A systematic literature review on the barriers faced by newcomers to open source software projects

Igor Steinmacher[a,b,*], Marco Aurelio Graciotto Silva[a], Marco Aurelio Gerosa[b], David F. Redmiles[c]

[a] *Computing Department*
Federal University of Technology - Parana
[b] *Computer Science Department*
University of Sao Paulo
[c] *Department of Informatics*
University of California, Irvine

Abstract

Context: Numerous open source software projects are based on volunteers' collaboration and require a continuous influx of newcomers for their continuity. Newcomers face barriers that can lead them to give up. These barriers hinder both developers willing to make a single contribution and those willing to become a project member.

Objective: This study aims to identify and classify the barriers that newcomers face when contributing to Open Source Software projects.

Method: We conducted a systematic literature review of papers reporting empirical evidence regarding the barriers that newcomers face when contributing to Open Source Software (OSS) projects. We retrieved 291 studies

*Corresponding author [Address: Rua das Cerejeiras, 60 - CEP 87301-350 - Campo Mourao–PR–Brazil – Phone +55(44)88383380]

Email addresses: igorfs@utfpr.edu.br (Igor Steinmacher), magsilva@utfpr.edu.br (Marco Aurelio Graciotto Silva), gerosa@ime.usp.br (Marco Aurelio Gerosa), redmiles@ics.uci.edu (David F. Redmiles)

by querying 4 digital libraries. Twenty studies were identified as primary. We performed a backward snowballing approach, and searched for other papers published by the authors of the selected papers to identify potential studies. Then, we used a coding approach inspired by open coding and axial coding procedures from Grounded Theory to categorize the barriers reported by the selected studies.

Results: We identified 20 studies providing empirical evidence of barriers faced by newcomers to OSS projects while making a contribution. From the analysis, we identified 15 different barriers, which we grouped into five categories: social interaction, newcomers' previous knowledge, finding a way to start, documentation, and technical hurdles. We also classified the problems with regard to their origin: newcomers, community, or product.

Conclusion: The results are useful to researchers and OSS practitioners willing to investigate or to implement tools to support newcomers. We mapped technical and non-technical barriers that hinder newcomers' first contributions. The most evidenced barriers are related to socialization, appearing in 75% (15 out of 20) of the studies analyzed, with a high focus on interactions in mailing lists (receiving answers and socialization with other members). There is a lack of in-depth studies on technical issues, such as code issues. We also noticed that the majority of the studies relied on historical data gathered from software repositories and that there was a lack of experiments and qualitative studies in this area.

Keywords: Open Source Software, Software Engineering, Newcomers, Beginners, Novices, Joining, Contribution, Barriers to Entry, Onboarding, Open Collaboration, Socialization, Systematic Literature Review

1. Introduction

Several open source software (OSS) communities rely on volunteers. According to Qureshi and Fang [33], it is essential to motivate, engage, and retain newcomers to promote a sustainable number of developers in a project. However, newcomers often face barriers for contributing to a project [12]. On many occasions, these barriers lead them to give up. Besides, as stated by Fogel [16], "if a project doesn't make a good first impression, newcomers may wait a long time before giving it a second chance." Therefore, a major challenge for OSS projects is to provide ways to reduce these barriers.

Newcomers need to learn social and technical aspects of a project before making a code contribution. They generally post their questions and request help in project forums and mailing lists or send emails to specific developers who have central roles in the project (e.g., owners and project leaders) [32, 42]. However, receiving replies that do not offer guidance or unpolished answers can result in the dropout of newcomers [38]. Lack of awareness and guidance during their first steps (setting up and choosing the right means to start with), for instance, also discourage further contributions [39]. Mainly before making their first contribution, newcomers may be susceptible to several barriers, such as expectation breakdowns, reception problems, setup misconfiguration, and learning curves. Each of these may have varying levels of importance to and impact on the overall joining process [37]. Therefore, it is important to understand the type of barriers newcomers face and their influence. This understanding is a start to making possible the creation of mechanisms and tools to reduce these barriers.

By lowering the barriers, it is expected that OSS communities will benefit

from more contributions. Studies conducted on open collaboration communities from other domains showed that it is possible to receive more contributions by lowering the entry barriers. Wikipedia has been the subject of some of these studies. For example, Faulkner et al. [15] found that modifying first time warnings prompted additional newcomer contributions. Morgan et al. [30] showed that user-friendly tools, safe spaces for newcomers, and positive interactions between newcomers and established community members are promising tools for newcomer retention.

However, to date, to the best of our knowledge, no single study has directly focused on identifying and classifying the barriers faced by newcomers to OSS projects, despite specific problems being dealt with or reported upon in several studies in the literature. Moreover, the knowledge about the barriers faced by newcomers to OSS projects is spread across the literature from different domains, such as Software Engineering, Computer Supported Cooperative Work, and Information Systems. Therefore, a systematic review [25] can aggregate in a single location the information regarding the barriers that is currently dispersed across various studies.

Thus, the objective of this research is to identify, by means of a systematic literature review, the barriers faced by newcomers when contributing to OSS projects. The primary studies were identified by querying digital libraries. We also made use of two snowballing approaches: backward snowballing, i.e., looking at the references of the papers selected from the digital libraries, and author snowballing, i.e., searching for other papers published by the authors of the selected papers.

After identifying the primary studies, we extracted the barriers empiri-

cally evidenced in the papers and classified them using a coding approach inspired by the coding procedures from Grounded Theory [9]. Using this approach, we categorized the barriers identified by type and according to their origin.

The contributions of this paper include (i) summarizing the existing evidence on barriers faced by newcomers to OSS projects and organizing the barriers into a single model, (ii) providing a quick reference for researchers interested in conducting further studies on newcomers to OSS, and (iii) providing grounded evidence of barriers and some guidelines that can be useful for OSS communities in helping newcomers. We hope that OSS communities and researchers will take advantage of this paper to better understand the barriers in their context and design strategies to address them.

The remainder of this paper is organized as follows. The protocol of the systematic review is presented in Section 2. In Section 3, we characterize the projects considered by each study. In Sections 4 and 5, we report the results of the analysis of the selected studies, including the classification via the type of barrier and the origin of the barrier. A discussion is presented in Section 6 and threats to validity are presented in Section 7. Finally, conclusions and further work are presented in Section 8.

2. Research method

This study has been undertaken as a systematic literature review (SLR) based upon guidelines established for the Software Engineering domain [25, 27, 26]. In this section, we provide the protocol used in the SLR, specify the research question and its components, and establish the requirements regarding the source and primary study selection, the evidence collection, and the method of synthesis of such evidences. The results regarding each step are provided alongside the protocol, except for evidence extraction and analysis, which are addressed in Section 4.

2.1. Research questions

Newcomers present different technical skills, time availability, and reasons for joining an OSS project. Notwithstanding, when developers decide to support an OSS project, they need to learn social and technical aspects of the project before making a contribution. During this learning period, newcomers face barriers that can result in their decision to give up contributing. Our main goal is to identify the barriers that are faced by newcomers to make their contributions. We expect that by reducing the barriers, projects can benefit from both more occasional contributions and more long-term contributors. Thus, we have defined the following as our main research question:

- **RQ 1.** What are the barriers that hinder the contribution of newcomers in OSS projects?

By answering this question, we aim to capture barriers that a newcomer faces when contributing in an OSS project. We are not interested in newcomers' motivations for contributing to a project but in the issues they may

face after deciding to contribute to a project. We also make no distinction regarding the size, quantity, or frequency of contributions made by newcomers.

When dealing with OSS projects, we are aware that there are different ways newcomers can start to contribute, including translation, bug triage, bug reporting, user support, and coding. In our current study, we focus only on source code contributions. Therefore, we define newcomers as developers who want to make their first code contribution to a project. Moreover, when we refer to OSS projects, we are talking about community-based open source projects [8] and open collaboration communities [17]. Community-based OSS projects rely on the efforts of volunteers to be maintained. In these projects, mailing lists, issue trackers, and source code (in versioning systems) are publicly available. Any skilled person who wants to contribute can get the current code, choose (or report) an issue, address it, and submit a patch to be included in the product.

2.2. Criteria for selection of studies

The search process encompassed two approaches. The first one was based on queries in digital libraries. However, as reported in the literature, using just this approach, especially for systematic reviews in software engineering, is often inefficient. As suggested by others [22, 26], we used a single step citation analysis (snowballing), and a single step author snowballing to complement the search process. Details about the way we used such approaches are described in the following subsections.

Regardless of the mechanism used to search, the studies were screened according to various criteria pertinent to the research question. We estab-

lished the following criteria for the inclusion of studies. Regarding a paper, it must be available as a full paper, written in English, and published in a peer-reviewed venue, including workshops, conferences, and journals. As for the studies reported by the papers, they must report barriers faced by newcomers to contribute to open source software projects and present empirical evidence.

The exclusion criteria consisted of removing duplicate results and excluding papers regarding newcomers or open source software projects that were not within the scope of this research. Papers that were clearly duplicated or for which we found newer and more complete versions were excluded. Regarding studies not relevant to the purpose of this review, we excluded studies regarding newcomers, but not to open source software; studies about open source software that do not study newcomers; and studies that do not provide empirical evidence (studies that present just methods or unevaluated approaches/tools).

The selection process consisted of the following steps:

1. Search in digital libraries: we queried digital libraries, and the references of the retrieved studies were stored in a local repository to be further analyzed.
2. Title, abstract, and keywords analysis: titles, abstracts, and keywords were read to verify which studies met the inclusion and exclusion criteria.
3. Author snowballing: studies found and selected by the search in the digital libraries were analyzed regarding their authors. We searched for other papers published by the same authors. Candidate papers were

submitted to the same process used for papers found in digital libraries: title, abstract, and keywords analysis (step 2).

4. Introduction and conclusion analysis: the initial and closing sections of the studies were evaluated regarding their objectives and results. This analysis enabled the researchers to further verify if the papers answered the research question and met all inclusion and/or any exclusion criteria. When the reading of the opening and closing sections was inconclusive, the entire paper was read to decide on its inclusion or exclusion.
5. Backward snowballing sampling: studies found and selected by the search in the digital libraries and by author snowballing were analyzed regarding their references. Candidate papers were submitted to the same process used for papers found in digital libraries and by author snowballing: screening by title, abstract, and keywords, followed by analysis of the introduction and conclusion. We ran just one level of citation snowballing.
6. Full paper reading: finally, after selecting a paper by its introduction and conclusion, the entire paper was read to decide on its inclusion or exclusion.

It is important to note that for every step, more than one researcher read each paper independently. For conflicting evaluations, researchers further discussed the paper to reach a consensus. In cases where there was no consensus or there was doubt, the study was included to avoid premature exclusion.

2.2.1. Search in digital libraries

The mechanism available for searching in most digital libraries is based upon textual search expressions. Thus, its definition is crucial for the effectiveness of the searching step and the systematic review as a whole. A common approach regarding the establishment of the search expression is its characterization based on PICO components: population, intervention, control, and outcome [3]. Based upon previous studies, we knew that most papers on the subject were case studies that employed quantitative, qualitative, or mixed methods. Thus, establishing a control element was not feasible. It is worth noticing that recent guidelines on systematic review for Software Engineering are also oriented to omit this characterization [26]. Based upon these recommendations, we have kept the population and intervention components, and omitted comparison and outcome.

A systematic review for Software Engineering establishes the problem as the target of the systematic review and the intervention as what can be observed in the context of the systematic review [3]. Considering such definitions, we initially established the population as open source projects and the intervention as contributions of newcomers.

After analyzing the terms related to the population and intervention, we derived the keywords and synonyms presented in Table 1 to establish the search expression. The synonyms were suggested by experts and extracted from papers we used as a control of this study [42, 13, 10, 21, 23].

Using the terms listed in Table 1, we defined a generic search expression. Considering a set of known studies (the control group) we had previously established, we performed several trials and iterations until coming to a fi-

Table 1: Terms used to build the queries (terms marked with an * were dismissed in the final query).

