 5⁸, I draw a statistically representative sample from *(DI)* to end up with 273 fork repositories and 304 upstream repositories. To evaluate the validity of manual coding, I randomly selected 30 repositories from the representative sample, and then the first three authors independently coded these repositories. The three authors then measured the inter-rater agreement using Cohen’s Kappa [106] as the measure of agreement. In the end, the Kappa agreement for fork repositories was nearly perfect (i.e., 0.91), while the score for upstream repositories was substantial (i.e., 0.76). Based on this encouraging result, the first author then completed the manual coding for the rest of the representative sample.

For significance testing, I validate hypothesis *(H1) Newcomer OSS-Candidates are more likely to target software repositories*, using the one proportion Z-test [73] as it compares an observed proportion to a theoretical one when the categories are binary.

Proportion of Software and Non-Software Repositories. Table 4.3 shows the proportion of software and non-software based repositories that New-

⁸<https://www.surveysystem.com/sscalc.htm>

comer OSS-Candidates target. I see that 66% of Newcomer OSS-Candidates target repositories are software based and follow sound software engineering practices in each dimensions. Furthermore, Newcomer OSS-Candidates are less likely to target non-software based repositories, accounting for 24%. Specifically, I observe that 10% of repositories are classified as Others. Through the manual analysis, these repositories are either “No longer accessible” or “Empty”. Upon in-depth analysis of repositories (i.e., Fork and Upstream), I observe that the dominant repositories for software and non-software are upstream i.e., 52% and 55%.

Frequency of Contributed Repository Kinds. Figure 4.1 shows that Documentation (21%), Experimental (15%), Web-based-applications, libraries and frameworks (15%) are the most frequently targeted upstream software repositories kinds. The other kinds of repositories that Newcomer OSS-Candidates frequently target are Academic (12%), Web (11%), and Application Software (9%). On the other side, I find that Experimental (24%) and Web-based-application, libraries, and frameworks (17%) are the most commonly targeted fork repositories kinds. The other kinds of fork repositories commonly targeted are Documentation (13%) and Academic (12%).

The statistical test validates a significant difference between the proportion of software and non-software repositories that Newcomer OSS-Candidates target, with a p-value < 0.001 . The result indicates that my proposed hypothesis, i.e., *(H1) Newcomer OSS-Candidates are more likely to target software repositories*, is established.

RQ1 Summary: Results show that 66% of Newcomer OSS-Candidates target software based repositories. The proposed hypothesis that *(H1) Newcomer OSS-Candidates are more likely to target software repositories* is established. Furthermore, Experimental and Documentation are the most frequently targeted software repository kinds for both fork and upstream repositories with 24% and 21%, respectively.

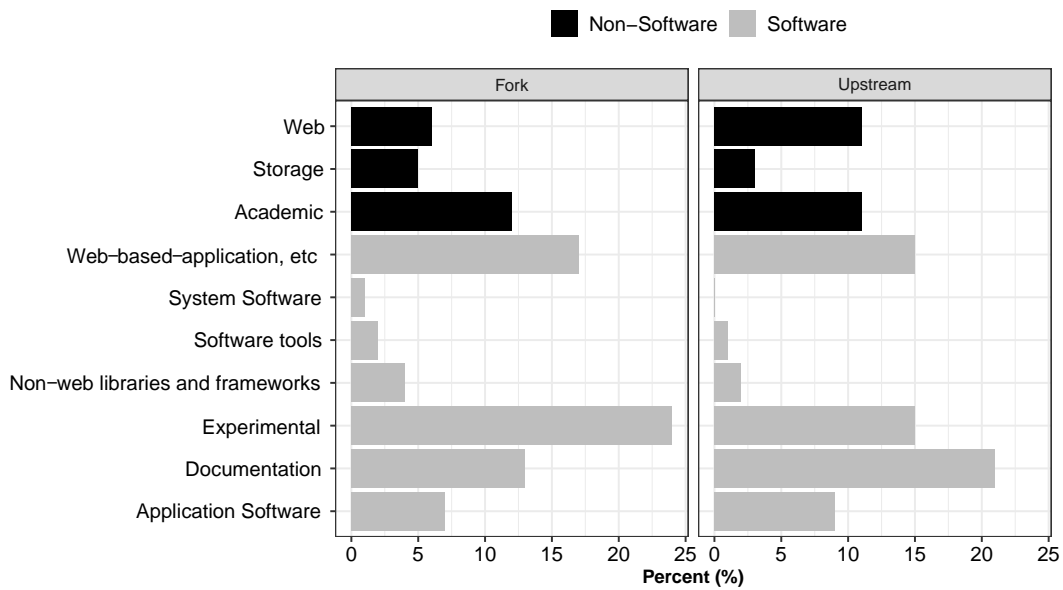


Figure 4.1. Frequency for contributed repository kinds with Fork and Upstream. Experimental and Documentation are the most frequently targeted software repository kinds, i.e., 24% and 21%, respectively.

3.2 (RQ2) What are the kinds of first Contributions that come from Newcomer OSS-Candidates?

Approach. To answer RQ2, I analyze the first contributions with two types, i.e., first commit and first PR. As such, I constructed a new dataset from RQ1, which is *(D2) First Contribution Dataset*. To do so, I first obtain the earliest GitHub repositories each of the 171 Newcomer OSS-Candidates. For the quality purpose, I ignore any *test and not meaningful* commits by filtering out experimental repositories that have been identified in RQ1. Furthermore, from the initial list of 171 participants, I remove another five participants. Three participants had not made any contributions to their fork or upstream repositories, and another two participants had become inactive since the initial survey. Hence, I ended up with a total of 166 first commits and 97 PRs from 166 Newcomer OSS-Candidates. I then classify the contributions according to Hattori and Lanza [42]:

- *Development (forward engineering and non-software)*: based on the forward-engineering type proposed by Hattori and Lanza (2008), the development activities relate to incorporation of new features and implementation of new requirements for both software and non-software. Examples of development for non-software repositories include adding new content for websites or documentation.
- *Repository Initializing (sub-category of development)*: derived from the forward-engineering category, I identify any first commits as the initializing commits to a new repository.
- *Re-engineering*: maintenance activities are related to refactoring, redesign and other actions to enhance the quality of the code without properly adding new features.
- *Corrective Engineering*: maintenance activities handle defects, errors and bugs in the software.
- *Management*: maintenance activities are those unrelated to codification, such as formatting code, cleaning up, and updating documentation.

To validate the understanding of the taxonomy of contribution kinds, I randomly selected 30 contributions of first commits and PRs, and then the first three authors independently coded these contributions, similar to RQ1. Since Hattori and Lanza [42] used a set of keywords, I applied the keywords as an initial guide. However, when deciding the classification, I consider the commit and PR attributes (i.e., title, message, and description) to have a better understanding of the context. Similar to RQ1, I use Cohen’s Kappa. The Kappa agreement scores for classifying contribution kinds of first commits and PRs were both substantial (i.e., 0.72 and 0.79, respectively). After the agreement measurement, the first author then completed the remaining sample.

To validate the hypothesis (*H2*) *Contributions to GitHub repositories from Newcomer OSS-Candidates are more likely to do development activities*, similar to RQ1, I use the one proportion Z-test [73]. To fit the formula of the statistical test, I merge *Development* and *Repository Initializing* into the *Development* category, and I merge *Re-engineering*, *Corrective Engineering*, and *Management* into the *Maintenance* category.

Frequency of Contribution’s Kinds. Table 4.4 depicts the distribution for kinds of contributions made by Newcomer OSS-Candidates. For the first commit contributions, as shown in the table, 31% and 43% of Newcomer OSS-Candidates engage in development activities and repository initializing activities in the first commits. The result suggests that Newcomer OSS-Candidates are more likely to engage in development activities (i.e., $31\% + 43\% = 74\%$) when submitting first commits. Upon closer inspection, I find that 98% and 77% of development activities and repository initializing activities involve code related changes. For the first PR contributions, the manual classification shows that 60% of Newcomer OSS-Candidates engage in management activities when submitting their PRs, indicating that Newcomer OSS-Candidates are more likely to target maintenance activities. Furthermore, I find that 45% of management activities are related to formatting code, and 55% are associated with cleaning up and updating documentation. More specifically, 4% of their first commits and 4% of first PRs contributions are classified as Others. Through the manual analysis, I find that these contributions are inaccessible (i.e., 404 errors), not be classified

Table 4.4. Frequency for Contribution’s Kinds of Newcomer OSS-Candidates. In the first commits, 43% of Newcomer OSS-Candidates are typically engaged in repository initializing activities, and 60% are engaged in the management activities of the PRs.

First Contributions	Kinds	Percent (%)	Code (%)	Doc (%)
First Commit :	Development	31	98	2
	Repository Initializing	43	77	23
	Re-engineering	7	100	0
	Corrective Engineering	2	100	0
	Management	13	5	95
	Others	4	100	0
sum		100		
Pull Request :	Development	9	89	11
	Repository Initializing	3	33	67
	Re-engineering	17	76	24
	Corrective Engineering	6	100	0
	Management	60	45	55
	Others	4	100	0
sum		100		

into any category based on the taxonomy, or not written in English.

The statistical tests confirm statistically significant differences between the proportion of development and maintenance activities for both types of contributions (first commit and PR), with a p-value < 0.001 . For the type of first commit contributions, the test result validates that Newcomer OSS-Candidates are more likely to engage in development activities. However, for the type of first PR contributions, the test result confirms that Newcomer OSS-Candidates are more likely to be involved in maintenance activities. To conclude, the raised hypothesis, (*H2*) *Contributions to GitHub repositories from Newcomer OSS-Candidates are more likely to do development activities*, is established in first commit contributions,

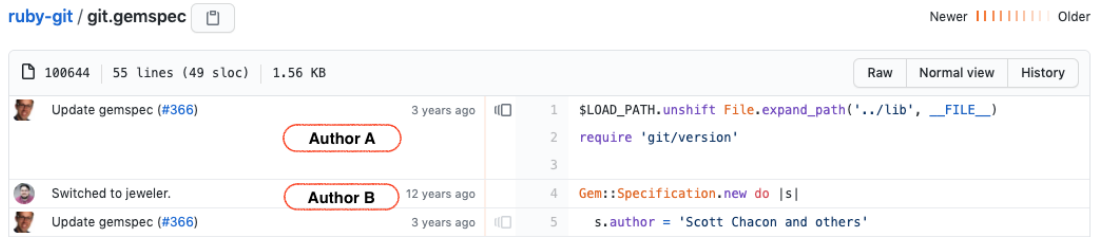


Figure 4.2. An example of how I define developers practice social coding, where more than one author contributes to the git.gemspec file.

while it is not established in first PR contributions.

RQ2 Summary: For the first commit contributions, I find that 74% of contributions from Newcomer OSS-Candidates are related to development activities. For the first PR contributions, the results show that 60% of the contributions are associated with management activities. Furthermore, statistical tests confirm that *(H2) Contributions to GitHub repositories from Newcomer OSS-Candidates are more likely to do development activities* is established in first commit contributions, but it is not established in first PR contributions.

3.3 (RQ3) To what extent do Newcomer OSS-Candidates practice social coding with their first contributions?