Keyword	**Synonyms**
Open Source Software	OSS, Open Source, Free Software, FLOSS, FOSS
Newcomers	newcomer, newbie, new developer, new member, new contributor, new people*, novice, beginner, potential participant, joiner, new committer*
Contribution	joining*, retention, first steps*, entrance*, initial steps*, joining process, onboarding, contributing*

nal search expression. Some synonyms, such as "new people," "entrance," and "new committer," were removed, as we could either not find any paper with such terms or the search result would find too many studies that, by skimming the titles, were considered out of the scope of this systematic review. Furthermore, we also evaluated whether the search expression was still effective, *i.e.*, whether it recovered the studies established as control.

```
(
    ("OSS" OR "Open Source" OR "Free Software" OR FLOSS OR FOSS)
    AND
    (newcomer OR "joining process" OR newbie OR "new developer" OR "new member" OR "new
    contributor" OR novice OR beginner OR "potential participant" OR
    retention OR joiner OR onboarding OR "new committer")
)
```

After defining the search expression, we defined criteria for the selection of digital libraries. The digital libraries should index papers on open source software research written in English, support searching using Boolean expressions, and provide access to the complete text of the paper. Based upon these criteria, we selected four digital libraries to conduct our search: ACM

Digital Library[1], IEEE Digital Library[2], Scopus[3], and Springer Link[4].

Moreover, we consulted several experts regarding recommended venues for studies relevant to this research. The specialists provided a set of conferences, workshops, journals, and specialized websites. Most of them were indexed by the digital libraries we selected. The suggested venues that were not indexed by the preselected digital libraries were excluded due to other source selection criteria. For instance, specialists' indications of conferences in languages other than English and websites that do not support searching using Boolean expressions were excluded.

2.2.2. Snowballing sampling

In addition to querying the digital libraries, we conducted a single step snowballing sampling following two approaches. The first approach consisted of checking if the authors of the selected studies published other relevant studies that we could not retrieve from the digital libraries. To find other publications, we accessed authors' profiles in ACM and IEEE libraries and checked their DBLP[5] profile and personal homepages (when available).

The second approach was backward snowballing sampling. We analyzed the reference lists of all selected papers to find other possible relevant studies. Just one level of snowballing sampling was conducted, *i.e.*, we did not apply snowballing sampling for studies found by a previous snowballing process.

[1] `http://dl.acm.org/`

[2] `http://ieeexplore.ieee.org`

[3] `http://www.scopus.com/`

[4] `http://link.springer.com/`

[5] `http://www.informatik.uni-trier.de/~ley/db/`

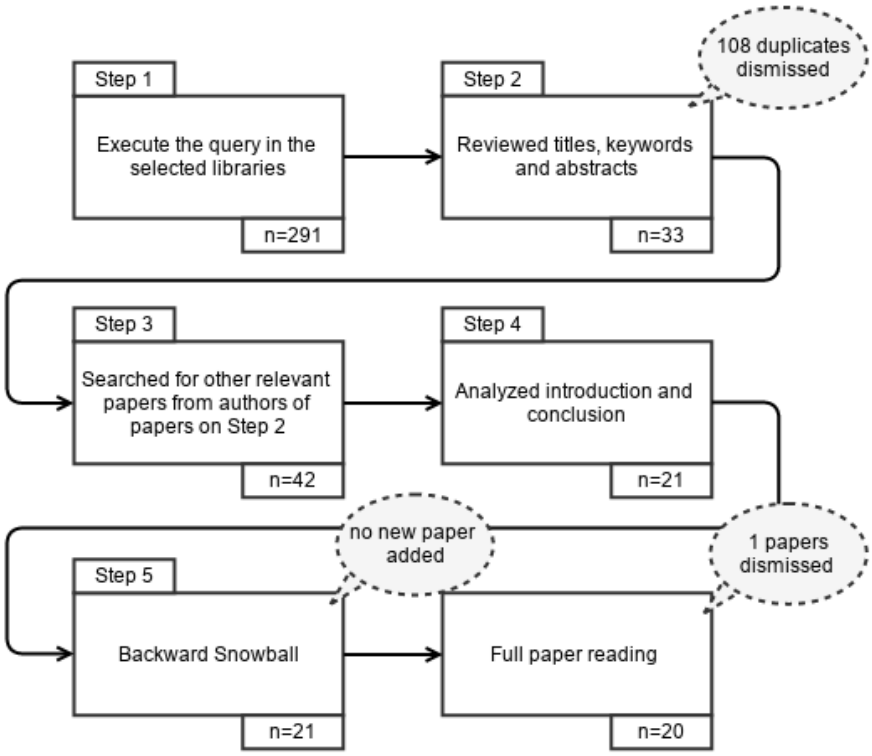


Figure 1: Paper selection process.

2.3. Study selection

We used the query presented in Section 2.2.1 to retrieve the candidate papers from the digital library systems in April 2013, as shown in Fig. 1. We retrieved 291 papers by searching and 42 other papers by snowballing, totaling 333 papers. After applying the selection criteria, we selected 20 papers for data extraction and analysis. In the following, we present the detailed results of executing the steps presented in Section 2.2 and its correlated Fig. 1.

Search in digital libraries. As a result, we gathered 291 candidate papers: 59 from IEEE, 84 from ACM, 132 from Scopus, and 16 from Springer

Link.

Title, abstract, and keywords analysis. For the first step, the titles, abstracts and keywords were independently analyzed by two researchers. During this step, the researchers identified and dismissed 96 (32.99%) duplicates gathered from different libraries. From the 195 remaining entries, 33 papers were selected after a consensus meeting.

Author snowballing. We applied author snowballing on the selected papers. We checked other papers published by the authors of these 33 studies, and we found 20 other candidate papers. After analyzing the abstracts of these papers, we selected nine relevant papers, bringing us to 42 candidate papers.

Introduction and conclusion analysis. All 42 papers had their introductions and conclusions analyzed by the researchers. Among them, 21 were dismissed. We did not find the full text of three papers; two papers were extended abstracts; two were proposals to thesis/dissertation workshops; 10 did not study or present evidence of barriers faced by newcomers (including three papers that only presented tools with no evaluation); and four papers were clearly not presenting barriers faced by newcomers to OSS.

Backward snowballing sampling. After reading the introduction and conclusion sections, we conducted a backward snowballing sampling on the 21 remaining papers, looking for papers cited by the selected studies. However, no additional paper was selected by this approach. An important thing to note regarding the snowballing sampling is that the previously selected studies appeared repeatedly, especially [42, 13, 5, 11].

After the snowballing, the papers had their full content read by the re-

Table 2: Papers found and included per source.

Source	Found	Relevant Studies	% Search Efficacy
IEEE	59	10	16.95%
ACM	84	9	10.71%
Scopus	132	8	6.06%
Springer Link	16	3	12.50%
Sum	291	30	10.31%
Repeated	*96*	*12*	
Subtotal	*195*	*18*	
Authors*	20	2	
Citation Analysis*	22	0**	
Total		*20*	

* Values presented in the column "Found" represent the number of papers considered after title analysis.

** 12 papers previously selected were found during citation analysis.

searchers. One study was discarded because it was related to user support Q&A forums and not source code contribution. Therefore, we extracted and analyzed data from 20 selected studies.

2.3.1. Study Selection Summary

We summarize the total papers found and selected by each source and approach in Table 2. In the table, it is possible to check how the studies are distributed according to their source. Scopus returned the largest number of hits, but its precision was the lowest amongst the sources we used. Meanwhile, IEEE, ACM and Springer Link returned fewer hits but achieved higher levels of efficacy. This can be explained because Scopus is a meta-search that

indexes the literature from different publishers, including ones not related to Computing. We can see that Scopus, ACM and IEEE would independently retrieve more than 40% of the primary studies, with several studies available simultaneously on more than one of them (duplicated).

When a duplicate paper was found, sources that actually publish the papers were prioritized over those that just index articles from other publishers. We analyzed the duplicates to verify where the overlaps among papers were. The results are shown in Table 3. It is possible to check that the Scopus meta-search retrieved 81 papers duplicated from the other 3 sources. We also noticed that ACM and IEEE index some venues together, such as ICSE (International Conference on Software Engineering), which is a co-published conference. Moreover, ACM also indexes some publications from Springer Link; in this systematic review, 5 papers from Springer Link were also gathered from the ACM Digital Library.

Table 3: Analysis of Duplicate Source.

Sources	**Duplicates**
ACM & Scopus	43
IEEE & Scopus	16
ACM, IEEE & Scopus	15
ACM & IEEE	6
ACM, Springer & Scopus	5
Springer & Scopus	2
Paper in different venues (Scopus)	1

Regarding the reasons for excluding papers, we account for them in Table 4. Once a paper fit any exclusion criterion or did not comply with any

inclusion criteria, it was excluded. For instance, 113 papers were excluded because they were not related to OSS. Meanwhile, from those that addressed OSS, 50 papers did not address newcomer issues. It is important to note that we just considered one reason of exclusion per paper in Table 4, but one paper could actually be classified under more than one exclusion reason in the selection process.

Table 4: Excluded papers and exclusion reasoning.

Exclusion reason	**Excluded**	**Studies**
		333
Duplicate	108	225
Not in English	1	224
Invalid type	29	195
Full paper not available	3	192
Not related to OSS nor newcomers	101	91
Related to newcomers but not to OSS	12	79
Related to OSS but not tonewcomers	50	29
Previous version of a more complete study	4	25
Not related to source code contributions	1	25
Does not present empirical study	6	**20**

The 20 selected studies and their respective identifiers are presented in Table 5. Throughout the paper, we will use the identifiers in the format PS? for the papers selected for analysis in the systematic review.

Table 5: Selected Primary Studies.

Identifier	Reference	Identifier	Reference
[PS1]	Ben et al. [2]	[PS11]	Park and Jensen [32]
[PS2]	Bird [4]	[PS12]	Qureshi and Fang [33]
[PS3]	Bird et al. [5]	[PS13]	Schilling et al. [35]
[PS4]	Canfora et al. [6]	[PS14]	Steinmacher et al. [39]
[PS5]	Capiluppi and Michlmayr [7]	[PS15]	Steinmacher et al. [38]
[PS6]	Cubranic et al. [11]	[PS16]	Stol et al. [40]
[PS7]	Ducheneaut [13]	[PS17]	Von Krogh et al. [42]
[PS8]	He et al. [21]	[PS18]	Weiss et al. [45]
[PS9]	Jensen et al. [23]	[PS19]	Zhou and Mockus [46]
[PS10]	Midha et al. [29]	[PS20]	Zhou and Mockus [47]

2.4. Overview of the studies

While reading the papers in full, we collected some data regarding the type of study conducted, and type of data analyzed. The data analyzed by the selected studies are predominantly gathered from source code repositories (12 studies), mailing list archives (12 studies), and bug/issue trackers (8 studies). Other data used by the studies include questionnaires, interviews with developers, observations, and project profiles. Fourteen studies used more than one data source in their analysis. An interesting fact here is that the main method used to find issues faced by newcomers was the analysis of historical data. Only seven studies report interviews, observations, or questionnaire results.

Most of the papers reported case studies and gave a quantitative analysis

of collected data. We found that 19 studies (95%) used the case study method to evidence the problems. Only one study conducted both an experiment and a case study, and another one conducted only an experiment. In terms of the type of analysis, 13 studies (65%) analyzed the data using only quantitative approaches, as depicted in Fig. 2. Two studies (10%) used only qualitative methods to analyze the data, and four studies (20%) used both quantitative and qualitative analysis to evidence the barriers.

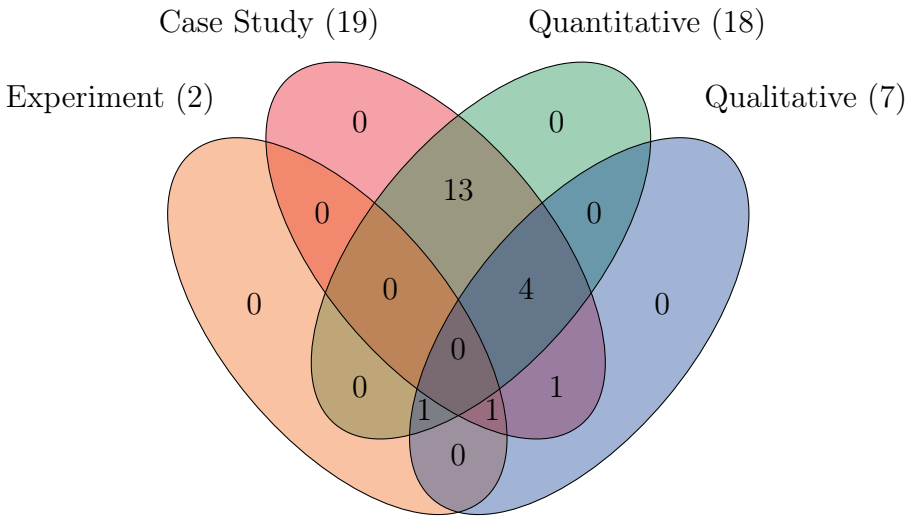


Figure 2: Type of analysis conducted in experiments and case studies.