* Social Coding: in Terms of Multiple Authorship

Approach. Social coding is a very loose term Dabbish et al. [23] used to describe the ability for developers to advertise (openly share and allow modification) their code on social platforms such as GitHub. In the paper, as shown in Figure 4.2, I select one social coding practice in terms of multiple authorship to analyze where a contributor modifies either someone else’s codes or others may modify this contributor’s codes in the future. In the example, there are two authors (i.e., author A for lines 1–3 and author B for line 4) that contribute to a single file (i.e., git.gemspec) in a repository (i.e., ruby-git). To do so, I use the *D2* dataset from RQ2, which contains first commit and first PR contributions. I identify

Algorithm 1 Identify social coding in terms of whether a contribution is modified by a single author or multiple authors.

Input : First *Commit/PR* performed by an author *au*
Output : Contribution type of First *Commit/PR*: single or multiple authors

```
1  $F \leftarrow$  A set of files modified by First Commit/PR;  
2  $Type(F) =$  single author;  
3 for  $f \in F$  do  
4    $D \leftarrow extract\_authors(git-blame(f));$   
5   if  $au \in D$  &  $|D| > 1$  then  
6      $Type(f) =$  multiple author;  
7   end  
8 end  
9 return  $Type$ ;
```

Figure 4.3. Identify social coding in terms of whether a contribution is modified by a single author or multiple authors.

social coding using Algorithm 1 and the `git-blame`⁹ command on each contained file in the commit to check whether the files receive changes from more than one author (lines 3–4 in Algorithm 1). Considering that one PR may include multiple commits, I analyze all commits inside each PR with Algorithm 1. Specifically, I found that 21 out of 97 PRs (22%) have multiple commits.

To validate the hypothesis (*H3*) *Newcomer OSS-Candidates are more likely to contribute to a file with multiple authorship*. Similar to RQ1, I use the one proportion Z-test [73].

Social coding (Multiple Authorship). Table 4.5 presents the frequency of social and non-social contributions in terms of authorship done by Newcomer OSS-Candidates. As shown in the table, the majority of Newcomer OSS-Candidates do not practice social coding after joining GitHub. For instance, I find that 73% of the first commits and 59% of the first PRs are contributed by a single author. Such results suggest that Newcomer OSS-Candidates are less likely to practice social coding in terms of sharing multiple authorship, when placing their first GitHub contributions.

The statistical test validates that for the first commits, there is a statistically significant difference between the proportion of social and non-social contribu-

⁹<https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-blame>

Table 4.5. Frequency of social and non-social contributions from Newcomer OSS-Candidates in terms of single/multiple authorship. After joining GitHub, 73% and 59% of Newcomer OSS-Candidates have non-social based contributions in their first commits and PRs.

Social coding practice (First Commit)	Percent (%)	
multiple	27	■
single	73	■
Social coding practice (Pull Request)	Percent (%)	
multiple	41	■
single	59	■

tions, with a p-value < 0.001 , where Newcomer OSS-Candidates are likely to practice non-social coding. For the first PRs, there are no statistically significant difference, with a p-value > 0.05 . To conclude, the proposed hypothesis ($H3$) *Newcomer OSS-Candidates are more likely to contribute to a file with multiple authorship*, is not established in both first commits and PRs.

RQ3 Summary: The results show that after joining GitHub, a majority of Newcomer OSS-Candidates (i.e., 73% of first commits and 59% of PRs) do not share code with other authors. Furthermore, statistical tests validate that ($H3$) *Newcomer OSS-Candidates are more likely to contribute to a file with multiple authorship*, is not established in both first commit and PR contributions.

4 Analyzing Proportion of Onboarding to OSS Projects in GitHub

To answer the research question, it comprises the approach and results.

4.1 (RQ4) What is the proportion of Newcomer OSS-Candidates that eventually onboard an OSS project?

Approach. To answer RQ4, I perform both quantitative and qualitative analyses. I find that making contributions to an OSS project is not trivial, and involves a process that follows two steps:

- *Fork an OSS repository.* The first step for any Newcomer OSS-Candidate is to fork an OSS repository. Hence, I extracted 936 fork repositories out of a total of 2,392 repositories from the *D1* dataset. Then, to identify whether this repository is an engineered software project, I matched each fork repository against a curated dataset by Munaiah et al. [67].
- *Identify contributions.* During step one, I found that many participants who only fork the repository, without contributing back to either the fork or upstream repository. Hence, I performed an in-depth analysis through two particular ways of onboarding i.e., either the fork or upstream repositories.

For the qualitative analysis, I conducted a follow-up survey¹⁰ to acquire the perception of the participants. I sent an online survey invitation to Newcomer OSS-Candidates through emails and ended up receiving 27 responses. The survey is split into two questions, confirming whether participants had contributed to an OSS repository. The first question is related to whether the participant had onboarded an OSS project (i.e., Since joining GitHub, did you successfully make a contribution to any Open Source Software project?). In the second question, I explore the barriers faced by OSS newcomers [92]. Hence, I asked participants to rate each barrier (i.e., Social Interaction, Newcomer Previous Knowledge, Finding a Way to Start, Technical Hurdles, and Documentation) on a five-point Likert scale.

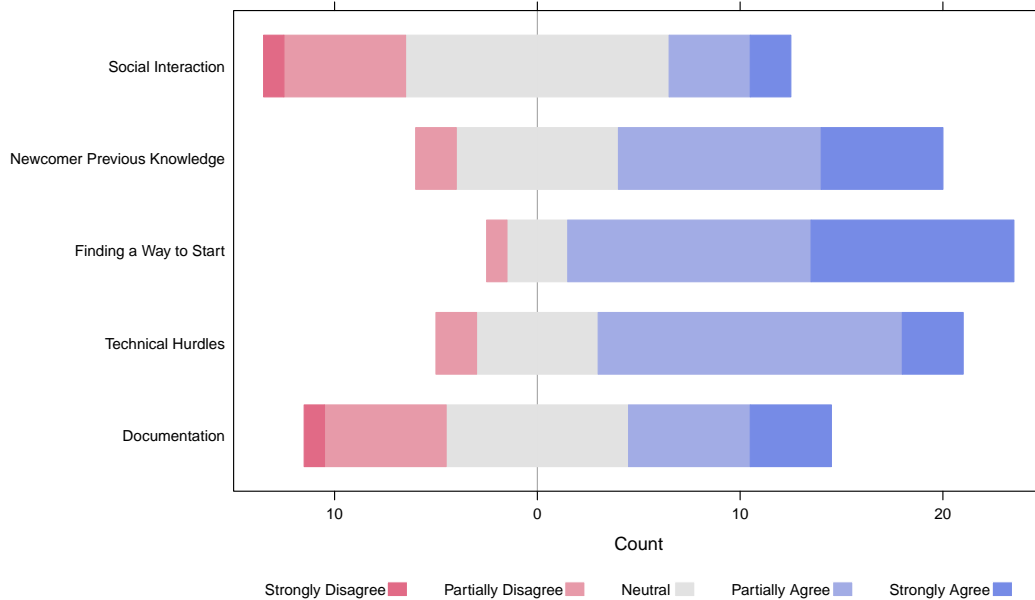
Onboarding Process in GitHub. Table 4.6 presents the distribution of how Newcomer OSS-Candidates onboard OSS projects in terms of the quantitative analysis. I show that 49% of Newcomer OSS-Candidates onboard OSS projects, while 51% do not. Furthermore, 51% of Newcomer OSS-Candidates only

¹⁰Survey details are available at <https://forms.gle/JQiVamovUXdJiy8z5>

Figure 4.4. Qualitative analysis using a follow-up survey to acquire the perception of Newcomer OSS-Candidates.

Response to “Since joining GitHub, did you successfully make a contribution to any OSS project?”	Count (#)	Percent (%)
Has made a contribution to an OSS project.	19	70%
Has never made a contribution to an OSS project.	8	30%
sum	27	100

(a) Answers to Q1 in the follow-up survey.



(b) Barriers faced by Newcomer OSS-Candidates. Most Newcomer OSS-Candidates (i.e., 22 out of 27 responses) strongly agree that *finding a way to start* is a barrier.

Table 4.6. Frequency of Newcomer OSS-Candidates that started the onboarding process for OSS repositories.

Match to the Munaiah(2016) dataset	Onboarding Steps	Count (#)	Percent (%)
Started Onboarding Process :		81	49
	Fork an OSS repository (51%)		
	Contribute to fork OSS repository (22%)		
Eventually Onboarded:	Contribute to original OSS repository (30%)		
Not Onboard:		85	51
Sum		166	100

fork the OSS repositories not making any contributions (Fork an OSS repository), and 22% have contributed in the form of making commits to their own fork OSS repositories (Contributed to fork OSS repository). Meanwhile, 30% of Newcomer OSS-Candidates eventually onboard by submitting PRs directly to the original OSS repositories (Contributed to original OSS repository). On the other hand, for the qualitative analysis, the survey results show that 19 out of 27 Newcomer OSS-Candidates (70%) claim that they have made contributions to OSS repositories. Figure 4.4 (a) shows the distribution of Newcomer OSS-Candidates onboarding OSS projects by means of qualitative analysis.

Barriers faced by Newcomer OSS-Candidates. Figure 4.4 (b) shows the results of the Likert-scale question related to barriers. The figure shows that *finding a way to start* is the most crucial barrier, with 22 responses being positive (i.e., 12 agree and 10 strongly agree responses). The second most crucial barrier is *technical hurdles*, receiving 18 positive responses (i.e., 15 agree and 3 strongly agree responses) which is related to "Technical Barrier" explained by [4, 94], which usually involves issues related to the complexity of the systems, lack of domain-specific knowledge, or other technological challenges. This aligns with findings where Newcomer OSS-Candidates faced difficulties in making technical

contributions or understanding the technological aspects of OSS projects. *Newcomer previous knowledge* is considered the third most crucial barrier with 16 responses (i.e., 10 agree and 6 strongly agree responses). On the other hand, the respondents are more likely to disagree with the statement that *social interaction* and *documentation* can be barriers for them to onboard OSS projects (i.e., 7 negative responses for each barrier). Although *social interaction* challenges, aligns with "Interpersonal barrier" explained by Balali et al. [4] where communication challenges occurs when newcomers are integrated into a diverse team, where individuals with distinct objectives, varied cultural backgrounds, and differing interpersonal skills collaborate.

RQ4 Summary: The quantitative analysis shows that 30% of Newcomer OSS-Candidates eventually onboarded OSS projects. The follow-up user survey also shows that 19 out of the 27 participants (70%) claim that they have made contributions to OSS repositories. I find that *finding a way to start* is the most agreed barrier for Newcomer OSS-Candidates.

5 Discussions

In this section, I discuss lessons learned and then revisit the expected implications against the actual results.

5.1 Lessons learned

This paper discusses two lessons learned that would be useful for future replication or improvements of the study. In the first lesson, I acknowledge that extracting the first contribution is not as trivial as I first envisioned. This is because the actual first commit might be just an ad-hoc test for the user, and not an actual meaningful contribution to a repository. In this research, I manually filtered out such contributions, but future work should consider a more systematic approach.

The second lesson to acknowledge is the process of onboarding may take a long time as it may be tied with the process of making a contribution to GitHub. As shown in the results for RQ4, different Newcomer OSS-Candidates are at different

stages of the onboarding process and may take time before they decide to submit the PR. Thus, I need to take into consideration a long enough time-window to evaluate whether or not a Newcomer OSS-Candidate will end up onboarding an OSS project. This lesson aligns with the "Process barriers" explained by the works [94, 107] where newcomers encounter difficulties in gaining a comprehensive understanding of the software they are tasked with contributing to, as well as in determining where to initiate their work.

5.2 Implications (Expectations vs. Actual Results)

Based on the results, I revisit the expected implications against the actual results of the study.

Suggestions for Newcomers. I speculated that the research would help Newcomer OSS-Candidates understand the kinds of contributions they target before onboarding a real OSS project. Actually, I found in Table 4.4 that Newcomer OSS-Candidates are not only engaged in adding new content, but 60% of them are also interested in management activities related to formatting code, cleaning up, and updating documentation through the submission of PRs. One example of this can be seen in the AEOL's repository¹¹, where a PR is submitted to add a new function to the project. Furthermore, RQ2 also reveals that after joining GitHub, 43% of Newcomer OSS-Candidates prefer to add new content in order to initialize or start a repository in their first commit. I found a common pattern is an initial commit that is uploading a website to the GitHub repository.¹² Finally, based on the RQ3 quantitative analysis, the majority of Newcomer OSS-Candidates have non-social based contributions in their contributions. As shown in Table 4.5 from RQ3 that after joining GitHub, Newcomer OSS-Candidates contributes in terms of single authorship are 73% of their first commits and 59% of their PRs, respectively. On the basis of evidence, I conclude that it is unlikely that Newcomer OSS-Candidates will be onboard to OSS projects immediately after joining GitHub.

¹¹<https://github.com/AEOL/round-robin/pull/1>

¹²<https://github.com/maanizfar/vanilla-js-web-projects/commit/e208d861b80762be8aa545567a300a7fad6aacf7>

I also speculated that I would reveal barriers on why some Newcomer OSS-Candidates never end up contributing to an OSS projects. According to the survey responses in RQ4, finding a way to start is one of the most challenging barriers, with 22 responses being positive (i.e., 12 agree and 10 strongly agree responses). Hence, inspired by these examples and combining all results, I recommend that Newcomer OSS-Candidates should not be afraid to individually contribute to their own code, contribute to upstream software repositories, or fork OSS projects before attempting to onboard. Last, regarding the most challenging barrier (i.e., finding a way to start), to this end, Newcomer OSS-Candidates should leverage suggestions provided by Subramanian et al. [98], including minor feature additions (a change of around 36 lines of code), minor documentation changes, selecting bug fixes, and changing catering to revised dependencies as first-timer friendly, which may relieve this problem. In addition, there are online resources¹³ that help Newcomer OSS-Candidates choose easy issues or opportunities to find ways to start contributing.

Suggestions for OSS Projects. It was speculated that the findings would reveal insights into what contributions may attract a Newcomer OSS-Candidate. Through the qualitative analysis of RQ2, Table 4.4 shows that in the first commits, 43% of Newcomer OSS-Candidates are typically engaged in adding new content to initialize the repositories, and 60% are involved in management activities in their PRs. Hence, I suggest that Newcomer OSS-Candidates may not have required skills to make immediate contributions. Instead, they may start with software based upstream experimental repositories. Hence, for OSS project, it might start with tasks to update the documentation, formatting or cleaning up code. One example of this can be seen in Bviveksingh’s upstream repository¹⁴, where a PR is submitted to update a software version.

I also speculated that OSS projects may benefit from the study, by identifying and offering the right contributions for the right Newcomer OSS-Candidates. Based on the results, I could not be able to provide concrete examples of contributions that match a specific Newcomer OSS-Candidate as the majority is a mixture of management and development activities. A potential future venue

¹³<https://hacktoberfest.digitalocean.com/>

¹⁴<https://github.com/Bviveksingh/angular-starter/pull/1>

for research could be to explore the kinds of OSS projects that these Newcomer OSS-Candidates end up onboarding. This would provide insights into matching the contributions to the onboarded OSS projects.

Suggestions for Researchers. It was speculated that non-software repositories that are personal have always been regarded as a challenge and are often filtered out from the dataset. I find that the majority of targeted repositories are software based repositories. Results include experimental (24%), documentation (21%), and web-based-application-libraries-and-frameworks (17%).

For researchers, this insight helps to understand the role of software based experimental, documentation, and web-based-application-libraries-and-frameworks repositories in platforms like GitHub, that should cater for developers. A potential avenue for research is to perform a finer-grain of analysis to understand the nature of these repositories.

6 Threats to Validity

In this section, I now discuss threats to the validity of the study.

External Validity. Two external threats are identified. I performed an empirical study on Newcomer OSS-Candidates that use GitHub the platform, and my observations may not be generalized to other platforms. Hence, I use GitHub as a case study. Another external threat is whether or not the 171 participants are representative of all Newcomer OSS-Candidates of the GitHub platform. Hence, I rely on the first contribution community. To represents the global population, future work should be conducted with other communities.

Construct Validity. I summarize three threats regarding construct validity. First, the qualitative analysis of manually classifying repositories and contribution kinds (RQ1, RQ2) are prone to error. To mitigate this threat, I took a systematic approach to first test the comprehension with 30 samples using Kappa agreement scores with three separate individuals. The second threat is to identified first contributions in RQ2 may not be actual contributions. To mitigate this, I perform a manual inspection to ignore any test, not meaningful contributions (i.e.,

commits or PRs) from any experimental repositories. The third potential threat exists in the quantitative analysis of matching engineered software projects using the curated database provided by Munaiah et al. [67]. I did contact the authors for assistance to help run the latest scripts, but were unsuccessful. Although the curated database might be outdated, I am confident that with the dataset, I was able to match 936 repositories.

Internal Validity. I identify three internal threats. The first threat is the first contributions by Newcomer OSS-Candidates may not be meaningful; they just want to get into the GitHub way of doing things. To mitigate this, I applied the first filter. The second internal threat to validity is related to results obtained from the quantitative analysis of RQ3 adapted to data visualization. As per the result, 27% and 41% of social coding is done by Newcomer OSS-Candidates in their first commits and PRs. The final threat is regarding errors in the tracking of repositories, due to repositories being deleted or a user changing user ids, as studied by

7 Related Work

A steady influx of new developers to an OSS project is crucial for its sustainability. In this section, I compare and contrast the work to the prior studies in three parts: first, I introduce the studies that are related to motivation for newcomers and OSS projects; second, I consider the studies regarding onboarding OSS projects; third, I discuss the studies with respect to the barriers that newcomers face.

7.1 Studies on Onboarding Motivators:

The sustainability and ongoing success of many community-based Open Source Software (OSS) projects hinge critically on the steady arrival of newcomers [37]. To facilitate the attraction of these newcomers and to alleviate the challenges they face during the onboarding process, there has been significant research focused on understanding the motivations of contributors. Studies have delved into understanding the motivations of contributors in Open Source Software (OSS)

from the perspective of distinct communities and roles. For instance, initial research primarily focused on well-established OSS communities like Linux [43] and Apache [80]. More recently, attention has shifted to newer communities, such as those involved in blockchain technology [13]. In an extensive review by Von Krogh et al. [108], which examined literature up to 2009, the researchers identified ten categories of motivations for newcomers. These motivations were classified into three groups: intrinsic, internalized-extrinsic, and extrinsic. Intrinsic motivation is characterized by the impetus behind an action being derived from the inherent interest or personal enjoyment associated with its execution. This form of motivation encompasses a range of factors such as ideology, altruism, kinship, and the simple pleasure derived from the activity itself [81]. Conversely, extrinsic motivation pertains to actions driven by the pursuit of distinct, external outcomes. Such motivations are often centered around obtaining tangible rewards or benefits, such as career advancements and financial compensation [38]. Developers have the capacity to internalize certain extrinsically motivated behaviors, thus transforming them into internalized-extrinsic motivations. These internalized-extrinsic motivations encompass factors such as reputation, reciprocity, the pursuit of knowledge, and the intrinsic value derived from one’s own use of the acquired skills or products [26, 39]. Within the literature, there is also notable attention directed towards the motivating forces and elements of attraction that guide new participants toward engagement in projects. As an illustration, Lakhani and Wolf [51] conducted research indicating that newcomers are primarily driven by external incentives, such as improved employment prospects and career progression. Additionally, these novices are motivated by intrinsic factors rooted in enjoyment, code-related challenges, and the enhancement of their programming capabilities. Moreover, Silva et al. [88], through the utilization of a combined methodology involving surveys and interviews, have discerned that the primary impetus driving students to participate in the Google Summer of Code (GSoC) program is the pursuit of an enriching experience, rather than an immediate commitment to becoming regular contributors. Although the monetary stipends offered hold significance as a motivating factor, the primary focus of student participation in open-source software (OSS) projects lies in the acquisition of valuable work experience and the cultivation of their careers. Besides, there is a complete body

of work that explored OSS developer's motivation and project's attractiveness Meirelles et al. [62], Santos et al. [82], Shah [83], Ye and Kishida [109]. Studies have also investigated the progression from newcomer to a core project member Ducheneaut [28], Fang and Neufeld [34], Krogh et al. [48], Marlow et al. [58], Nakakoji et al. [69]. On the other hand, Choi et al. [19] identified the seven most frequently used socialization tactics which have impact on newcomers' commitment to online groups. Other parts of the literature focus on the forces of motivation and attractiveness that drive newcomers towards projects. For example, Lakhani and Wolf [51] have found that external benefits (e.g., better jobs, career advancement) motivate primarily new contributors, along enjoyment-based intrinsic, code-based challenges, and improving programming skills. Compared to these, *my study* investigates how Newcomer OSS-Candidates contribute to both software (e.g., experimental, documentation, and web-based-application-libraries-and-frameworks) and non-software (e.g., academic, Web, and storage) repositories. Different to prior work, the goal is to study potential Newcomer OSS-Candidates that have the intention to onboard an OSS project.

7.2 Studies on Onboarding to Organizations

Organizational socialization, often referred to as onboarding, denotes a systematic progression by which new employees transition from their initial status as external entities to attain the status of integral constituents within an organization. The term "onboarding" encapsulates the comprehensive process aimed at facilitating the acquisition of essential knowledge, competencies, and behaviors requisite for achieving success within the context of their newly affiliated organizations. [8, 105]. Similarly, within the context of onboarding organizations, a pivotal concept revolves around that of the "newcomer," signifying an individual who is freshly joining an organization's workforce. It is worth noting that newcomers can also encompass individuals who are transitioning within the organization, such as those moving from one department to another or from one team to another. In contrast, two contrasting terms come to the fore: "outsiders" and "insiders." An "outsider" denotes an individual who is unfamiliar with the organization or team, while an "insider" denotes a seasoned staff member. It is

important to underscore that the transformation from an outsider to an insider is a gradual process that unfolds over time, with onboarding constituting the initial phase of this progression. *“To cross inclusionary boundaries means that [an outsider] becomes an insider with all the rights and privileges that go with such a position.”* [105]. Besides, existing scholarly investigations on onboarding can be categorized into four discrete viewpoints, as outlined by references [47]. These include the stages of progression for newcomers as outlined by BUCHANAN II [16], Feldman [35], the roles of actors involved with the onboarding of newcomers by Ashforth [3], Morrison [66], the strategies and practices employed by organizations for the onboarding of newcomers as elucidated by Bauer [7], Van Maanen and Schein [105], and finally, the content that newcomers are required to learn during the onboarding process, a topic explored by Chao et al. [18], Feldman [35]. Moreover, Dagenais et al. [25] identified three key factors that facilitate the integration of newcomer developers into software projects. These factors encompass early experimentation, the internalization of various project structures and cultures, and the regular validation of progress. Their investigation, grounded in qualitative research methods, involved interviews with 18 developers who had recently become part of ongoing software projects. It is noteworthy that the majority of the interviewees possessed prior experience in software development, and all were operating within agile teams, although the primary focus of the study did not center on agility. The sole agile practice emphasized as particularly beneficial and effective for newcomer orientation was the daily Scrum meeting. Compared to the literature, *my study* of systematic mapping highlights trends, identifies gaps, and provides insights into the evolution of onboarding tools and methods. It aims to deepen the understanding of onboarding challenges, practices, and the impact on software development communities and professional environments.