It is possible to see the lack of studies conducting qualitative analysis as supporting the existence of problems that hinder newcomers' contributions. Quantitatively analyzing historical data can bring highlights of the barriers faced by newcomers, but conducting qualitative analysis can enrich the evidence, reveal new facts, and help in finding the issues faced by newcomers. There is still room available for studies based on observations, interviews and

experiments that can help reveal the barriers faced in practice.

We also analyzed the primary studies to check how the publications are distributed in time. Fig. 3 depicts this distribution. We can see that the topic appeared in 2003, and just a few papers appeared from then until 2010. The last 3 years contain more than 50% of the relevant publications for this review (11 out of 20). It is important to note that the searches were performed in April 2013, so it is possible that more papers were published in 2012 and 2013 and were not indexed so far.

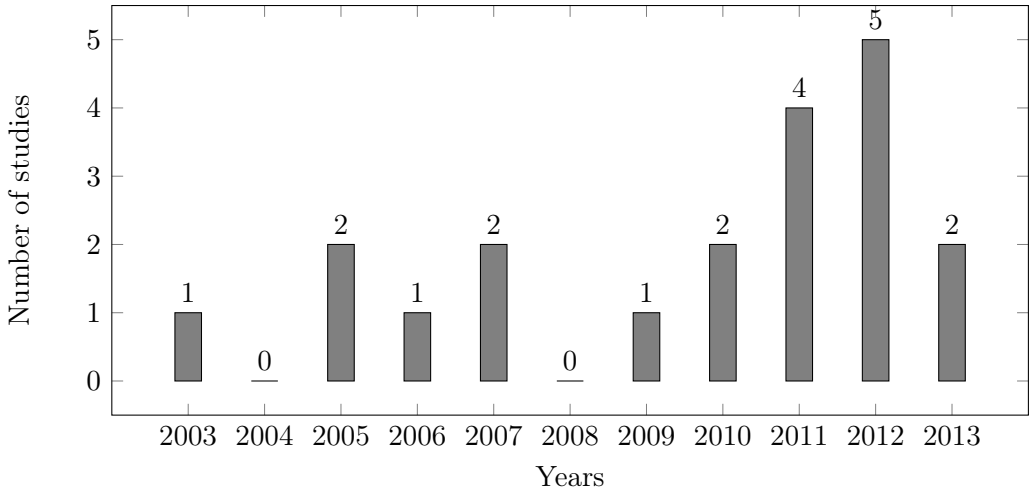


Figure 3: Temporal view of the publications (queries from April 2013).

Out of the 20 papers considered relevant, 5 were published in journals, and 15 in conference proceedings. The primary studies are spread in different forums. Software Engineering is the area with the highest number of studies published (9 out of 20). Two articles were published in Management journals. Forums of CSCW, Information System, Visualization and Green Computing also appeared.

2.5. Data extraction

Given the set of primary studies, the researchers read the documents in full and extracted the necessary data. First, we created a list of barriers that were evidenced by each paper. Each barrier was related to information about the study and the type of evidence that was used to indicate it. After this identification, each barrier was linked to text segments that supported it in the papers they were identified in.

Using the text segments, we classified the barriers applying an approach inspired on open coding and axial coding procedures from Grounded Theory (GT) [9]. Although the purpose of the GT method is the construction of substantive theories, according to Corbin and Strauss [9], the researcher may use only some of its procedures to meet one's research goals.

2.6. Synthesis

Based upon the extracted data, we synthesized the barriers identified by the coding as a set of contribution barriers faced by newcomers in open source software projects. A hierarchical classification was established for the barriers, which was represented as a mind map. Such classification, which is presented and detailed in Section 4, was used to answer the primary research question of this study. In the next section, we present a characterization of the projects considered in the primary studies.

3. Projects analyzed in the selected studies

Before analyzing the studies regarding barriers found by newcomers when contributing source code, it is important to characterize the projects considered by each study. After all, we are interested in a diverse and representative population of projects. There were 28 projects under investigation, plus studies that analyzed large sets of projects from Apache Foundation and SourceForge.net. In Fig. 4 we present a summary of the number of projects analyzed per selected study. Most of the studies (12) analyzed three or less projects. Only four studies used a set with more than seven projects from a given software forge: three of them analyzed subsets from SourceForge.net projects, and one analyzed projects from Apache Foundation. The remaining three studies considered 4, 5, and 7 projects, respectively.

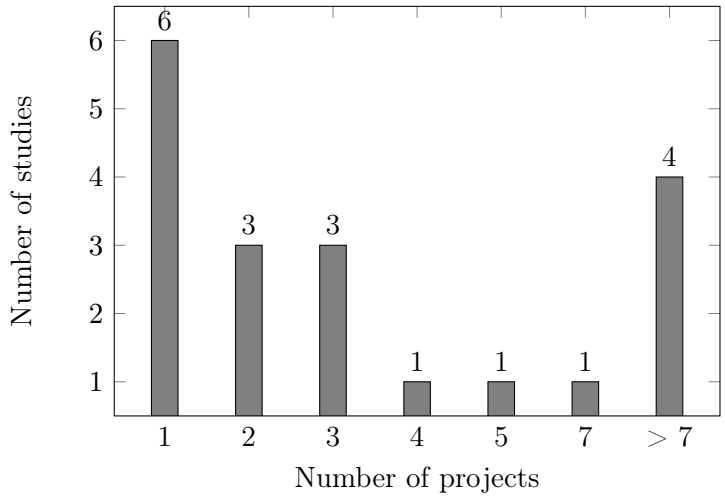


Figure 4: Number of projects analyzed in the selected studies.

Regarding the projects analyzed, as presented in Fig. 5, six projects appeared in more than one study: Apache httpd (3 studies), PostgreSQL (4

studies), Python (4 studies), Mozilla (2 studies), Eclipse (2 studies), and Jboss (2 studies). Another 22 projects appeared in the studies, of which 2 projects were not disclosed by the authors [PS14]. We did not consider the four studies that focus on forges [PS8, PS10, PS12, PS19], which can include some of the projects that appeared as objects of other studies.

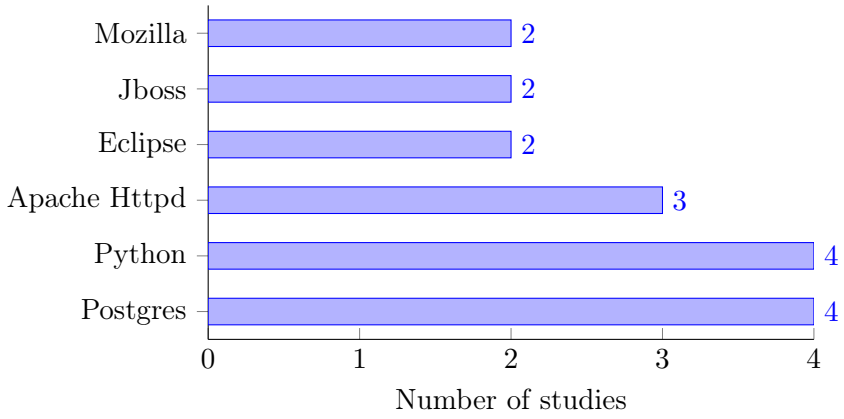


Figure 5: Projects most analyzed and the number of studies that used them as objects of study.

To verify what types of projects were analyzed by these studies, we gathered various characteristics of the projects: activity level (very high, high, low, or very low), number of developers, size in lines of code (LOC), language, domain, typical user community, and age. These data are presented in Table 6. The data were gathered from OpenHub when available. For projects that were not available on this platform, we extracted the data manually from the project repository when available. We did not find information available for three projects: Arla, AVIS, and MeDiCi.

Considering the projects for which we gathered data, almost all of the projects analyzed (22 out of 23) had more than 5 years of development and,

Table 6: Characteristics of the studied projects (data gathered on Sept. 10, 2013).

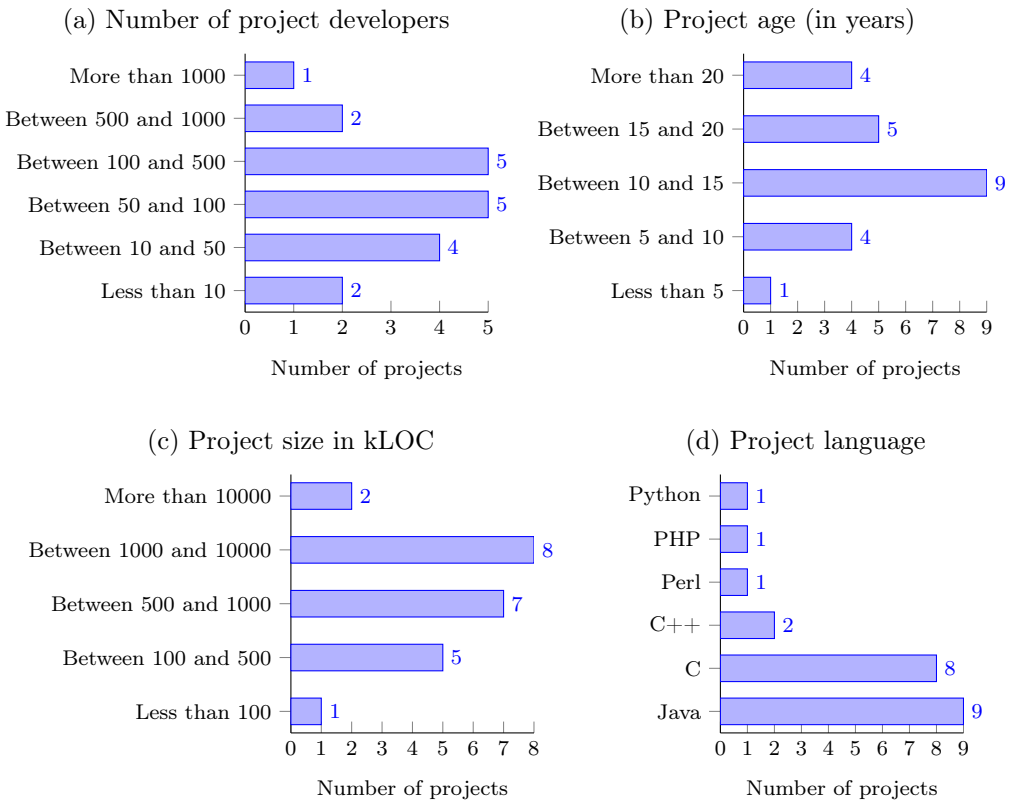
Project	Activity	# of dev	LoC	Language	Domain	Final Users	Age
Arla	NO INFO				Filesystem	Technical	
AVID	NO INFO				Visualization Tool	Technical	
CraftBukkit	High	51	67,774	Java	Game Server	General Audience	2 years
Eclipse	Very High	115	2,679,475	Java	IDE	Technical	12 years
FileZilla	Very Low	15	250,726	C++	Network system	General Audience	12 years
FreeBSD	Very High	204	5,013,437	C	Kernel	Technical	> 20 years
Freenet	Very High	174	439,184	Java	Network system	Technical	13 years
Gantt	High	27	122,261	Java	Proj Management	General Audience	10 years
GIMP	High	70	731,486	C	Image Handling	Specific	16 years
Gnome	Very High	1053	8,024,516	C	GUI	General Audience	16 years
Hackystat	Very Low	1	296,225	Java	Framework	Technical	6 years
Hadoop	Very High	43	2,277,110	Java	Framework	Technical	7 years
Httpd	Very High	33	22,405,508	C++	HTTP Server	Technical	17 years
JBoss #	Very High	133	752,477	Java	Application Server	Technical	14 years
Jonas	High	12	9,873,057	Java	Application Server	Technical	8 years
KDE	Very High	663	23,343,508	C++	GUI	General Audience	16 years
MediaWiki	Very High	172	963,843	PHP	Wiki	General Audience	10 years
MeDiCi	NO INFO						
Mozilla	Very High	966	9,326,438	C	Browser	General Audience	11 years
Parrot	Very High	31	283,895	Perl	Virtual Machine	Technical	12 years
Postgres	Moderate	23	685,332	C	Database Server	Technical	17 years
Python	Very High	64	866,177	Python	Prog Language	Technical	> 20 years
Samba	Very High	66	1,445,110	C	File System	Technical	> 20 years
ServiceMix	Very High	7	626,483	Java	Integration Container	Technical	5 ears
Subversion	Very High	37	564,579	C	SCM	Technical	13 years
Wine	Very High	85	2,550,965	C	Desktop Emulator	General Audience	> 20 years

JBoss is now called WildFly

according to the classification employed by OpenHub and adopted by others [31], were considered Very Old projects. Seventeen of the 23 projects had more than 500 KLoC, standing in the top 10% of projects indexed by OpenHub in terms of lines of code (according to the data reported by Nagappan et al. [31]). For the number of developers, 13 out of the 23 projects have more than 50 developers, a number achieved by only 2.6% of the projects analyzed by Nagappan et al. [31]. In addition, we can see in Fig. 6 (d) that 17 of the projects studied were written in either Java or C.