7.3 Studies on the Onboarding Process:

Numerous studies within the field of software engineering have delved into the onboarding procedures associated with open-source software (OSS) projects. Fagerholm et al. [31] provided initial insights and ongoing research findings pertaining to the onboarding process within virtual OSS teams. Furthermore, the impact of newcomers’ onboarding into OSS projects extends beyond the realm of OSS itself,

as demonstrated by the work of Dagenais et al. [24] and Begel and Simon [9] in the context of commercial software development environments. Additionally, Ducheneaut [28] took a sociological perspective when investigating onboarding, focusing on the viewpoints and experiences of individual developers. In addition to the aforementioned online theories and strategies, it is noteworthy that mentoring has traditionally served as a widely embraced approach within open-source software (OSS) communities. Mentoring is acknowledged as a pivotal element in facilitating the successful onboarding of newcomers into OSS projects, a perspective supported by the research of Fagerholm et al. [32] and Musicant et al. [68]. In their study, [99] characterized mentoring as a fundamental mechanism for the transfer of essential knowledge within enterprise contexts. Formal mentoring programs for newcomers in open-source software (OSS) projects are infrequently implemented, primarily because mentoring students takes a lot of time and energy, and experienced individuals are often preoccupied with their own commitments [28]. Nonetheless, an increasing number of OSS communities have come to recognize the significance of both attracting and mentoring newcomers, primarily driven by the imperative of sustaining project longevity [32]. In order to facilitate effective mentorship, researchers have undertaken investigations within OSS communities, with a predominant focus on assessing the impact [86, 87] and devising strategies for the mentoring process [5, 85, 97]. In contrast, another study conducted by Krogh et al. [48] introduces a joining script designed to guide developers interested in participating in open-source software (OSS) projects. Nakakoji et al. [69] have also examined OSS projects and proposed a classification of eight potential joining roles organized into concentric layers, referred to as "the onion patch." Furthermore, Zhou and Mockus [110] have identified a connection between an individual's willingness and the overall climate of a project, influencing the likelihood of an individual transitioning into a long-term contributor. Besides, there have been several studies that investigated the onboarding process. Fagerholm et al. [31] presented preliminary observations and results of in-progress research that studied the process of onboarding into virtual OSS teams. Commercial software development settings are also affected by newcomers onboarding towards OSS projects, as described by Begel and Simon [9], Dagenais et al. [24]. Ducheneaut [28] approached onboarding from a sociological point of view by considering

the perspective of individual developers. Previously, mentorship activity is recognized as an important factor for effective onboarding of newcomers towards OSS projects Fagerholm et al. [31, 32], Musicant et al. [68]. Swap et al. [99] described mentoring in their study as a basic knowledge transfer mechanism in the enterprise. A joining script is proposed in another study by Krogh et al. [48] for developers who want to take participate in OSS project. Nakakoji et al. [69] also studied the OSS project and proposed eight possible joining roles comprise of concentric layers called “the onion patch”. Zhou and Mockus [110] found that the willingness of individual and project’s climate were associated with odds that an individual would become a long-term contributor. Different from previous research, *my study* looks at the activities of potential newcomers before they onboard to Open Source Software on GitHub projects.

7.4 Studies on Social Coding on GitHub

GitHub, as a prominent social coding platform, stands out as the most widely adopted open-source version control platform. It functions as a repository for hosting various software and non-software projects, providing essential features such as pull request management, issue tracking, and workflow management [57, 77]. As of the time of this study, GitHub hosted an impressive count of over three hundred million repositories and served as a platform for more than a hundred million developers¹⁵. The inclination of individuals to establish new teams or transition to existing ones is significantly enhanced by the presence of technical standards and platforms. GitHub, in particular, serves as a prevalent and widely embraced distributed platform for code sharing and version control. Developers commonly possess a high level of familiarity with systems like git, which is supported by GitHub, and the collaborative processes it enables, thereby streamlining the process of contributing to, initiating, or transitioning to team projects on this platform. Nevertheless, GitHub extends its facilitative role in team formation and migration beyond technical aspects, thanks to its incorporation of *social coding* features. These features enable developers to monitor and assess each other’s activities, thereby forming comprehensive impressions of their social and technical competencies and conduct. The actions undertaken by developers

¹⁵<https://octoverse.github.com/>

on GitHub, such as bug reporting, pull request submissions, and commenting, are not solely archived but can also be tracked by fellow users and conveniently accessed through user-designated "home" pages. These records furnish a valuable vantage point not only for assessing an individual's technical competencies but also for gauging their social aptitudes and inclinations¹⁶. Lima et al. [56] constructed networks delineating the relationships among individuals who follow and collaborate with each other on GitHub, using push events related to code commits as a basis. Their findings reveal that following relationships on the platform tend to be unidirectional, and small teams frequently consist of members who are geographically proximate to one another. In a related study, Marlow et al. [59] emphasizes the pivotal role played by GitHub records in shaping developers' perceptions of one another, primarily through assessments of the work contributions made by their peers. In an interview conducted by Begel et al. [10], Brain Doll underscores that, in the current job landscape, "the number one way of getting a job...right now" is to showcase one's work on GitHub. His endorsement of GitHub is substantiated by an article featured in D.Terdiman [27]. Dabbish et al. [23] corroborate that developers indeed utilize these records, not only to form judgments about their peers but also to manage and enhance their own online reputations. Other notable advantage offered by a social coding platform is its capacity to solicit contributions to software projects from members of the community [21]. The continuous influx of newcomers into these projects, actively engaging in the development process, stands as a critical determinant of project success [71]. To enhance the prospects of project success, it is imperative to devise strategies that facilitate the onboarding of new developers and encourage their active participation in the project's development activities. Different from these studies, *my study* is conducted to specifically characterize the contributions patterns, social coding activities, and onboarding proportion of novice developers to GitHub, a key OSS platform.

7.5 Studies on the barriers to Onboarding:

Newcomers play a vital role in ensuring the viability, sustained success, and perpetuity of open-source software (OSS) projects, as highlighted by Kula and Robles

¹⁶<https://github.com/about>

[49] in their 2019 study. Despite their eagerness to engage in OSS endeavors, the onboarding of newcomers is often impeded by a range of social and technical obstacles, as documented in previous research [41, 63, 93, 95]. These barriers become apparent when newcomers endeavor to make contributions to OSS projects, encompassing a multifaceted array of challenges. In a comprehensive systematic literature review, Steinmacher et al. [93] have identified six primary categories of barriers encountered during the onboarding of newcomers, which encompass aspects related to newcomers' attributes, orientation, reception, documentation, technical complexities, and cultural disparities. Contrarily, research indicates that the impediments posed by social factors during the onboarding process are not only more prevalent but also more profound than technical obstacles [95]. In an additional study conducted by Mendez et al. [63], it was observed that tool and infrastructure-related challenges are prevalent within the spectrum of newcomer onboarding barriers, and these issues often encompass embedded social and cultural elements, such as gender biases. In contrast to conventional software development practices, communication emerges as a particularly pivotal social hurdle for newcomers in the realm of software development. This heightened significance is attributed to factors such as limited opportunities for team members to convene in person, disparities in time zones, and variations in cultural backgrounds [70, 100]. Facilitating the integration of newcomers into open-source software (OSS) projects necessitates measures aimed at mitigating the various barriers they encounter. To this end, a range of methodologies, guidelines, and tools have been put forth to assist newcomers in surmounting these obstacles. A common approach to address technical impediments involves the provision of technical training, as indicated by several studies [4, 61]. This training serves the purpose of equipping newcomers with the essential skills and knowledge, thereby expediting their onboarding process [14, 22]. Additionally, in the context of supporting newcomers' onboarding, numerous research initiatives have recommended the implementation of mentoring systems [5, 102], the assignment of easily manageable bug resolution tasks [50, 101], and the establishment of a dedicated portal site designed specifically for newcomers. Furthermore, *my work* takes a first look at potential Newcomer OSS-Candidates before they onboard. Hence, insights show that learning the social platform contribution process (i.e.,

PR process) may co-inside with onboarding.

8 Conclusion

In this work, I studied the activities of a particular category of potential contributors (i.e., Newcomer OSS-Candidates) towards OSS projects on GitHub. To do that, I (i) analyze what kinds of repositories they target, (ii) investigate what kinds of contributions come from them, (iii) analyze to what extent they practice social coding with their contributions, and (iv) explore what proportion of them eventually onboard an OSS project.

I observe that (i) 66% of Newcomer OSS-Candidates target software based repositories; (ii) the majority of their contributions are related to development activities and maintenance activities, respectively, for commits and PRs; (iii) Newcomer OSS-Candidates are less likely to practice social coding in their contributions in terms of multiple authorship; and (iv) 70% of them eventually onboarded OSS projects in a follow-up survey and cited that *finding a way to start* is the most crucial barrier. As GitHub continues to grow, so does the possibility to attract potential contributors to OSS projects. My work presents the first step towards understanding these potential contributors and reveals insights to provide a guidance for them to sustain themselves and join an OSS project.

5 | Conclusion

Onboarding within the realm of Software Engineering (SE) is recognized as a critical practice, significantly influencing both open source software (OSS) and proprietary software projects. Over the past decade, advancements in onboarding tools and methodologies have facilitated more efficient integration processes, gaining widespread acceptance in both OSS and organizational sectors. GitHub, renowned as the premier Open Source version control platform, hosts a vast array of over 3 hundred million repositories. It functions as a social coding platform for both software and non-software projects. The literature underscores the criticality of continually incorporating newcomers into GitHub OSS projects and onboard them to the development process for the success of these ventures.

The dissertation titled "Understanding Newcomer Activities Prior to Onboarding Open Source Software (OSS) Projects on GitHub" comprises the studies. The initial segment, titled "Systematic Map related Work" about the key topics addressed of Developers' Onboarding in Software Engineering", conducts a systematic review of related work pertaining to developers' onboarding in the field of software engineering. The objective is to delineate the diversity and progression of covering key topics addressed in both open-source software (OSS) and non-OSS contexts. The subsequent segment, "Newcomer OSS-Candidates, characterizing contributions, and analyzing onboarding proportion to GitHub OSS projects," delves into empirical investigations broken into three parts: (i) identifying Newcomers OSS-candidates (ii) validation and characterizing their pre-onboarding contributions, and finally (iv) investigate their onboarding proportion to GitHub OSS projects, a renowned OSS platform. The overarching goal of the dissertation is to amalgamate insights garnered from a comprehensive and wide-ranging

systematic mapping study with focused empirical observations. This holistic approach aims to provide a thorough comprehension of developer onboarding within the OSS landscape, with a particular emphasis on pre-onboarding strategies for integration and the potential contributions of novice developers.

1 Contributions

The findings derived from this thesis have the potential to offer advantages to a broad spectrum of stakeholders, including developers, researchers, and practitioners. In the following sections, I summarize the contributions and provide recommendations for each part as follows:

Part I Map for future Onboarding Research

- Concerning topics addressed in onboarding studies, I identify that the majority of studies have leveraged data on contributions to gain a deeper understanding of people (38 studies) and systems (21 studies) within the comprehension of the software engineering process. Conversely, the characterization of contribution assessments has been less frequently addressed, with an emphasis on the evaluation of team or developer performance when such characterization occurs.

Suggestion. The study uncovers a wide range of topics, with a notable concentration on understanding software engineering phenomena and constructing models and tools. The implication here is that there is an opportunity for future research to explore less addressed areas, such as the characterization of contribution assessments and the training aspect of onboarding.

These overall findings of the systematic literature review collectively highlight the dynamic and evolving nature of SE onboarding research, pointing toward potential areas for future investigations to enrich the field.

Part II Newcomer OSS-Candidates: Characterizing Contributions of Novice Developers

Table 5.1. Guidelines for (N)Newcomers and (PM)Project Maintainers

Guidelines
N) Encouraged to start with manageable tasks to overcome initial barriers like finding a way to start.
N) Take a start with forking experimental software-based repositories.
N) Create upstream software-based documentation repositories.
N) Initially focusing on adding new content by making the first commit.
N) For PR, focus on formatting code, cleaning up, and updating documentation.
N) Engage in individual contributions in the beginning at social coding platform.
PM) Start with tasks to update documentation, formatting or cleaning up code.
PM) finding a way to start is notable barrier, suggesting need for clearer pathways for engagement in OSS projects.
PM) OSS projects will benefit by offering right tasks for a suitable newcomer.

- For target repositories, I observe that 66% of Newcomer OSS-Candidates target software based repositories. Furthermore, Experimental and Documentation are the most frequently targeted software repository kinds for fork and upstream repositories, i.e., 24% and 21%, and web-based-application-libraries-and-frameworks (17%). respectively.

Suggestion. The study finds that Newcomer OSS-Candidates primarily target software-based repositories. This indicates a potential interest in software development activities among newcomers to GitHub. For researchers, this insight helps to understand the role of software based experimental, documentation, and web-based-application-libraries-and-frameworks repositories in platforms like GitHub, that should cater for developers. A potential avenue for research is to perform a finer-grain of analysis to understand the nature of these repositories.

- For the first commit contributions, I find that 74% of contributions from Newcomer OSS-Candidates are related to development activities. For the first PR contributions, the results show that 60% of contributions are associated with management activities.

Suggestion. Newcomers tend to focus more on development activities in their initial contributions, suggesting that initial engagements are often about adding new content or doing management changes. Hence, I suggest that Newcomer OSS-Candidates may not have the required skills to make immediate contributions. Instead, they may start with software-based upstream experimental repositories. Hence, for OSS project, it might start with tasks to update the documentation, formatting or cleaning up code.

- Third, concerning social coding, the results show that after joining GitHub, a majority of Newcomer OSS-Candidates (i.e., 73% of first commits and 59% of PRs) do not share code with other authors.

Suggestion. The majority of Newcomer OSS-Candidates do not practice social coding with their first contributions, indicating a tendency towards individual work in the initial stages. I also conclude that it is unlikely that Newcomer OSS-Candidates will be onboard to OSS projects immediately after joining GitHub.

- Fourth, concerning onboarding proportion, results shows that 30% of Newcomer OSS-candidates eventually onboarded engineered OSS repositories. Complementary, a follow-up user survey shows that 70% of the studied participants ended up making contributions to an OSS repository. Newcomer OSSCandidates strongly agreed that they face the barrier of finding a way to start, while social interaction received the most mixed responses as a barrier.

Suggestion. A significant proportion of Newcomer OSS-Candidates eventually contribute to OSS projects. However, finding a way to start is a notable barrier, suggesting the need for clearer pathways for engagement in OSS projects.

Combining all results, I recommend that Newcomer OSS-Candidates should not be afraid to individually contribute to their own code, contribute to upstream software repositories, or fork OSS projects before attempting to onboard. Table 5.1 presents more clearer and concrete guidelines for Newcomer OSS-Candidates and project maintainers to follow.

Appendix A. Selected studies

- **C001.** Sharma, G.G. and Stol, K.J., 2020. Exploring onboarding success, organizational fit, and turnover intention of software professionals. *Journal of Systems and Software*, 159, p.110442.
- **C002.** Gregory, P., Strode, D.E., Sharp, H. and Barroca, L., 2022. An onboarding model for integrating newcomers into agile project teams. *Information and Software Technology*, 143, p.106792.
- **C003.** Yates, R., Power, N. and Buckley, J., 2020. Characterizing the transfer of program comprehension in onboarding: an information-push perspective. *Empirical Software Engineering*, 25, pp.940-995.
- **C004.** Britto, R., Smite, D., Damm, L.O. and Börstler, J., 2020. Evaluating and strategizing the onboarding of software developers in large-scale globally distributed projects. *Journal of Systems and Software*, 169, p.110699.
- **C005.** Tan, X., Zhou, M. and Zhang, L., 2023. Understanding Mentors' Engagement in OSS Communities via Google Summer of Code. *IEEE Transactions on Software Engineering*.
- **C006.** Steinmacher, I., Silva, M.A.G., Gerosa, M.A. and Redmiles, D.F., 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, 59, pp.67-85.
- **C007.** Padala, H.S., Mendez, C., Fronchetti, F., Steinmacher, I., Steine-Hanson, Z., Hilderbrand, C., Horvath, A., Hill, C., Simpson, L., Burnett, M. and Gerosa, M., 2020. How gender-biased tools shape newcomer experiences in oss projects. *IEEE Transactions on Software Engineering*, 48(1), pp.241-259.
- **C008.** Steinmacher, I., Treude, C. and Gerosa, M.A., 2018. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, 36(4), pp.41-49.

- **C009.** Yue, Y., Wang, Y. and Redmiles, D., 2022. Off to a Good Start: Dynamic Contribution Patterns and Technical Success in an OSS Newcomer's Early Career. *IEEE Transactions on Software Engineering*, 49(2), pp.529-548.
- **C010.** Fagerholm, F., Guinea, A.S., Borenstein, J. and Münch, J., 2014. Onboarding in open source projects. *IEEE Software*, 31(6), pp.54-61.
- **C011.** Liu, C., Yang, D., Zhang, X., Ray, B. and Rahman, M.M., 2018. Recommending github projects for developer onboarding. *IEEE Access*, 6, pp.52082-52094.
- **C012.** Bao, L., Xia, X., Lo, D. and Murphy, G.C., 2019. A large scale study of long-time contributor prediction for GitHub projects. *IEEE Transactions on Software Engineering*, 47(6), pp.1277-1298.
- **C013.** Assavakamhaenghan, N., Wattanakriengkrai, S., Shimada, N., Kula, R.G., Ishio, T. and Matsumoto, K., 2023. Does the first response matter for future contributions? A study of first contributions. *Empirical Software Engineering*, 28(3), p.75.
- **C014.** Gharehyazie, M., Posnett, D., Vasilescu, B. and Filkov, V., 2015. Developer initiation and social interactions in OSS: A case study of the Apache Software Foundation. *Empirical Software Engineering*, 20, pp.1318-1353.
- **C015.** Foundjem, A., Constantinou, E., Mens, T. and Adams, B., 2022. A mixed-methods analysis of micro-collaborative coding practices in OpenStack. *Empirical Software Engineering*, 27(5), p.120.
- **C016.** Wang, D., Kondo, M., Kamei, Y., Kula, R.G. and Ubayashi, N., 2023. When conversations turn into work: a taxonomy of converted discussions and issues in GitHub. *Empirical Software Engineering*, 28(6), p.138.
- **C017.** Pinto, G., Steinmacher, I., Dias, L.F. and Gerosa, M., 2018. On the challenges of open-sourcing proprietary software projects. *Empirical Software Engineering*, 23, pp.3221-3247.

- **C018.** Calefato, F., Gerosa, M.A., Iaffaldano, G., Lanubile, F. and Steinmacher, I., 2022. Will you come back to contribute? Investigating the inactivity of OSS core developers in GitHub. *Empirical Software Engineering*, 27(3), p.76.
- **C019.** Wang, Y. and Redmiles, D., 2021. IIAG: a data-driven and theory-inspired approach for advising how to interact with new remote collaborators in OSS teams. *Automated Software Engineering*, 28(2), p.5.
- **C020.** Li, Z., Yu, Y., Wang, T., Yin, G., Li, S. and Wang, H., 2021. Are you still working on this? An empirical study on pull request abandonment. *IEEE Transactions on Software Engineering*, 48(6), pp.2173-2188.
- **C021.** Trinkenreich, B., Guizani, M., Wiese, I., Conte, T., Gerosa, M., Sarma, A. and Steinmacher, I., 2021. Pots of Gold at the End of the Rainbow: What is Success for Open Source Contributors?. *IEEE Transactions on Software Engineering*, 48(10), pp.3940-3953.
- **C022.** Barcomb, A., Stol, K.J., Fitzgerald, B. and Riehle, D., 2020. Managing episodic volunteers in free/libre/open source software communities. *IEEE Transactions on Software Engineering*, 48(1), pp.260-277.
- **C023.** Sarker, J., Turzo, A.K., Dong, M. and Bosu, A., 2023. Automated Identification of Toxic Code Reviews Using ToxiCR. *ACM Transactions on Software Engineering and Methodology*.
- **C024.** Calefato, F., Lanubile, F. and Vasilescu, B., 2019. A large-scale, in-depth analysis of developers' personalities in the apache ecosystem. *Information and Software Technology*, 114, pp.1-20.
- **C025.** Wessel, M., De Souza, B.M., Steinmacher, I., Wiese, I.S., Polato, I., Chaves, A.P. and Gerosa, M.A., 2018. The power of bots: Characterizing and understanding bots in oss projects. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW), pp.1-19.
- **C026.** Wang, Y., Kadiyala, H. and Rubin, J., 2021. Promises and challenges of microservices: an exploratory study. *Empirical Software Engineering*, 26(4), p.63.