Despite the small number of projects, we found a high diversity of projects. It also seems that most of the authors looked for mature, well-established

Figure 6: Summary of projects characteristics.



projects to conduct their studies. However, most of the projects chosen (66.67%) were products used during the development cycle (Application Servers, Frameworks, IDE, etc.). The remaining projects (33.33%) had, as their final users, the general (or 'non-developer') public. The higher focus on established, large projects that focus on products used during the development cycle can introduce a bias in the results of the studies analyzed.

4. Contribution barriers

The main purpose of this systematic review was to find the barriers faced by newcomers to OSS projects as evidenced by the literature. For each selected study, we analyzed any barrier reported that was empirically identified or evaluated. The identification and classification of such barriers were accomplished by using a coding approach based on procedures of Grounded Theory as described in Section 2.5. The result of the categorization is shown in Fig. 7. The figure presents five categories of barriers: Social Interactions, Newcomers' Previous Knowledge, Finding a Way to Start, Documentation, and Technical Hurdles.

In the following subsections we detail each of the categories of barriers found. Each of the categories and barriers were reported to give a sense of what each of them represents independently.

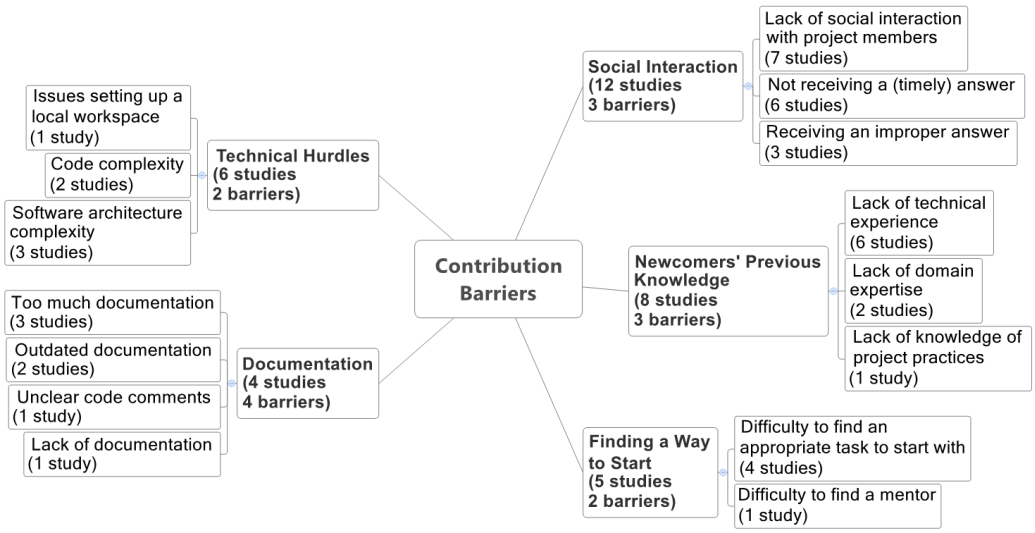


Figure 7: Barriers evidenced in the literature.

4.1. Social interaction

This category represents the barriers related to the manner in which newcomers interact with the community members interact, including issues related to who the members they exchange messages with are, the size of newcomers' contact network, and how the community communicates with them. This category is the most evidenced among the selected studies, appearing in 12 studies (60%). Within this category, we found evidence of three different barriers that can influence newcomers: lack of social interaction with project members; not receiving a (timely) answer; and receiving an improper answer.

Lack of social interaction with project members. This category represents the studies that evidence role, centrality, and the size of newcomers' social network as barriers that can influence their success in long-term contribution. Ducheneaut [PS7] analyzed the mailing list archives of the Python project. He made an in-depth analysis of the socialization history of one successful newcomer in this community. Based on this individual, the author identified a set of socialization activities that contributed to his success in the project. He highlighted the influence of social and political organization for newcomers willing to become core developers, emphasizing the need to build an identity in the project: *"what the newcomer has to learn is how to participate and how to build an identity that will help get his ideas accepted and integrated."* He also reported on the importance of political support to becoming a core member: *"while proposing sound technical solutions to problems is an important aspect of Fred's successful participation, these solutions are not enough by themselves: establishing strategic links with key members of the project beforehand is what truly allows them to be respected*

and accepted."

The social status and the need to build an identity is quantitatively supported by the evidence presented by Bird [PS2]. He analyzed data from mailing lists, source code repository histories, and bug tracking databases to understand how successful OSS projects work. To check the social status of newcomers, he analyzed the indegree social network metric. This metric represents the breadth of response to an individual and, thus, status within the community. Using statistical tools, he found that *"the social network measure, indegree, ... had a significant effect on immigration,"* which presented very significant influence in the three projects analyzed.

More quantitative evidence on the effect of the socialization was presented by He et al. [PS8]. They conducted a social network analysis on SourceForge.net projects, looking at the centrality of newcomers' nodes. They found that newcomers tend to collaborate more with existing members than with other newcomers *"to enlarge his or her influence and position, because the veterans often have more important position and richer experiences."* Zhou and Mockus [PS20] also found evidence that social interaction is correlated with successful newcomers. They did not look at the centrality of the members that the newcomers interacted with but instead at the size of the newcomers' peer groups. They quantitatively analyzed the mailing lists of the Mozilla and GNOME projects and reported that *"the attributes of her peer group, in particular, its social clustering and productivity significantly influence her opportunity to become a Long Term Contributor."*

Other research in this direction was conducted by Qureshi and Fang [PS12]. A little different from previously presented studies, these authors

were looking for different socialization patterns. They used the number of interactions between newcomers and core members during a time period to quantitatively analyze social relationships of newcomers. They analyzed trajectories of 133 newcomers in 40 projects from the moment they joined. They identified four distinct classes of newcomer behavior, considering their initial amount of interactions with core members and the growth of these interactions, and reported that *"individual joiners begin with different initial levels and follow different growth patterns, suggesting the existence of heterogeneity in the socialization trajectories."* By analyzing these different patterns, they evidenced that *"... it is important to recognize that socialization with core developers has a significant impact on joiners' status progression... it is perhaps more important to achieve a high level of interaction as quickly as possible."*

We can see that all studies that analyzed the importance of social interactions show a correlation between the centrality of newcomers' social relationships and newcomers' successful permanence as a contributor. However, there is no clear evidence of the causal relationship between social network centrality and newcomer success. We found evidence of the influence of this barrier in seven of the primary studies [PS2, PS3, PS7, PS8, PS12, PS19, PS20].

All of them report results from case studies, and only one brought qualitative evidence of social interaction influencing the contribution process [PS7]. All studies relied on historical information from mailing lists, issue trackers and source code versioning systems. Even Ducheneaut [PS7] analyzed historical material to conduct his 'computer-aided ethnography' research. It is

necessary to conduct more in-depth interviews and mixed method research to understand the reasons behind the observations made in these studies.

The level of interaction needed to achieve recognition or become a contributor of a project can vary according to the project, due to the heterogeneity of OSS projects. This is a limitation for most of the studies, since five studies drew their conclusions based on analysis of three or less OSS projects [PS2, PS3, PS7, PS19, PS20]. One of the studies [PS12] analyzed the socialization level quantitatively and determined that there are different types of interaction behavior that result in long-term contribution. Even conducting this study, by far, there is still a gap in understanding the relationship between project characteristics and the influence of interaction with core members on becoming a long-term contributor, which is indicated in some of the studies [PS2, PS3, PS7]. Characteristics such as project age, community structure, governance model and existence of sponsors can influence the impact of such barriers.

Receiving an improper answer. The answers received from the community play an important role when a newcomer is trying to contribute. Newcomers can feel unwelcome or insulted if they are not answered politely or positively. Three studies [PS9, PS15, PS16] brought evidence of the negative impact of the content of answers received by newcomers in their first interactions.

By qualitatively analyzing mailing list messages from three different projects, Jensen et al. [PS9] found impolite replies to newcomers as a potential problem. They report that *"1.5% of newbie replies were rated as rude/profane ... flaming was more common than we expected, and the potential negative*

effects of such behavior could be significant. . . ." Steinmacher et al. [PS15] also conducted a qualitative analysis on mailing list messages and drew similar conclusions as those reported by Jensen et al. In addition, they conducted a survey with the dropouts and reported that *"from the 13 respondents who intended to contribute to the project, six sent answers related to reception."* The authors then concluded that *"receiving inadequate answers and the [level of] experience of the respondent affect the decision of newcomers."* In a study not focused on barriers to contribution, Stol et al. [PS16] reported on feedback from a student that dealt with an OSS project and evidenced that *"the community's response was not considered to be very helpful."*

From the studies that reported issues regarding receiving improper answers, two of them sourced their evidence from a qualitative analysis of the mailing list history [PS9, PS15]; the other relied on feedback obtained from semi-structured interviews with students aiming to identify architectural patterns in OSS projects [PS16]. Although their goal was not making a contribution, the subjects performed steps that are similar to those performed by newcomers willing to make their first contribution to a project.

Generally, newcomers demand attention and friendly hands to start contributing because *"humans need attention from other people, and developers are no exception"* [PS20]. There are cases in which an improper answer or the lack of an answer can play a role in the decision of a newcomer to contribute or not. Therefore, the use of in-depth interviews and more contextualized research (such as observation or ethnography) need to be conducted to understand the reasons behind the observations made in these studies. There are other cases in which newcomers will keep trying to step on board be-

cause people with different profile, skills, cultures, and motivations can feel and attack this problem differently.

Not receiving a (timely) answer. Just like the barrier presented beforehand, newcomers can be demotivated or made to feel unimportant when they do not receive a timely answer. Five studies [PS9, PS15, PS16, PS17, PS20] reported on the impact of not receiving a timely answer from the community as a problem that can impact future participation in the project.

Jensen et al. [PS9] used mailing list history to quantitatively evidence that *"nearly 80% of newbie posts received replies, and that receiving timely responses, especially within 48 hours, was positively correlated with future participation."* Another piece of quantitative evidence was given by Zhou and Mockus [PS20]. They analyzed the time until newcomers receive the first answer and found that *"low attention in the form of too rapid response would reduce the odds [to become a long term contributor] by 28% in Mozilla and by 39% in Gnome."*

As part of a broader research, von Krogh et al.[PS17] analyzed the mailing list history of the Freenet project and found quantitative evidence that *"a high 78% of the population of development list participants attempted to initiate dialogue, via starting a new thread, at least once. Of these attempts only 29 (10.5%) participants did not receive any reply to their initial posting and subsequently did not appear on the developer list again."*

In a qualitative analysis of a debrief session conducted with newcomers that contributed to OSS projects, Steinmacher et al. [PS14] found that *"many demotivating facts that occurred. . . : emails not answered after a week could make the group withdraw; . . . a message posted in a forum to announce a new*

translation was not read and resulted in concurrent work and wasted time." This provided some qualitative, contextualized evidence on how the lack of an answer can impact a newcomer's contribution in an OSS project. Stol et al. [PS16] also reported on a debrief from students: *"the community had not replied (yet) on their request for feedback."*

A contradictory result was presented in a more recent study by Steinmacher et al. [PS15]. They conducted a case study using the mailing list and issue tracker history of the Hadoop Project and, interestingly, found that *"the lack of answer was not evidenced as a problem that influences newcomers' decision to remain or abandon the project."* This finding is opposite that of the other studies presented. They found that the dropout rate for newcomers that did not receive answers was not worse than when newcomers were answered.

There are both quantitative and qualitative results that support these results. However, the barrier still needs to be further explored. The quantitative evidence found relies only on mailing list histories from a few projects and is hard to generalize due to the already mentioned high heterogeneity of OSS projects. In addition, both studies that had qualitative results gathered feedback from students.

4.2. Newcomers' Previous Knowledge

This category consists of the barriers related to the experience of newcomers regarding the project and the manner in which they show this experience when joining the projects. We classified the barriers found into lack of domain expertise, lack of technical expertise, and lack of knowledge of project practices.

Lack of domain expertise. When working on a software project, it is important for the developers to have some previous knowledge of the application domain. It is no different in OSS projects, and, according to Gacek and Arief [18], the people who contribute code to an open source project are always users of the product.

This is evidenced by Von Krogh et al. [PS17], who claimed that new features added by newcomers *"emerge from the newcomers' prior domain knowledge ..."* Stol et al. [PS16] evidenced it as a barrier to start in a project, as the subjects of their research *"reported their unfamiliarity with the domain to be a hindrance."*

The evidence found in the literature suggests that previous domain knowledge can increase the odds of a successful joining. However, we found no study that focused specifically on verifying the influence of this variable in the success of a newcomer. This is a suitable object of study for future research.

Lack of technical expertise. To become a developer, an individual must have (or acquire) some project-specific technical skills. Schilling et al. [PS13] studied a set of newcomers to KDE. By analyzing newcomers' previous knowledge the authors found that *"...level of practical development experience is strongly associated with their continued permanence."* Another interesting finding is that knowledge obtained via academic education is not significantly associated with successful contributions. A possible explanation for this result is that quantity (years of education) does not assure quality [36].

Several studies provide more evidence towards technical expertise and

its importance for successful contributions by measuring software artifacts developed by newcomers. For instance, some studies reported that sending messages or patches to a mailing list or issue tracker presenting previous technical skills can benefit the newcomer when joining. Stol et al. [PS16] evidenced that *"when newcomers mentioned that they had already tried some options to fix their problem and have put efforts to look for a solution in the forums ... then the responders were quick to respond and were very helpful."* They explain it as *"a message of legitimacy from the newcomer along the lines of 'I have done my homework, can I get some help now." '*

Von Krogh et al. [PS17] reported that a person who finds a problem and fixes it, or comes with the issue and provides a patch to solve it, is more likely to join the project. *"It shows the developer favors hand-on solutions to technical problem, and that demonstration of technical knowledge in the form of software code submission matters more than signaling of interest and experience"* [PS17].

Ducheneaut [PS7] reported an ethnographic study providing evidence that expertise should be demonstrated by proposing code changes, proving the extent of the technical experience: *"... one also has to create material artifacts."* Ducheneaut reinforced it: *"one can submit bug reports and, simultaneously (this is important), a proposed solution to fix these bugs."*

Bird [PS2] investigated the issue of technical skill by measuring the impact of sending patches and its acceptance by the project when starting the contribution, and he found that *"community perception of a participant's technical skills and knowledge has an effect on becoming a developer."* This result is based upon a previous study, where Bird et al. [PS3] ran a quanti-

tative study based on data mining, the results of which were evaluated using a statistical hazard-rate model with similar findings.

Therefore, newcomers that want to contribute must check if the technical skills required for a given task or project match with their skills. If newcomers are not able to show their skills, they will not be able to contribute immediately. Moreover, a newcomer often does not have enough technical expertise and must develop it within the project: *"For every individual there is a 'race' going on: will s/he become skilled and reputable enough to become a developer before s/he loses interest?"* [PS3]. Actually, there is a blend of technical expertise and social interaction, where the interactions are driven by artifacts that reflect the technical expertise: *"Therefore we encompass capacity and willingness into a single dimension"* [PS20]. It is the result of these demonstrations that will allow both newcomers and developers to perceive the level and possibly lack of technical expertise that hinders effective contributions to the project. The results obtained by the quantitative [PS3, PS13], qualitative [PS7], and mixed method studies [PS20] agree to this extent, reinforcing the importance of technical expertise represented as patches and its demonstration to the developers.

Lack of knowledge of project practices. We found just one study presenting evidence of learning project practices as a problem that influences newcomers not to contribute. The study conducted by Schilling et al. [PS13] found that previous knowledge regarding the project practices influences newcomers' first steps. They reported that *"familiarity with the coordination practices of the project team has a strong association with the time they spend on their projects after GSoC [Google Summer of Code]."*

The authors used historical data from the project to analyze the previous interactions of the subjects before they participated in GSoC to draw their conclusions. Thus, the evidence regarding previous knowledge of project practices refers to previous interactions in the project. It would be interesting to conduct an in-depth investigation on how previous interactions are related to knowledge of project practices and also on how previous knowledge of practices of other projects impacts newcomers.

4.3. Finding a way to start

This category represents the problems related to difficulties that newcomers face when trying to find the right way to start contributing.

Difficulty to find an appropriate task to start with. Find the appropriate task to work on was classified as a problem because new developers have difficulty to find the bugs that are of interest, that match their skill sets, that are not duplicates, and that are important for their future community [43]. Von Krogh et al. [PS17] reported the impact of the issue of finding the right task to work on. They found that *"in 56.7% of the cases members of the community encouraged the new participants to find some part of the software architecture to work on that would match with their specialized knowledge. In only 16.7% of the cases new participants were both encouraged to join and given specific technical tasks to work on."* This occurs because, according to their interviews, the community expects new participants to find their own task to work on instead of receiving a specific piece of work.

Park and Jensen [PS11] reported that *"... subjects expressed a need for information specific to newcomers, for instance, ... what to contribute to (e.g. open issues, required features, sample tasks to start with)."* We can see

that the community wants the newcomers to pick the task themselves [PS17]; however, newcomers have no clue of how to do this [PS11].

Finding the appropriate task is also presented as a barrier by Ben et al. [PS1] and Capiluppi and Michlmayr [PS5]. They studied the impact that the choice of the module in which a newcomer worked had on future participation. Ben et al. [PS1] found that *"developer contribution is influenced by the instability of the code he or she starts to contribute. The code with more developers involve in will lead to less contribution in some degree."* After looking at the source code history, Capiluppi and Michlmayr [PS5] reported that *"... new developers, when joining the project, tend to work more easily on new modules than on older ones ... potential new developers should be actively fostered adding new ideas or directions to the project."*

This problem was evidenced by four studies [PS1, PS5, PS11, PS17] found in this systematic review. The point of view of communities is that newcomers should be able to find the most appropriate task themselves, as reported by von Krogh et al. [PS17]. However, other research studies show that the projects should give special attention to this issue [PS1, PS5, PS11]. Three primary studies relied on the analysis of versioning system histories [PS1, PS5] or mailing list archives [PS17], and one study [PS11] contained feedback from newcomers that were asked to explore the default environment of OSS projects and report what they would have wanted to know about the project before deciding to join. None of them presented evidence collected from project members or newcomers that actually faced the issue. Conducting such a study would uncover the reasons and enable practitioners to offer better support to newcomers. There are some initiatives to support newcom-

ers on this specific issue. One of them is OpenHatch[6], which tries to identify easy issues in several OSS projects and classify them according to language skills; another project is Mozilla, which also provides an easy way to identify simple bugs for newcomers [7].

Difficulty to find a mentor. In industrial settings, it is a common practice to offer mentorship to newcomers to support their first steps [1]. However, in OSS projects that rely on volunteers, it is not a common approach to offer formal mentorship programs. The difficulty to find a mentor is evidenced in one primary study [PS4]. Canfora et al. [PS4] proposed a tool called YODA that aimed to recommend mentors to newcomers. They evaluated the tool by surveying some project members identified as mentors and mentees in the mailing list. They found that *"developers indicated that mentoring is important, although it seems that developers are more likely to admit that they performed mentoring than they were mentored."*

Mentorship is presented as a good way to help newcomers [PS4]. However, it is not clear whether this type of policy can be applied in OSS communities, as it depends on experienced developers to perform this task. Although the entire motivation of the work conducted by Canfora et al. [PS4] is the difficulty to find a mentor, there was no previous empirical evidence of this fact. The evidence was part of the results of the tool evaluation, which was conducted to confirm the mentor recommendations when they asked potential mentors and mentees about their perceived importance of mentoring. Two

[6] `http://www.openhatch.org`

[7] `https://bugzilla.mozilla.org/buglist.cgi?quicksearch=sw:[good+first+bug]`

other primary studies made use of anecdotes to address mentorship; however, they did not present any evidence regarding it [PS6, PS14].

Some issues related to this barrier are still open and deserve to be studied further. For example, the following questions need to be answered: is it really difficult to find a mentor? Can a proper mentorship improve the odds of a newcomer becoming a long-term contributor? How can formal mentorship be adopted in OSS projects?

4.4. Documentation

Documentation is important to newcomers because, in OSS projects, they are expected to learn about the technical and social aspects of the project on their own [34]. The barriers in this category report on which documentation problems have been evidenced as possible barriers to newcomers in OSS projects.

Outdated documentation. Providing outdated documentation can become a barrier to newcomers instead of helping them. Steinmacher et al. [PS14] reported some issues that this barrier can cause: *"we can see many demotivating facts that occurred . . . outdated information in the issue tracker made the developers waste time on an already existent feature and on checking each issue they pick to address. . . "* Stol et al. [PS16] also reported problems regarding outdated documentation. They reported that the subjects *". . . were uncertain whether the available diagrams were still up to date and relevant for the current version of the software. . . Another reported challenge was the uncertainty whether the available documentation was up to date for the current version of the software."*

Even finding two studies providing evidence of this barrier, we found

no study that focused specifically on verifying the influence of this variable on the success of a newcomer. In addition, both studies that presented qualitative evidence gathered feedback from students after they worked with OSS projects. It is necessary to conduct more focused studies to gather more evidence of this barrier and to better understand what the consequences of this barrier are.

Too much documentation. In many OSS projects, newcomers need to explore existing information in mailing lists, source code repositories, issue trackers, and project pages. Rich and up-to-date documentation is essential for newcomers trying to understand the project. However, overzealous attempts to address this can lead to the opposite problem of overwhelming documentation and sources of information, resulting in information overload. Two primary studies [PS11, PS6] identified this overload as a barrier to newcomers. In both of them, the authors conducted experiments to assess the benefits that tools can bring to newcomers. Cubranic et al. [PS6] presented and assessed Hipikat, a tool that supports newcomers by building a group memory linking information from different sources. They conducted a study with newcomers (with previous experience in software development) to evaluate the tool. They reported that *"since the search space is so large, newcomers tend to have difficulty coming up not only with a good conjecture, but also the way of searching through the documentation and code to verify it . . . "* [PS6].

The projects need to provide easy ways to find and navigate the information provided by the projects, linking different sources of information and enabling the recommendation of *"relevant parts of the group memory given*

information about a task a newcomer is trying to perform" [PS6]. The two primary studies that presented evidence of information overload were motivated by an anecdote that there is a need to support newcomers in understanding all of the information provided by OSS projects. They conducted well-planned studies and showed that it is possible to make it easier for newcomers to understand projects and navigate through different sources of information, reducing this barrier.

Unclear code comments. When newcomers want to contribute to OSS projects with source code, they need to understand the code. One way to document the code is by using code comments. By providing unclear code comments, the newcomers will possibly spend more time trying to understand the artifacts they need to work with. This is evidenced in the work by Stol et al. [PS16]. One of the students who participated in their study reported that *"the code was not very well documented, which made it more difficult to understand the source code."*

Even providing just a single piece of evidence, in a superficial way, based on feedback from one student who was working with a specific OSS project, Stol et al. [PS16] revealed this barrier. To better understand the barrier and provide more conclusive evidence, it is necessary to study this phenomenon in-depth to answer various questions, for example, what is an unclear comment? What is the impact of these comments? How is it related to the person's characteristics, motivation, and background?

4.5. Technical Hurdles

This category consists of the technical barriers imposed by the project when newcomers are dealing with the code. To contribute, a newcomer usu-

ally needs to change existing source code. Therefore, it is necessary for newcomers to have enough knowledge about the code to begin their contributions. This category includes barriers related to issues to set up a local workspace, and to understand and modify the source code.