- **C027.** Wessel, M., Serebrenik, A., Wiese, I., Steinmacher, I. and Gerosa, M.A., 2022. Quality gatekeepers: investigating the effects of code review bots on pull request activities. *Empirical Software Engineering*, 27(5), p.108.
- **C028.** Kavaler, D., Devanbu, P. and Filkov, V., 2019. Whom are you going to call? determinants of @-mentions in github discussions. *Empirical Software Engineering*, 24, pp.3904-3932.
- **C029.** Maeprasart, V., Wattanakriengkrai, S., Kula, R.G., Treude, C. and Matsumoto, K., 2023. Understanding the role of external pull requests in the NPM ecosystem. *Empirical Software Engineering*, 28(4), pp.1-23.
- **C030.** Warncke-Wang, M., Ho, R., Miller, M. and Johnson, I., 2023. Increasing Participation in Peer Production Communities with the Newcomer Homepage. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW2), pp.1-26.
- **C031.** Fritz, T., Murphy, G.C., Murphy-Hill, E., Ou, J. and Hill, E., 2014. Degree-of-knowledge: Modeling a developer's knowledge of code. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 23(2), pp.1-42.
- **C032.** Wang, Z., Feng, Y., Wang, Y., Jones, J.A. and Redmiles, D., 2020. Unveiling elite developers' activities in open source projects. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(3), pp.1-35.
- **C033.** Lee, S., Wu, R., Cheung, S.C. and Kang, S., 2019. Automatic detection and update suggestion for outdated api names in documentation. *IEEE Transactions on Software Engineering*, 47(4), pp.653-675.
- **C034.** Barcomb, A., Kaufmann, A., Riehle, D., Stol, K.J. and Fitzgerald, B., 2018. Uncovering the periphery: A qualitative survey of episodic volunteering in free/libre and open source software communities. *IEEE Transactions on Software Engineering*, 46(9), pp.962-980.

- **C035.** Zhang, Y., Liu, H., Tan, X., Zhou, M., Jin, Z. and Zhu, J., 2022. Turnover of companies in OpenStack: Prevalence and rationale. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(4), pp.1-24.
- **C036.** Chinthanet, B., Reid, B., Treude, C., Wagner, M., Kula, R.G., Ishio, T. and Matsumoto, K., 2021. What makes a good Node. js package? Investigating Users, Contributors, and Runnability. *arXiv preprint arXiv:2106.12239*.
- **C037.** Trinkenreich, B., Guizani, M., Wiese, I., Sarma, A. and Steinmacher, I., 2020. Hidden figures: Roles and pathways of successful oss contributors. *Proceedings of the ACM on human-computer interaction*, 4(CSCW2), pp.1-22.
- **C038.** Guizani, M., Chatterjee, A., Trinkenreich, B., May, M.E., Noa-Guevara, G.J., Russell, L.J., Cuevas Zambrano, G.G., Izquierdo-Cortazar, D., Steinmacher, I., Gerosa, M.A. and Sarma, A., 2021. The long road ahead: Ongoing challenges in contributing to large oss organizations and what to do. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2), pp.1-30.
- **C039.** Liu, Y., Noei, E. and Lyons, K., 2022. How README files are structured in open source Java projects. *Information and Software Technology*, 148, p.106924.
- **C040.** Wessel, M., Wiese, I., Steinmacher, I. and Gerosa, M.A., 2021. Don't disturb me: Challenges of interacting with software bots on open source software projects. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2), pp.1-21.
- **C041.** Kapitsaki, G.M., Tselikas, N.D., Kyriakou, K.I.D. and Papoutsoglou, M., 2022. Help me with this: A categorization of open source software problems. *Information and Software Technology*, 152, p.107034.
- **C042.** Wang, T., Wang, S. and Chen, T.H.P., 2023. Study the correlation between the readme file of GitHub projects and their popularity. *Journal*

of Systems and Software, 205, p.111806.

- **C043.** Li, R., Pandurangan, P., Frluckaj, H. and Dabbish, L., 2021. Code of conduct conversations in open source software projects on github. Proceedings of the ACM on Human-computer Interaction, 5(CSCW1), pp.1-31.
- **C044.** Santos, F., Vargovich, J., Trinkenreich, B., Santos, I., Penney, J., Britto, R., Pimentel, J.F., Wiese, I., Steinmacher, I., Sarma, A. and Gerosa, M.A., 2023. Tag that issue: Applying API-domain labels in issue tracking systems. arXiv preprint arXiv:2304.02877.
- **C045.** Frluckaj, H., Dabbish, L., Widder, D.G., Qiu, H.S. and Herbsleb, J.D., 2022. Gender and participation in open source software development. Proceedings of the ACM on Human-Computer Interaction, 6(CSCW2), pp.1-31.
- **C046.** Mueller, D. and Izquierdo-Cortazar, D., 2019. From art to science: The evolution of community development. IEEE Software, 36(6), pp.23-28.
- **C047.** Steinmacher, I., Gerosa, M.A. and Redmiles, D., 2014, February. Attracting, onboarding, and retaining newcomer developers in open source software projects. In Workshop on Global Software Development in a CSCW Perspective (Vol. 16, p. 20).
- **C048.** Azanza, M., Irastorza, A., Medeiros, R. and Díaz, O., 2021, May. Onboarding in software product lines: Concept maps as welcome guides. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 122-133). IEEE.
- **C049.** Ju, A., Sajnani, H., Kelly, S. and Herzig, K., 2021, May. A case study of onboarding in software teams: Tasks and strategies. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 613-623). IEEE.
- **C050.** Medeiros, R., 2021, May. Unburdening onboarding in software product lines. In 2021 IEEE/ACM 43rd International Conference on Software

Engineering: Companion Proceedings (ICSE-Companion) (pp. 260-262). IEEE.

- **C051.** Horiguchi, H., Omori, I. and Ohira, M., 2021, March. Onboarding to open source projects with good first issues: A preliminary analysis. In 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 501-505). IEEE.
- **C052.** Labuschagne, A. and Holmes, R., 2015, May. Do onboarding programs work?. In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories (pp. 381-385). IEEE.
- **C053.** Pham, R., 2014, November. Improving the software testing skills of novices during onboarding through social transparency. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 803-806).
- **C054.** Hilton, M. and Begel, A., 2018, May. A study of the organizational dynamics of software teams. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (pp. 191-200).
- **C055.** Casalnuovo, C., Vasilescu, B., Devanbu, P. and Filkov, V., 2015, August. Developer onboarding in GitHub: the role of prior social links and language experience. In Proceedings of the 2015 10th joint meeting on foundations of software engineering (pp. 817-828).
- **C056.** Tan, X., Zhou, M. and Sun, Z., 2020, November. A first look at good first issues on GitHub. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 398-409).
- **C057.** Fronchetti, F., Shepherd, D.C., Wiese, I., Treude, C., Gerosa, M.A. and Steinmacher, I., 2023. Do CONTRIBUTING Files Provide Information about OSS Newcomers' Onboarding Barriers?.

- **C058.** Santos, I., Wiese, I., Steinmacher, I., Sarma, A. and Gerosa, M.A., 2022, March. Hits and Misses: Newcomers' ability to identify Skills needed for OSS tasks. In 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 174-183). IEEE.
- **C059.** Rodeghero, P., Zimmermann, T., Houck, B. and Ford, D., 2021, May. Please turn your cameras on: Remote onboarding of software developers during a pandemic. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 41-50). IEEE.
- **C060.** Rehman, I., Wang, D., Kula, R.G., Ishio, T. and Matsumoto, K., 2020, September. Newcomer candidate: Characterizing contributions of a novice developer to github. In 2020 IEEE international conference on software maintenance and evolution (ICSME) (pp. 855-855). IEEE.
- **C061.** Stanik, C., Montgomery, L., Martens, D., Fucci, D. and Maalej, W., 2018, September. A simple nlp-based approach to support onboarding and retention in open source communities. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 172-182). IEEE.
- **C062.** He, H., Zhou, M., Wang, Q. and Li, J., 2023, May. Open Source Software Onboarding as a University Course: An Experience Report. In 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 324-336). IEEE.
- **C063.** Steinmacher, I., Conte, T.U., Treude, C. and Gerosa, M.A., 2016, May. Overcoming open source project entry barriers with a portal for newcomers. In Proceedings of the 38th International Conference on Software Engineering (pp. 273-284).
- **C064.** Kumar, S., Wallace, C. and Young, M., 2016, May. Mentoring trajectories in an evolving agile workplace. In Proceedings of the 38th International Conference on Software Engineering Companion (pp. 142-151).

- **C065.** Mendez, C., Padala, H.S., Steine-Hanson, Z., Hilderbrand, C., Horvath, A., Hill, C., Simpson, L., Patil, N., Sarma, A. and Burnett, M., 2018, May. Open source barriers to entry, revisited: A sociotechnical perspective. In Proceedings of the 40th International conference on software engineering (pp. 1004-1015).
- **C066.** He, H., Su, H., Xiao, W., He, R. and Zhou, M., 2022, November. GFI-bot: automated good first issue recommendation on GitHub. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1751-1755).
- **C067.** Xiao, W., He, H., Xu, W., Tan, X., Dong, J. and Zhou, M., 2022, May. Recommending good first issues in GitHub OSS projects. In Proceedings of the 44th International Conference on Software Engineering (pp. 1830-1842).
- **C068.** Silva, J., Wiese, I., German, D.M., Treude, C., Gerosa, M.A. and Steinmacher, I., 2020, November. A theory of the engagement in open source projects via summer of code programs. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 421-431).
- **C069.** Guizani, M., Zimmermann, T., Sarma, A. and Ford, D., 2022, May. Attracting and retaining oss contributors with a maintainer dashboard. In Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society (pp. 36-40).
- **C070.** Lee, A., 2018. One-time contributors to FLOSS: surveys and data analysis. ACM SIGSOFT Software Engineering Notes, 43(1), pp.1-6.
- **C071.** Sarma, A., Gerosa, M.A., Steinmacher, I. and Leano, R., 2016, November. Training the future workforce through task curation in an OSS ecosystem. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 932-935).

- **C072.** Bayati, S., 2018, May. Understanding newcomers success in open source community. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings (pp. 224-225).
- **C073.** Pham, R., Stoliar, Y. and Schneider, K., 2015, August. Automatically recommending test code examples to inexperienced developers. In Proceedings of the 2015 10th joint meeting on foundations of software engineering (pp. 890-893).
- **C074.** Feng, Z., Chatterjee, A., Sarma, A. and Ahmed, I., 2022, November. A case study of implicit mentoring, its prevalence, and impact in apache. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 797-809).
- **C075.** Santos, F., 2023. Supporting the Task-driven Skill Identification in Open Source Project Issue Tracking Systems. ACM SIGSOFT Software Engineering Notes, 48(1), pp.54-58.
- **C076.** Panichella, S., 2015, September. Supporting newcomers in software development projects. In 2015 IEEE international conference on software maintenance and evolution (ICSME) (pp. 586-589). IEEE.
- **C077.** Santos, I., Pimentel, J.F., Wiese, I., Steinmacher, I., Sarma, A. and Gerosa, M.A., 2023. Designing for Cognitive Diversity: Improving the GitHub Experience for Newcomers. arXiv preprint arXiv:2301.10912.
- **C078.** Xiao, W., Li, J., He, H., Qiu, R. and Zhou, M., 2023. Personalized First Issue Recommender for Newcomers in Open Source Projects. arXiv preprint arXiv:2308.09038.
- **C079.** Santos, F., 2023, May. Skill Recommendation for New Contributors in Open-Source Software. In 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 311-313). IEEE.