Issues setting up a local workspace. The feedback obtained by Stol et al. [PS16] evidenced that newcomers have difficulties in setting up their environment. He reported some obstacles, for example, *"a challenge was that some [subjects] did not have any experience or knowledge on checking out source code from the version control system."* They also reported problems in building the project, which is a step of the workspace setup: *"some [subjects] encountered build errors, which caused them to manually fix the build configuration."* Another complaint was that *"the compilation process 'was a nightmare'; this was caused by the many dependencies on other subsystems that had to be downloaded in addition."*

Documentation can certainly help newcomers on setting up a local workspace. In fact, some communities provide tutorials and step-by-step cookbooks on how to obtain the code, setup up and build a local workspace. However, some projects have peculiar workspace requirements that cannot be ameliorated just by documentation. For instance, management of required building libraries can be addressed by employing building systems, with dependency handling mechanisms. Although their usage requires newcomers' technical expertise, the provision of a building system is responsibility of the project.

It is worth noticing this barrier was evidenced just by the study conducted by Stol et al. [PS16]. Conducting observational studies and interviews with newcomers would help identifying more specificities of this barrier.

Code complexity. The study conducted by Midha et al. [PS10] analyzed how code complexity influenced newcomers contribution to OSS projects. Using statistical tools to analyze many characteristics of 450 SourceForge projects, Midha et al. [PS10] showed that *"cognitive [code] complexity has a strong negative influence on the number of contributions from new developers. As OSS thrives upon voluntary contributions, the project managers must actively control the source code complexity in order to attract contributions from new developers. In a complex piece of code, it takes longer for a developer to determine the flow of logic resulting in slower progress of the project."*

This is another case of a barrier supported by evidence from only one primary study. In this case, the evidence was obtained from a case study counting on a large sample, aiming to verify whether source code's cognitive complexity was negatively associated with the number of contributions to the OSS source code from new developers.

Software architecture complexity. Besides code complexity (unit level), as in the previous barrier, it is important to consider the software architecture (integration and system level). Stol et al. [PS16] highlighted some complaints of newcomers about the project structure of OSS projects. One subject reported that *"the hierarchy of the source code directory was counter intuitive for someone with little architecting experience."* Other subjects *"also reported that manually browsing the source code for patterns was 'tricky' and very time consuming."* Cubranic et al. [PS6] also presented an issue faced during their experiment: *"We also had reports of a pair missing a relevant suggestion because they lacked knowledge about the overall structure of the*

system. . . "

Park and Jensen [PS11] analyzed *"the potential benefits of information visualization in supporting newcomers through a controlled experiment."* They found that *"the impact of information visualization. . . suggests that providing visual information to newcomers may reduce the challenges they face when learning about a new project."* The feedback obtained from the subjects revealed that *"providing visual information such as the class diagrams or dependency views . . . would help new developers understand the structure of existing code and find problems to work on."*

The main complaint regarding code is that its structure is hard to understand and that learning it takes too much time. The difficulty in dealing with this type of barrier is related to the newcomer's previous knowledge. Approaches such as visualization [PS11] and artifact recommendation [PS6] can support newcomers in overcoming this barrier, but further solutions should be investigated.

4.6. Summary

Considering the model defined in Fig. 7, based upon barriers identified by using GT procedures throughout the selected studies, we can summarize the evidence for each problem as shown in Table 7. In Fig. 8, we also present the barriers ranked according to the number of studies that evidence them. The category most thoroughly studied is social interactions, accounting for 12 studies, followed by Newcomers' Previous Knowledge with 8 studies. The other three categories range from 4 to 6 related studies each. It is possible to note that, thus far, the literature is focused on the social issues. The more technical barriers, such as software architecture complexity, dealing with the

versioning system, setting up a local workspace, and understanding the code, are poorly studied.

Table 7: Studies that evidence each barrier.

Category	Barrier	Studies
Social Interaction [12 studies]	Lack of social interaction with project members [7 studies]	[PS2], [PS3], [PS7], [PS8], [PS12], [PS19], [PS20]
	Not receiving a (timely) answer [6 studies]	[PS9], [PS14], [PS15], [PS16], [PS17], [PS20]
	Receiving an improper answer [3 studies]	[PS9], [PS15], [PS16]
Newcomers' Previous Knowledge [8 studies]	Lack of technical experience [6 studies]	[PS2], [PS3], [PS7], [PS13], [PS17], [PS20]
	Lack of domain expertise [2 studies]	[PS16], [PS17]
	Lack of knowledge of project practices [1 study]	[PS13]
Technical Hurdles [6 studies]	Issues setting up a local workspace [1 study]	[PS16]
	Code complexity [1 study]	[PS10]
	Software architecture complexity [3 studies]	[PS6], [PS11], [PS16]
Finding a Way to Start [5 studies]	Difficulty to find appropriate task to start with [4 studies]	[PS1], [PS5], [PS11],[PS17]
	Difficulty to find a mentor [1 study]	[PS4]
Documentation [4 studies]	Too much documentation [3 studies]	[PS6], [PS11], [PS16]
	Outdated documentation [2 studies]	[PS14], [PS16]
	Unclear code comments [1 study]	[PS16]
	Lack of documentation [1 study]	[PS16]

*** Some studies support more than one barrier

Considering studied barriers, we found that the most evidenced were (1) lack of technical experience, from the Newcomers' Previous Knowledge category, and (2) lack of social interaction with project members, from the Social Interaction category. Overall, barriers related to technical issues, which include code issues and difficulty to find a way to start, were individually backed by less evidence.

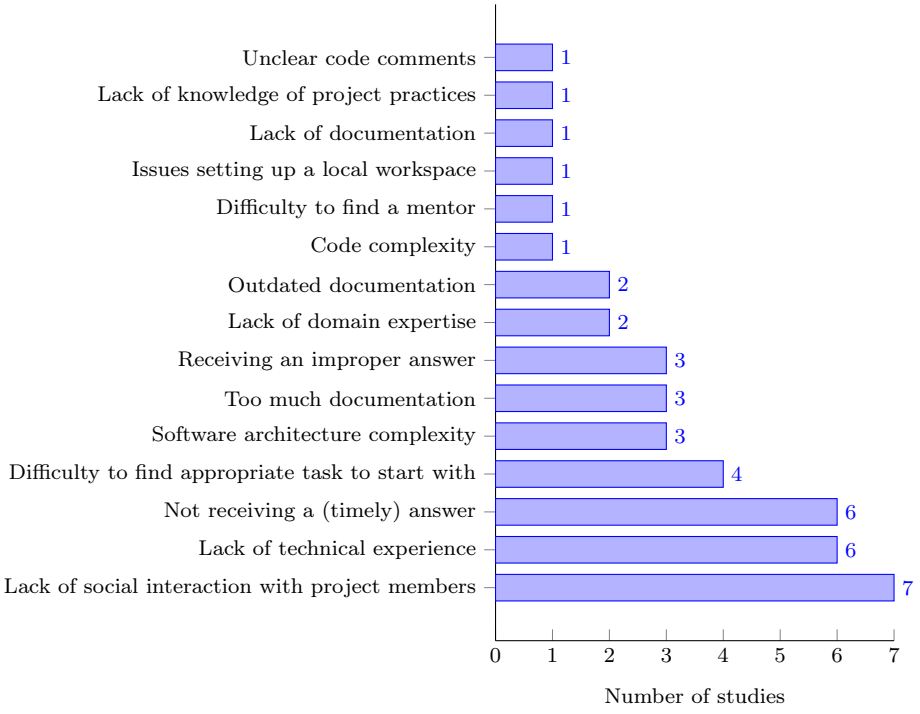


Figure 8: Barriers ranked by the number of studies that evidence each.

5. Origin of the barriers and Some Initial Guidelines

In addition to the identification and classification of the barriers based on their type, we organized the barriers according to their origin, discussing them along with a initial set of guidelines to help newcomers to overcome the barriers.

Three different "origins" were identified during our analysis: newcomers, community, and the product. Some of the barriers are related to more than one origin because different actions can be taken to change the barriers' impact. For example, *"setting up local workspace"* is a barrier that can be originated by the newcomer's profile (*e.g.*, inexperience, or issues regarding installation of development tools), but the community can also contribute to the barrier if it does not provide some level of support in preparing the workspace.

In the following subsections we discuss the barriers along with some guidelines for supporting newcomers, enabling practitioners and researchers to take action and propose tools and processes targeting the origin of the barriers.

5.1. Newcomers

We classified newcomers as the origin of the barriers caused by their behavior or profile. By behavior, we mean the manner in which newcomers interact with the community, while by profile, we mean newcomers' knowledge background. These barriers are mainly under the *"Newcomers' Previous Knowledge"* category and are related to newcomers' profiles. In this section, we present the barriers and EFOREvarious hints for helping newcomers guide themselves to overcome these barriers.

We found that in OSS projects, newcomers themselves play an important role during the contribution process. This means that before contributing to a project, newcomers must, for example, verify whether their skills match with the skills needed to work in the project or choose a project of a known business domain. It is necessary to find a project that fits a newcomers' profile to enable their contribution. This type of problem are usually not addressed by the community directly. Newcomers themselves need to take action and search for the proper knowledge that is appropriate when joining the project.

Newcomers' behavior is also a problem that influences future contributions. Newcomers that showed proactivity, sending patches and participating in discussions, were better received by the community. Some of the primary studies reported that the content of a message influences the reception of a newcomer. Being social and acquiring political supporters is also an important problem when a newcomer wants to become a core developer. However, it is important to note that this social and political behavior was important for newcomers to become long-term contributors or to be accepted by the communities as members.

By checking the literature, we observed that good social skills, allied with the right technical skills and proactivity, can positively influence the contributions. T

5.2. Community

Community was considered the origin of the barriers influenced by the project processes and the behavior of the community members when interacting directly with the newcomers. Improvements in community receptivity

and more appropriate collaborative environments for OSS development can result in better support for newcomers, positively influencing successful contributions.

We found that lack of information made available by the project and lack of support given by the community are important problems that can hinder newcomers when they want to join a project. For example, Park and Jensen [PS11] reported that the SourceForge repository could be made more interactive and user-friendly, making it look less geeky/daunting for newcomers. Community members can support newcomers by making it clear what is expected from them and what the process and practices are that must be followed to contribute. Thus, a more informative and less technical initial environment is beneficial for newcomers.

Community was classified as the origin of some social interaction barriers. These problems are related to the way the community interacts with newcomers. OSS communities need to take special care with the barriers reported by some authors [PS9, PS14, PS15, PS16] concerning community recognition and giving proper answers to newcomers. They evidenced that the absence of responses, improper answers, and not receiving recognition from the community can lead to newcomer dropout.

Large and more organized projects could nominate people with social skills to receive newcomers in the communication channels. Newcomers need to be welcomed, avoiding the use of project specific terms and jargons, and they need to receive proper directions in a positive way when they reach the community. For the community members who have the opportunity to interact with newcomers, social skills are required. Improper reception can result

in losing valuable resources. In addition, it is interesting to make newcomers aware of the average time to receive answers and about the critical periods (such as pre-release) to help manage their expectations. Newcomers need to be warned and oriented instead of being improperly received or dismissed.

5.3. Product

The product was classified as the origin of problems related to source code and documentation. Regarding source code, its structure and complexity [PS6, PS10, PS11, PS16] were the barriers reported by the literature, and the product was assigned as their origin. Barriers related to documentation were also identified in some primary studies [PS6, PS11, PS14, PS16]. Even knowing that these artifacts (source code and documentation) are the result of community interaction, we classified these barriers as originated by the product because our goal is to point to where practitioners and research community need to pay attention.

The main product of the OSS projects is the code. Code complexity was identified as a barrier in the study conducted by Midha et al. [PS10]. They analyzed a large set of projects and verified that code complexity negatively influenced newcomers' decision to contribute to the projects. To help newcomers in reducing this technical barrier, Cubranic et al. [PS6] and Park and Jensen [PS11] conducted experiments to analyze the potential of tools in supporting newcomers in understanding the code structure. The former [PS6] proposed and evaluated a tool (Hipikat) that recommends the source code artifacts that should be related to the issue a newcomer is working on. The latter [PS11] evaluated the benefits of source code visualization tools in supporting newcomers. Both of them reported good results when newcomers

are supported by the tools.