- **C080.** Wessel, M., Gerosa, M.A. and Shihab, E., 2022, May. Software bots in software engineering: benefits and challenges. In Proceedings of the 19th International Conference on Mining Software Repositories (pp. 724-725).
- **C081.** Trinkenreich, B., 2021, May. 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) (pp. 293-298). IEEE.
- **C082.** Steinmacher, I., Pinto, G., Wiese, I.S. and Gerosa, M.A., 2018, May. Almost there: A study on quasi-contributors in open source software projects. In Proceedings of the 40th International Conference on Software Engineering (pp. 256-266).
- **C083.** Tan, X., Chen, Y., Wu, H., Zhou, M. and Zhang, L., 2023. Is It Enough to Recommend Tasks to Newcomers? Understanding Mentoring on Good First Issues. arXiv preprint arXiv:2302.05058.
- **C084.** Gerosa, M., Wiese, I., Trinkenreich, B., Link, G., Robles, G., Treude, C., Steinmacher, I. and Sarma, A., 2021, May. The shifting sands of motivation: Revisiting what drives contributors in open source. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1046-1058). IEEE.
- **C085.** Pinto, G., Ferreira, C., Souza, C., Steinmacher, I. and Meirelles, P., 2019, May. Training software engineers using open-source software: the students' perspective. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 147-157). IEEE.
- **C086.** Anderson, J., Steinmacher, I. and Rodeghero, P., 2020, September. Assessing the Characteristics of FOSS Contributions in Network Automation Projects. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 324-335). IEEE.
- **C087.** Barcomb, A., Stol, K.J., Riehle, D. and Fitzgerald, B., 2019, May. Why do episodic volunteers stay in FLOSS communities?. In 2019

IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 948-959). IEEE.

- **C088.** Huang, Y., Ford, D. and Zimmermann, T., 2021, May. Leaving my fingerprints: Motivations and challenges of contributing to OSS for social good. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1020-1032). IEEE.
- **C089.** Dutta, R., Costa, D.E., Shihab, E. and Tajmel, T., 2023. Diversity Awareness in Software Engineering Participant Research. arXiv preprint arXiv:2302.00042.
- **C090.** Tómasdóttir, K.F., Aniche, M. and Van Deursen, A., 2017, October. Why and how JavaScript developers use linters. In 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 578-589). IEEE.
- **C091.** Xiao, W., He, H., Xu, W., Zhang, Y. and Zhou, M., 2023. How Early Participation Determines Long-Term Sustained Activity in GitHub Projects?. arXiv preprint arXiv:2308.06005.
- **C092.** Pinto, G., Steinmacher, I. and Gerosa, M.A., 2016, March. More common than you think: An in-depth study of casual contributors. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER) (Vol. 1, pp. 112-123). IEEE.
- **C093.** Dias, L.F., Steinmacher, I., Pinto, G., Da Costa, D.A. and Gerosa, M., 2016, October. How does the shift to GitHub impact project collaboration?. In 2016 IEEE international conference on software maintenance and evolution (ICSME) (pp. 473-477). IEEE.
- **C094.** Rahman, A., Bhuiyan, F.A., Hassan, M.M., Shahriar, H. and Wu, F., 2022, June. Towards Automation for MLOps: An Exploratory Study of Bot Usage in Deep Learning Libraries. In 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1093-1097). IEEE.

- **C095.** Fang, H., Lamba, H., Herbsleb, J. and Vasilescu, B., 2022, May. " This is damn slick!" estimating the impact of tweets on open source project popularity and new contributors. In Proceedings of the 44th International Conference on Software Engineering (pp. 2116-2129).
- **C096.** Qiu, H.S., Li, Y.L., Padala, S., Sarma, A. and 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), pp.1-29.
- **C097.** Santos, F., Penney, J., Pimentel, J.F., Wiese, I., Steinmacher, I. and Gerosa, M.A., 2023. Tell Me Who Are You Talking to and I Will Tell You What Issues Need Your Skills.
- **C098.** Santos, F., Wiese, I., Trinkenreich, B., Steinmacher, I., Sarma, A. and Gerosa, M.A., 2021, May. Can i solve it? identifying apis required to complete oss tasks. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) (pp. 346-257). IEEE.
- **C099.** Silva, J.D.O., Wiese, I.S., German, D.M., Steinmacher, I.F. and Gerosa, M.A., 2017, September. How long and how much: What to expect from summer of code participants?. In 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 69-79). IEEE.
- **C100.** Vargovich, J., Santos, F., Penney, J., Gerosa, M.A. and Steinmacher, I., 2023. Givemelabeledissues: An open source issue recommendation system. arXiv preprint arXiv:2303.13418.
- **C101.** Gautam, A., Vishwasrao, S. and Servant, F., 2017, May. An empirical study of activity, popularity, size, testing, and stability in continuous integration. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR) (pp. 495-498). IEEE.
- **C102.** Middleton, J., Murphy-Hill, E., Green, D., Meade, A., Mayer, R., White, D. and McDonald, S., 2018, May. Which contributions predict whether developers are accepted into github teams. In Proceedings of the

15th International Conference on Mining Software Repositories (pp. 403-413).

References

- [1] B A. Kitchenham. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.
- [2] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, and Alexander Chatzigeorgiou. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. Information and Software Technology, 106:201–230, 2019.
- [3] Blake Ashforth. Role transitions in organizational life: An identity-based perspective. Routledge, 2000.
- [4] 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. Comput. Supported Coop. Work, page 679–714, 2018.
- [5] 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 Proceedings of the 16th International Symposium on Open Collaboration, pages 1–14, 2020.
- [6] Leonor Barroca, Peggy Gregory, Kati Kuusinen, Helen Sharp, and Raid AlQaisi. Sustaining agile beyond adoption. In 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 22–25. IEEE, 2018.
- [7] Talya N Bauer. Maximizing success. SHRM Foundation’s Effective practice guidelines series, 2010.

- [8] Talya N Bauer and Berrin Erdogan. Organizational socialization: The effective onboarding of new employees. 2011.
- [9] Andrew Begel and Beth Simon. Novice software developers, all over again. ICER'08 - Proceedings of the ACM Workshop on International Computing Education Research, 09 2008.
- [10] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and top-coder. IEEE software, 30(1):52–66, 2013.
- [11] H. Bernard. Research Methods in Anthropology: Qualitative and Quantitative Approaches. Rowman & Littlefield, 2011.
- [12] H. Borges, A. Hora, and M. T. Valente. Understanding the factors that impact the popularity of GitHub repositories. In ICSME, 2016.
- [13] Amiangshu Bosu, Anindya Iqbal, Rifat Shahriyar, and Partha Chakraborty. Understanding the motivations, challenges and needs of blockchain software developers: A survey. Empirical Software Engineering, 24(4):2636–2673, 2019.
- [14] Ricardo Britto, Daniela S Cruzes, Darja Smite, and Aivars Sablis. Onboarding software developers and teams in three globally distributed legacy projects: A multi-case study. Journal of Software: Evolution and Process, 30(4):e1921, 2018.
- [15] Jim Buchan, Stephen G MacDonell, and Jennifer Yang. Effective team onboarding in agile software development: techniques and goals. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pages 1–11. IEEE, 2019.
- [16] BRUCE BUCHANAN II. Building organizational commitment: The socialization of managers in work organizations. Yale University, 1972.
- [17] Victor R Basili, Gianluigi Caldiera, and H Dieter Rombach. The goal question metric approach. Encyclopedia of software engineering, pages 528–532, 1994.
- [18] Georgia T Chao, Anne M O’Leary-Kelly, Samantha Wolf, Howard J Klein, and Philip D Gardner. Organizational socialization: Its content and consequences. Journal of Applied psychology, 79(5):730, 1994.

- [19] Boreum Choi, Kira Alexander, Robert E Kraut, and John M Levine. Socialization tactics in wikipedia and their effects. In Proceedings of the 2010 ACM conference on Computer supported cooperative work, pages 107–116, 2010.
- [20] Jailton Coelho and Marco Tulio Valente. Why modern open source projects fail. In FSE, 2017.
- [21] Valerio Cosentino, Javier L. Cánovas Izquierdo, and Jordi Cabot. A systematic mapping study of software development with github. IEEE Access, 5:7173–7192, 2017. doi: 10.1109/ACCESS.2017.2682323.
- [22] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. Educating to achieve healthy open source ecosystems. In Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, pages 1–7, 2018.
- [23] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In Proceedings of the ACM 2012 conference on computer supported cooperative work, pages 1277–1286, 2012.
- [24] Barthélémy Dagenais, Harold Ossher, Rachel K. E. Bellamy, Martin P. Robillard, and Jacqueline P. de Vries. Moving into a New Software Project Landscape, page 275–284. Association for Computing Machinery, 2010.
- [25] Barthélémy Dagenais, Harold Ossher, Rachel KE Bellamy, Martin P Robillard, and Jacqueline P De Vries. Moving into a new software project landscape. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, pages 275–284, 2010.
- [26] Edward L Deci and Richard M Ryan. The support of autonomy and the control of behavior. Journal of personality and social psychology, 53(6):1024, 1987.
- [27] D.Terdiman. Forget linkedin: Companies turn to github to find tech talent. <http://www.cnet.com/news/forget-linkedin-companiesturn-to-github-to-find-tech-talent/>, 2012.
- [28] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. Computer Supported Cooperative Work (CSCW), 14: 323–368, 2005.

- [29] Nadia Eghbal. Roads and bridges: The unseen labor behind our digital infrastructure. Ford Foundation, 2016.
- [30] Nadia Eghbal. Working in public: the making and maintenance of open source software. Stripe Press, 2020.
- [31] Fabian Fagerholm, Patrik Johnson, Alejandro Guinea, Jay Borenstein, and Jürgen Münch. Onboarding in open source software projects: A preliminary analysis. In 2013 IEEE 8th International Conference on Global Software Engineering Workshops, 08 2013.
- [32] 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.
- [33] Fabian Fagerholm, Alejandro Sanchez Guinea, Jay Borenstein, and Jürgen Münch. Onboarding in open source projects. IEEE Software, 31(6):54–61, 2014.
- [34] Yulin Fang and Derrick Neufeld. Understanding sustained participation in open source software projects. J. Manage. Inf. Syst., 2009.
- [35] Daniel Charles Feldman. A contingency theory of socialization. Yale University, 1976.
- [36] Karl Fogel. Producing open source software: How to run a successful free software project. " O'Reilly Media, Inc.", 2005.
- [37] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C Murphy, and Jean-Rémy Falleri. Impact of developer turnover on quality in open-source software. In Proceedings of the 2015 10th joint meeting on foundations of software engineering, pages 829–841, 2015.
- [38] Bruno S Frey et al. On the relationship between intrinsic and extrinsic work motivation. International journal of industrial organization, 15(4):427–439, 1997.
- [39] Marylène Gagné and Edward L Deci. Self-determination theory and work motivation. Journal of Organizational behavior, 26(4):331–362, 2005.

- [40] Sivana Hamer, Christian Quesada-López, and Marcelo Jenkins. How have we researched developers' contributions in software engineering? a systematic mapping study. A Systematic Mapping Study.
- [41] Christoph Hannebauer, Matthias Book, and Volker Gruhn. An exploratory study of contribution barriers experienced by newcomers to open source software projects. In Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering, pages 11–14, 2014.
- [42] Lile P. Hattori and Michele Lanza. On the nature of commits. In ASE, 2008.
- [43] Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. Research policy, 32(7):1159–1177, 2003.
- [44] Eric von Hippel and Georg von Krogh. Open source software and the “private-collective” innovation model: Issues for organization science. Organization science, 14(2):209–223, 2003.
- [45] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining GitHub. In MSR, 2014.
- [46] Barbara Kitchenham. Procedures for performing systematic reviews. Keele, UK, Keele Univ., 33:1–26, 2004.
- [47] Howard J Klein, Beth Polin, and Kyra Leigh Sutton. Specific onboarding practices for the socialization of new employees. International Journal of Selection and Assessment, 23(3):263–283, 2015.
- [48] Georg Krogh, Sebastian Spaeth, and Karim Lakhani. Community, joining, and specialization in open source software innovation: A case study. Research Policy, 32:1217–1241, 02 2003.
- [49] Raula Gaikovina Kula and Gregorio Robles. The Life and Death of Software Ecosystems, pages 97–105. Springer, 2019.
- [50] Adriaan Labuschagne and Reid Holmes. Do onboarding programs work? In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, pages 381–385. IEEE, 2015.