Documentation is also a product generated by the community. Stol et al. [PS16] reported some comments from students that worked with OSS projects. They presented feedback that raises issues regarding outdated information, lack of documentation and diagrams, too much documentation, and lack of meaning in the code comments. Steinmacher et al. [PS14] also presented feedback from newcomers and reported that *"newcomers became demotivated due to outdated information."* Information overload is also reported and explored in two experiments conducted with tools aiming to facilitate newcomers' joining process [PS6, PS11].

Keeping the code simple and the documentation organized and up-to-date potentially increases the odds of receiving contributions from newcomers. Community members need to think of their product from the perspective of newcomers if they want to receive more contributions. The source code is the set of artifacts that need to be understood and changed by the members to contribute. Perhaps keeping the code simple and easy to understand is a good way to increase the number of newcomers contributing. Sometimes it is not easy to make it simple because the core of a large application is inherently complex, and the code reflects this complexity. However, directing newcomers to peripheral modules (at least warning them of the complexity) or offering tools to help them find the right artifacts to work on given a specific task would benefit newcomers during their first steps.

Regarding documentation, providing easy-to-access, organized and up-to-date documentation would benefit newcomers. In the case where there is no way to keep it up-to-date, it would be a nice policy to make newcomers

aware of the status of the documents (which ones are outdated, which are still up-to-date). In addition, the use of tools such as Hipikat that aim to link information from different sources to build a project memory would be beneficial to provide newcomers with a way to navigate the information.

6. Discussion

For purposes of simplicity, in Section 4, we presented some of the discussion of our results alongside each of the barriers, and in Section 5, provided some discussion about the origins of the barriers and some countermeasures in the form of guidelines. In this section, we discuss some of the key findings at a higher-level perspective.

In general, this study identified empirical evidence of barriers faced by newcomers to OSS projects. This empirical evidence is important, as many studies are motivated by or deal with anecdotal evidence. This paper brings evidence from reality, which is rarely precisely documented.

As a result of this study, we found that the most evidenced barriers are related to socialization, appearing in 75% (15 out of 20) of studies analyzed, with high focus on interaction in mailing lists (receiving response and socialization with other members). We also noticed a lack of in-depth studies on technical issues faced by newcomers. The reason can be attributed to the small number of qualitative studies found because it cannot be quantitatively extracted from mailing lists. For example, technical hurdles are evidenced by only five studies analyzed. Issues related to workspace setup is reported in only one study, by one subject in a debrief session. These kinds of issue deserve more attention, from both practitioners and researchers.

Some barriers identified in this systematic review are also reported and analyzed by the literature of other online communities; the most explored barriers are those related to community reception issues. For example, not receiving an answer is well evidenced quantitatively in Q&A literature [24, 28, 44]. Receiving impolite answers was also largely studied in the CSCW literature,

mainly on the analysis of reverts in Wikipedia [20, 14, 41]. The proposed strategies of automated answers and feedback used in Wikipedia [15, 19] can be adapted and then evaluated in OSS context.

Studies that used mixed methods to draw their conclusions are good examples of how to bring evidence from the historical data and contextualize them. It is the case of the studies conducted by von Krogh et al. [PS17] and Ducheneaut [PS7]. The former used interviews with project members and an analysis of issue tracker, mailing list, and project documents to bring qualitative evidence presented with and backed by quantitative information mined from a project's repository.

On the other hand, studies such as those conducted by Steinmacher et al. [PS14] and Jensen et al. [PS9] presented simplistic views of the problem when they drew conclusions from only analyzing the first messages from newcomers and their retention. The context is important: Why did they send the messages? What motivated them? Did they really want to contribute or just clarify some doubt? Did they contribute at the end but never got back to the mailing list? To answer such questions, we need to merge information from different sources (issue tracker, mailing lists, documentation, code repository) and verify the context by talking to practitioners. Another possibility is to conduct observational and ethnographic studies by analyzing the barriers and effects for newcomers in real settings.

It is worth noting that the main focus of analysis were large projects with a high number of developers and more than five years of existence. Moreover, projects that focused on products used during the development cycle and developed in Java and C were preferred. Such projects can be

classified as clearly successful projects which, combined with the historical data available, provide an easy target to search for newcomers. We observed that, although projects gain several newcomers, just a small percentage are successful in contributing some source code. Because the identification of barriers faced and surpassed by such newcomers is important, projects with a high number of developers (and newcomers) are easier to analyze to find evidence of such barriers. However, a high number of OSS projects present different characteristics, such as small teams and short lifetime, and were not considered for evaluation. Naturally, such projects provide less data and are less attractive than large successful projects, but when considering newcomers, they can account for different problems than those identified by our model or modify their importance. Further investigation is required regarding such projects to improve the model of barriers described in this paper.

We analyzed the characteristics and goals of the newcomers. However, many of the papers did not explicitly profile the newcomers they analyzed. This is probably related to the type of data analyzed and the type of study conducted, as most of the studies only used data coming from software repositories and did not go deeper in the analysis of the subjects. The problem is that the term newcomer can be used in a loose way, which can bias the results. Newcomers can be novice developers who are starting their career, people who are experienced developers from industry but are not used to OSS projects, or people who are migrating from other OSS projects. These three profiles are different and can face different barriers or experience barriers differently. Therefore, it would be a better approach to assess how these

different types of developers see the barriers and what their impressions of them are. For example, does a novice developer find more issues to contribute than an experienced developer without an OSS background?

In one case [PS13], the newcomers analyzed were participants of the Google Summer of Code. They were students that received a scholarship from Google to develop for KDE. Experienced KDE developers were assigned as formal mentors of the participants. There are other cases where students were the subjects of studies. Steinmacher et al. [PS15] received some feedback from Ph.D. students. The subjects were technically skilled and were newcomers to Open Source Software projects. Park and Jensen [PS11] conducted their experiment on visualization tools with students who had at least two years of experience with Java programming.

The participants of a Cubranic et al. [PS6] study were experienced in developing large or medium-sized software systems and familiar with tools commonly used (versioning systems, issue trackers, etc.). Ducheneaut [PS7] analyzed the steps followed by a Python developer from his first interactions with the project until he became a core developer. In this case, the subject was experienced and was motivated to become a core developer from the beginning.

Other studies did not focus on any specific newcomer profile. They analyzed first posts on mailing lists, history of socialization in mailing lists, participation in issue trackers, patches sent, and commits submitted. Therefore, their focus seems to be on developers that have some technology skills. As reported by Cubranic et al. [PS6], they were "interested in studying newcomers, not novices."

The goals of the studies also differ. The goal of some [PS3, PS7, PS18] was to provide guidelines or scripts on "how to become a core member." Other studies [PS19, PS20] model the chances of a person becoming a Long-Term Contributor based on his willingness and the project environment. Schilling et al. [PS13] analyzed the characteristics of subjects who started as participants of Google Summer of Code that remained in the project after the end of the program. Other studies do not mention the goals of newcomers in terms of their future in the project.

This heterogeneity of profiles, goals, and projects analyzed provides opportunities for future research, such as analyzing how these barriers are felt by different profiles and in different types of projects.

OSS researchers can also benefit from these results by using them to conceive strategies for newcomer support. To achieve this, it is necessary to put more effort on specific research topics, such as understanding and creating ways to measure the influence of the barriers in newcomers' experience, identifying and creating different strategies to lower each barrier, and proposing metrics to grade the support offered for each barrier. To gain a better understanding of the barriers and to what extent they need to be lowered, it is important to conduct more qualitative studies because this phenomenon occurs in a complex, social environment in which the context of its occurrence is important. Moreover, a qualitative view complements the existing literature, which relies mostly on quantitative evidence.

OSS practitioners can take advantage of the barriers model to organize the project environment and to adjust their behavior to better receive newcomers. The model can be used as a guide on what type of information or tools need

to be provided to newcomers. By providing ways to lower entry barriers, the projects can benefit from more contributions and, possibly, from more long-term contributors. Assuming that, as stated by Dagenais et al. [PS7], "newcomers are explorers who must orient themselves within an unfamiliar landscape," the model of barriers can be used by OSS communities to place the proper signs and maps to help newcomers orient themselves and to alert or warn them about the obstacles that they might face.

Although we considered the barriers as something that can hinder newcomers' contributions, some barriers can be used as filters by the projects. Findings from a Halfaker et al. [19] study on Wikipedia newcomers revealed that some entry barriers led to improved contributions in the future. Moreover, research conducted in the OSS domain [33, 13] demonstrated that socialization barriers are useful for maintaining community integration and the quality of the community's product. A clear direction for future work is to explore how the communities perceive these barriers and how they impact the quality of contributions from newcomers.

7. Threats to Validity

In this section, we identified and described the threats to the validity of our study. As for the construct validity, our main concern was to select every relevant study, despite issues regarding the sources of study, nomenclature, and fairness in the selection process. Regarding internal validity, the quality of studies we have found are an important threat. Most papers do not consider barriers as a main focus. Finally, the external validity is threatened by the sample of OSS projects considered, as well as the profiles and motivations of the newcomers considered by studies. These items pose some difficulties when attempting to extrapolate current results to other OSS projects.

7.1. Construct validity

Several tactics were employed to assure the construct validity of this systematic review. The protocol considered several sources for searching, and we validated them with five experts (chairs of previous conferences on OSS). In addition to venues that we had already included, they suggested libraries that, unfortunately, did not provide any search mechanism. Although that may be considered a threat, it is reasonable to consider that relevant research, available in local libraries, will eventually be published in indexed venues. Moreover, author and citation analysis can help to retrieve such papers.

We adjusted the search expression to cover all relevant papers that were of our knowledge and conducted pilot studies using a group of relevant studies. The same experts we consulted for sources validated this set of verified studies. Every step of the selection process was conducted by two researchers whose decisions always needed to agree or be made in consensus.

Despite all of those measures, this review may have missed some papers that address problems faced by newcomers to OSS projects because we did not perform our search into every possible source and some relevant papers may not contain the chosen terms. Moreover, some papers that address barriers but were not positioned as studies about newcomers to OSS trying to make code contributions were not included in this review. This can explain the lack of studies that investigate technical barriers. The adoption of authoring and citation analysis, in addition to searching in digital libraries, contributed to helping mitigate this issue. For instance, as reported in Table 2, two papers were found by author snowballing.

Another threat to our study was that it considered only newcomers as our object of interest. The selection for barriers was accomplished by the selection criteria of the systematic review. This was due to known limitation on the literature available on the topic, with few studies explicitly regarding barriers to newcomers on OSS projects.

The findings of this review may have also been affected because classification is a human process and it is based on some subjective criteria. In particular, the terms of the area do not have a common definition among all studies. The problems were classified based on the procedures of coding from Grounded Theory, which also relies on manual classification. The use of such an approach is not common in systematic reviews. However, we employed it because our goal was to identify and categorize the barriers faced by newcomers and identifying and categorizing concepts is one of the key components of grounded theory. To reduce the bias related to these concerns, this review involved two researchers cross checking each paper for inclusion

and the coding process, and a third researcher responsible for reviewing and discussing the information generated after each step.

Moreover, since our goal was to identify and classify the barriers evidenced in the literature, we reported each of the barriers as independent factors that influence newcomers. Therefore, there could be some relationships among the barriers, since they were not reported by the primary studies and require further investigation in other sources.

7.2. Internal validity

Most of the studies analyzed did not present as their main focus the analysis of the newcomers' needs or the problems they faced during their first steps. The papers that aim to analyze barriers faced by newcomers focus on very specific problems. We know that it would be hard – or even impossible – to identify every factor that can affect newcomers. The quality of the evidence provided by the selected studies is also a threat to the internal validity. There are a small number of studies reporting experiments that address the barriers. One approach employed by several studies was to identify successful newcomers and trace the actions they did differently from other newcomers that did not contribute to the project. However, although that can show a correlation between them, it may not necessarily be a barrier. Unfortunately, we could not find a study that quantitatively analyzed just this factor, although some qualitative studies did provide some evidence of barriers found by unsuccessful newcomers. With this paper, we expect to foster more papers addressing barriers and newcomer contribution, providing empirical primary studies that can be used to improve our results with newer iterations of this systematic review.