- [51] Karim Lakhani and Robert Wolf. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. Perspectives on Free and Open Source Software, 09 2003.
- [52] Manny M Lehman and Laszlo A Belady. Program evolution: processes of software change. Academic Press Professional, Inc., 1985.
- [53] Manny M Lehman, Dewayne E Perry, and Juan F Ramil. Implications of evolution metrics on software maintenance. In Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272), pages 208–217. IEEE, 1998.
- [54] Meir M Lehman, Juan F Ramil, Paul D Wernick, Dewayne E Perry, and Wladyslaw M Turski. Metrics and laws of software evolution-the nineties view. In Proceedings Fourth International Software Metrics Symposium, pages 20–32. IEEE, 1997.
- [55] Paul Luo Li, Amy J Ko, and Jiamin Zhu. What makes a great software engineer? In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 1, pages 700–710. IEEE, 2015.
- [56] Antonio Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: Github as a collaborative social network. In Proceedings of the international AAAI conference on web and social media, volume 8, pages 295–304, 2014.
- [57] Yuyang Liu, Ehsan Noei, and Kelly Lyons. How readme files are structured in open source java projects. Information and Software Technology (IST), 148: 106924, 2022.
- [58] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: Activity traces and personal profiles in github. In Proceedings of the 2013 conference on Computer supported cooperative work, CSCW '13, page 117–128, New York, NY, USA, 2013. Association for Computing Machinery.
- [59] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: activity traces and personal profiles in github. In Proceedings of the 2013 conference on Computer supported cooperative work, pages 117–128, 2013.

- [60] G. Mathew, A. Agrawal, and T. Menzies. Finding trends in software research. IEEE Transactions on Software Engineering, pages 1–1, 2018.
- [61] Gerardo Maturro, Karina Barrella, and Patricia Benitez. Difficulties of newcomers joining software projects already in execution. In 2017 International Conference on Computational Science and Computational Intelligence (CSCI), pages 993–998. IEEE, 2017.
- [62] Paulo Meirelles, Carlos Santos Jr, Joao Miranda, Fabio Kon, Antonio Terceiro, and Christina Chavez. A study of the relationships between source code metrics and attractiveness in free software projects. In 2010 Brazilian Symposium on Software Engineering, pages 11 – 20, 11 2010.
- [63] 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 Proceedings of the 40th International conference on software engineering, pages 1004–1015, 2018.
- [64] Justin Middleton, Emerson Murphy-Hill, Demetrius Green, Adam Meade, Roger Mayer, David White, and Steve McDonald. Which contributions predict whether developers are accepted into github teams. In Proceedings of the 15th International Conference on Mining Software Repositories, pages 403–413, 2018.
- [65] Audris Mockus. What make long term contributors: Willingness and opportunity in oss community. Proceedings - International Conference on Software Engineering, pages 518–528, 06 2012.
- [66] Elizabeth Wolfe Morrison. Newcomers’ relationships: The role of social network ties during socialization. Academy of management Journal, 45(6):1149–1160, 2002.
- [67] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. Curating github for engineered software projects. EMSE, 2016.
- [68] David R Musicant, Yuqing Ren, James A Johnson, and John Riedl. Mentoring in wikipedia: a clash of cultures. In Proceedings of the 7th International Symposium on Wikis and Open Collaboration, pages 173–182, 2011.

- [69] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki NISHINAKA, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. International Workshop on Principles of Software Evolution (IWPSE), 01 2003.
- [70] Kumiyo Nakakoji, Yunwen Ye, and Yasuhiro Yamamoto. Supporting expertise communication in developer-centered collaborative software development environments. Collaborative Software Engineering, pages 219–236, 2010.
- [71] Yunrim Park and Carlos Jensen. Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In 2009 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis, pages 3–10, 2009. doi: 10.1109/VISSOF.2009.5336433.
- [72] Yunrim Park and Carlos Jensen. Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In VISSOFT, 2009.
- [73] Raymond Paternoster, Robert Brame, Paul Mazerolle, and Alex Piquero. Using the correct statistical test for the equality of regression coefficients. Criminology, 36(4):859–866, 1998.
- [74] 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.
- [75] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and software technology, 64:1–18, 2015.
- [76] Lukas Pradel. Quantifying the ramp-up problem in software projects. In Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, pages 1–4, 2016.
- [77] Gede Artha Azriadi Prana, Christoph Treude, Ferdian Thung, Thushari Atapattu, and David Lo. Categorizing the content of github readme files. Empirical Software Engineering (EMSE), 24(3):1296–1327, 2019.
- [78] Israr Qureshi and Yulin Fang. Socialization in open source software projects: A growth mixture modeling approach. Organizational Research Methods - ORGAN RES METHODS, 13, 12 2010.

- [79] Eric Raymond. The cathedral and the bazaar. Knowledge, Technology & Policy, 12(3):23–49, 1999.
- [80] Jeffrey A Roberts, Il-Horn Hann, and Sandra A Slaughter. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects. Management science, 52(7):984–999, 2006.
- [81] Richard M Ryan and Edward L Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American psychologist, 55(1):68, 2000.
- [82] Carlos Santos, George Kuk, Fabio Kon, and John Pearson. The attraction of contributors in free and open source software projects. J. Strateg. Inf. Syst., 22(1):26–45, March 2013.
- [83] Sonali Shah. Motivation, governance, and the viability of hybrid forms in open source software development. Management Science, 52:1000–1014, 07 2006.
- [84] Gaurav G Sharma and Klaas-Jan Stol. Exploring onboarding success, organizational fit, and turnover intention of software professionals. Journal of Systems and Software, 159:110442, 2020.
- [85] 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 Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pages 421–431, 2020.
- [86] Jefferson De Oliveira Silva, Igor Scaliante Wiese, Daniel M German, Igor Fabio Steinmacher, and Marco Aurélio Gerosa. How long and how much: What to expect from summer of code participants? In 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 69–79. IEEE, 2017.
- [87] Jefferson O Silva, Igor S Wiese, Igor Steinmacher, and Marco A Gerosa. Students’ engagement in open source projects: an analysis of google summer of code. In Proceedings of the XXXI Brazilian Symposium on Software Engineering, pages 224–233, 2017.

- [88] Jefferson O Silva, Igor Wiese, Daniel M German, Christoph Treude, Marco A Gerosa, and Igor Steinmacher. Google summer of code: Student motivations and contributions. Journal of Systems and Software, 162:110487, 2020.
- [89] Susan Elliott Sim and Richard C Holt. The ramp-up problem in software projects: A case study of how software immigrants naturalize. In Proceedings of the 20th international conference on Software engineering, pages 361–370. IEEE, 1998.
- [90] I. Steinmacher, T. U. Conte, and M. A. Gerosa. Understanding and supporting the choice of an appropriate task to start with in open source software communities. In HICSS, 2015.
- [91] Igor Steinmacher and Marco Aurélio Gerosa. How to support newcomers onboarding to open source software projects. In Open Source Software: Mobile Open Source Technologies: 10th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2014, San José, Costa Rica, May 6-9, 2014. Proceedings 10, pages 199–201. Springer, 2014.
- [92] Igor Steinmacher, Marco Aurelio Gerosa, and David Redmiles. Attracting, onboarding, and retaining newcomer developers in open source software projects. In CSCW, 2014.
- [93] Igor Steinmacher, Marco Aurélio Graciotto Silva, Marco Aurelio Gerosa, and David Redmiles. A systematic literature review on the barriers faced by newcomers to open source software projects. IST, 2014.
- [94] Igor Steinmacher, Marco Aurélio Graciotto Silva, and Marco Aurélio Gerosa. Barriers faced by newcomers to open source projects: a systematic review. In Open Source Software: Mobile Open Source Technologies: 10th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2014, San José, Costa Rica, May 6-9, 2014. Proceedings 10, pages 153–163. Springer, 2014.
- [95] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David F. Redmiles. Social barriers faced by newcomers placing their first contribution in open source software projects. In CSCW, 2015.
- [96] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. Overcoming open source project entry barriers with a portal for newcomers. In ICSE, 2016.

- [97] Igor Steinmacher, Sogol Balali, Bianca Trinkenreich, Mariam Guizani, Daniel Izquierdo-Cortazar, Griselda G Cuevas Zambrano, Marco Aurelio Gerosa, and Anita Sarma. Being a mentor in open source projects. Journal of Internet Services and Applications, 12(1):1–33, 2021.
- [98] V. N. Subramanian, I. Rehman, M. Nagappan, and R. G. Kula. Analyzing first contributions on github: What do newcomers do. IEEE Software, pages 0–0, 2020.
- [99] Walter Swap, Dorothy Leonard, Mimi Shields, and Lisa Abrams. Using mentoring and storytelling to transfer knowledge in the workplace. J. of Management Information Systems, 18:95–114, 06 2001.
- [100] Xin Tan and Minghui Zhou. How to communicate when submitting patches: An empirical study of the linux kernel. Proceedings of the ACM on Human-Computer Interaction, 3(CSCW):1–26, 2019.
- [101] Xin Tan, Minghui Zhou, and Zeyu Sun. A First Look at Good First Issues on GitHub, page 398–409. Association for Computing Machinery, New York, NY, USA, 2020.
- [102] Maria Tomprou, Laura Dabbish, Robert E Kraut, and Fannie Liu. Career mentoring in online communities: Seeking and receiving advice from an online community. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pages 1–12, 2019.
- [103] Wladyslaw M Turski. Reference model for smooth growth of software systems. IEEE Transactions on Software Engineering, 22(8):599, 1996.
- [104] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In FSE, 2018.
- [105] John Eastin Van Maanen and Edgar Henry Schein. Toward a theory of organizational socialization. 1977.
- [106] Anthony J Viera, Joanne M Garrett, et al. Understanding Interobserver Agreement: The Kappa Statistic. Family Medicine, 37(5):360–363, 2005.

- [107] Giovanni Viviani and Gail C Murphy. Reflections on onboarding practices in mid-sized companies. In 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pages 83–84. IEEE, 2019.
- [108] Georg Von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W Wallin. Carrots and rainbows: Motivation and social practice in open source software development. MIS quarterly, pages 649–676, 2012.
- [109] Yunwen Ye and Kouichi Kishida. Toward an understanding of the motivation open source software developers. In Proceedings of the 25th International Conference on Software Engineering, ICSE '03, page 419–429, USA, 2003. IEEE Computer Society. ISBN 076951877X.
- [110] M. Zhou and A. Mockus. Who will stay in the floss community? modeling participant’s initial behavior. TSE, 2015.
- [111] Minghui Zhou and Audris Mockus. Growth of newcomer competence: Challenges of globalization. In FoSER, 2010.