To this end, one feature we expect from future primary studies on newcomers contributions is a better description and control of variables. Arguably, this is not trivial, as there are several variables at hand. However, it would be interesting to consider *"forks"* of projects, such as OpenOffice and LibreOffice, and how the entry barriers they have chosen to address contributed to newcomers' success. However, this approach would face the issue that, in such projects, several interventions are applied at once, making it difficult to identify the variable that really caused the effect. Another approach would be creating controlled experiments using real OSS projects. For example, it could provide a new tool, developed to address one barrier, and evaluate its effectiveness. This might not be suitable for large projects, but could be provided for smaller projects in software forges such as SourceForge and GitHub.

Once more studies become available, our systematic review protocol could be updated to consider barriers or tools used to mitigate a specific barrier and their effects, such as patch submissions and code contributions. For instance, several studies could extract similar information from the same source and provide evidence towards specific barriers (such as lack of technical experience and lack of social interaction with project members), which could be aggregated to produce stronger evidence.

7.3. External validity

The studies we selected in our systematic review considered just a small subset of available OSS projects. Although it would be infeasible to select a proper sample from the population, most papers considered few successful and mature software programs. We could find just four projects that used a

more heterogeneous collection [PS8, PS10, PS12, PS19], such as those hosted at SourceForge and Apache Foundation.

The choice of projects considered by most researchers is understandable. After all, data mining approaches require a considerable volume of data, which is often provided by mature and large projects. However, the barriers identified in them are not necessarily the same as those of smaller projects. Nonetheless, considering studies with small code bases [PS16, PS17], the results were compatible with those from studies of larger projects. This does not mean that the barriers are always the same, but at least it provides some hint that some barriers are independent of a project's characteristics. Further studies must be conducted to evaluate whether there are distinct barriers between such projects and assess the importance among them.

As discussed in the previous section, most of the papers did not explicitly profile the newcomers analyzed, and there is a high diversity of projects studied. Thus, it is difficult to generalize or to specify the implications of the results of this systematic review.

Another threat to external validation is that newcomers can be driven by different motivations to contribute and present different expectations from the project. In our model, we use the term *'newcomers to OSS'* to refer to volunteer contributors. However, OSS is increasingly being driven by paid employees in commercial companies. In such cases, some of the barriers can be softened or mitigated by the motivation that drives paid newcomers and by the socialization tactics used by companies.

8. Conclusion

In this study, we identified 20 papers that evidence barriers faced by newcomers while making a contribution to an OSS project. We aggregated the barriers evidenced across the related literature in a single place. By using a coding approach inspired by Grounded Theory procedures to organize the barriers, we proposed a model composed of five categories: social interactions, newcomers' previous knowledge, finding a way to start, documentation, and technical hurdles. The model extracted from the literature (presented in Fig. 7 and discussed in Section 4) is the main contribution of this systematic review, as it brought to light the barriers that were already evidenced in the literature as barriers for newcomers to contribute to OSS projects. This classification provides a baseline for further research related to contribution barriers faced by newcomers to OSS projects. We also classified the problems regarding their origin: newcomers, community, or product. Such classification can be used to provide a quick reference for researchers and OSS practitioners willing to investigate or implement tools and mechanisms to support newcomers.

Considering the most studied barriers, we found that the most evidenced are (1) lack of social interaction with project members; (2) not receiving a (timely) answer, both from Social Interaction category; and (3) newcomers' previous technical experience, from Newcomers' Previous Knowledge category. It is important to notice that although the social interaction issues were the most evidenced barriers, we found a lack of evidence of the causal relationship between these barriers and newcomer success. We also highlight that, overall, barriers related to technical issues, which include code issues

and difficulty to find a way to start, are individually backed by less evidence.

We noticed a high diversity of projects studied. Most of the authors looked for mature, well-established projects to conduct their studies. In addition, 66.67% of the projects studied were products used during the development cycle (Application Servers, Frameworks, IDE, etc.). The higher focus on established, large projects that deliver products used during the development cycle can introduce a bias in the results of the studies analyzed.

Most of the studies analyzed rely on the results of quantitative case studies using historical data gathered from software repositories. Historical data can highlight the real obstacles and problems faced by newcomers, but conducting experiments with newcomers, like [32, 11], and gathering information from the project members and newcomers by means of interviews and surveys can reveal the real problems and needs of the newcomers.

Based on the analysis conducted, we can conclude that newcomers that wish to contribute must have a blend of domain knowledge, technical skills, and social interaction, which can increase the odds of a successful joining. The interactions are driven by artifacts that reflect the technical and domain expertise. It is the result of these interactions that will allow both newcomers and developers to perceive the level and possibly lack of background that hinders effective contributions to the project.

It is also important to highlight that improvements in community receptivity and more appropriate collaborative environments for OSS development can result in better support for newcomers. Improper reception can result in losing valuable resources. Moreover, community members need to think of their product in terms of newcomers in case they want to receive more

contributions. Keeping the code simple and the documentation organized and up-to-date could potentially increase the odds of receiving contributions from newcomers.

Identifying the barriers evidenced by the literature (Section 4) and providing some guidelines (Section 5) for community members and newcomers were the initial steps towards better orienting newcomers' first steps. OSS projects can benefit from additional contributions if they offer the right support specific for newcomers who are trying to contribute to the project. A smooth first contribution may increase the total number of successful contributions made by single contributors and, hopefully, the number of long-term contributors.

In the future, we aim to conduct some qualitative studies to confirm the problems evidenced by the literature. We are conducting some interviews with experienced OSS developers and newcomers to verify the main problems faced by newcomers from their perspective. We also plan to refine the classification model based on the results of the interview analysis. Additionally, based on the model, it is possible to propose awareness mechanisms and tools to offer better support for newcomers.

Acknowledgements

The authors would like to thank UTFPR, Fundação Araucária, CNPq (proc. 477831/2013-3), NAPSoL-PRP-USP, NAWEB, and FAPESP for their financial support. Igor Steinmacher received grants from CAPES (BEX 2038-13-7).

References

[1] Begel, A., Simon, B., 2008. Novice software developers, all over again. In: Proceedings of the Fourth International Workshop on Computing Education Research. ICER '08. ACM, New York, NY, USA, pp. 3–14.

[2] Ben, X., Beijun, S., Weicheng, Y., 2013. Mining developer contribution in open source software using visualization techniques. In: Proceedings of the 2013 Third International Conference on Intelligent System Design and Engineering Applications. ISDEA '13. IEEE, Hong Kong, pp. 934–937.

[3] Biolchini, J., Mian, P. G., Natali, A. C. C., Travassos, G. H., May 2005. Systematic review in software engineering. Technical Report RT-ES 679/05, COPPE/UFRJ, Rio de Janeiro, RJ, Brazil.
URL `http://www.cin.ufpe.br/~in1037/leitura/systematicReviewSE-COPPE.pdf`

[4] Bird, C., 2011. Sociotechnical coordination and collaboration in open source software. In: Proceedings of the 2011 27th IEEE International Conference on Software Maintenance. ICSM '11. IEEE, Washington, DC, USA, pp. 568–573.

[5] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A., Hsu, G., 2007. Open borders? immigration in open source projects. In: Proceedings of the Fourth International Workshop on Mining Software Repositories. MSR '07. IEEE, Washington, DC, USA, pp. 1–8.

[6] Canfora, G., Di Penta, M., Oliveto, R., Panichella, S., 2012. Who is going to mentor newcomers in open source projects? In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. FSE '12. ACM, New York, NY, USA, pp. 44:1–44:11.
URL http://doi.acm.org/10.1145/2393596.2393647

[7] Capiluppi, A., Michlmayr, M., 2007. From the cathedral to the bazaar: An empirical study of the lifecycle of volunteer community projects. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (Eds.), Open Source Development, Adoption and Innovation. Vol. 234 of IFIP – International Federation for Information Processing. Springer US, Limerick, Ireland, pp. 31–44.
URL http://dx.doi.org/10.1007/978-0-387-72486-7_3

[8] Capra, E., Wasserman, A. I., 2008. A framework for evaluating managerial styles in open source projects. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (Eds.), Open Source Development, Communities and Quality. Vol. 275 of IFIP – The International Federation for Information Processing. Springer US, pp. 1–14.
URL http://dx.doi.org/10.1007/978-0-387-09684-1_1

[9] Corbin, J. M., Strauss, A., 2008. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 3rd Edition. SAGE Publications.

[10] Cubranic, D., Murphy, G. C., 2003. Hipikat: recommending pertinent software development artifacts. In: Proceedings of the 25th International

Conference on Software Engineering. ICSE 2003. IEEE, Washington, DC, USA, pp. 408–418.

[11] Cubranic, D., Murphy, G. C., Singer, J., Booth, K. S., Jun. 2005. Hipikat: a project memory for software development. IEEE Transactions on Software Engineering 31 (6), 446–465.

[12] Dagenais, B., Ossher, H., Bellamy, R. K. E., Robillard, M. P., de Vries, J. P., 2010. Moving into a new software project landscape. In: Proceedings of the ACM/IEEE 32nd International Conference on Software Engineering. Vol. 1. ACM, New York, NY, USA, pp. 275–284.

[13] Ducheneaut, N., Aug 2005. Socialization in an open source software community: A socio-technical analysis. Computer Supported Cooperative Work 14 (4), 323–368.

[14] Farzan, R., Kraut, R. E., 2013. Wikipedia Classroom Experiment: Bidirectional benefits of students' engagement in online production communities. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '13. ACM, New York, NY, USA, pp. 783–792. URL http://doi.acm.org/10.1145/2470654.2470765

[15] Faulkner, R., Walling, S., Pinchuk, M., 2012. Etiquette in wikipedia: Weening new editors into productive ones. In: Proceedings of the Eighth Annual International Symposium on Wikis and Open Collaboration. WikiSym '12. ACM, New York, NY, USA, pp. 5:1–5:4. URL http://doi.acm.org/10.1145/2462932.2462939

[16] Fogel, K., 2013. Producing Open Source Software: How to Run a Successful Free Software Project, 1st Edition. O'Reilly Media.
URL http://www.producingoss.com/

[17] Forte, A., Lampe, C., 2013. Defining, understanding, and supporting open collaboration: Lessons from the literature. American Behavioral Scientist 57 (5), 535–547.
URL http://abs.sagepub.com/content/57/5/535.abstract

[18] Gacek, C., Arief, B., Jan 2004. The many meanings of open source. IEEE Software 21 (1), 34–40.

[19] Halfaker, A., Geiger, R. S., Morgan, J., Riedl, J., 2013. The rise and decline of an open collaboration system: How wikipedia's reaction to sudden popularity is causing its decline. American Behavioral Scientist 57.
URL http://abs.sagepub.com/content/57/5/664

[20] Halfaker, A., Kittur, A., Riedl, J., 2011. Don't bite the newbies: How reverts affect the quantity and quality of Wikipedia work. In: Proceedings of the 7th International Symposium on Wikis and Open Collaboration. WikiSym '11. ACM, New York, NY, USA, pp. 163–172.
URL http://doi.acm.org/10.1145/2038558.2038585

[21] He, P., Li, B., Huang, Y., 2012. Applying centrality measures to the behavior analysis of developers in open source software community. In: 2012 Second International Conference on Cloud and Green Computing. CGC. IEEE, Washington, DC, USA, pp. 418–423.

[22] Jalali, S., Wohlin, C., 2012. Systematic literature studies: Database searches vs. backward snowballing. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '12. ACM, New York, NY, USA, pp. 29–38.
URL http://doi.acm.org/10.1145/2372251.2372257

[23] Jensen, C., King, S., Kuechler, V., 2011. Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In: 44th Hawaii International Conference on System Sciences. HICSS. IEEE, Kauai, HI, USA, pp. 1–10.

[24] Joyce, E., Kraut, R. E., 2006. Predicting continued participation in newsgroups. Journal of Computer-Mediated Communication 11, 2006.

[25] Kitchenham, B., Jul 2004. Procedures for performing systematic reviews. Tech. Rep. TR/SE-0401, Department of Computer Science, Keele University, Keele, UK.

[26] Kitchenham, B., Brereton, P., Dec 2013. A systematic review of systematic review process research in software engineering. Information and Software Technology 55 (12), 2049–2075.

[27] Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Joint Report EBSE 2007-001, Keele University and Durham University.

[28] Lampe, C., Johnston, E., 2005. Follow the (Slash) Dot: Effects of Feedback on New Members in an Online Community. In: Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group

Work. GROUP '05. ACM, New York, NY, USA, pp. 11–20.
URL http://doi.acm.org/10.1145/1099203.1099206

[29] Midha, V., Palvia, P., Singh, R., Kshetri, N., 2010. Improving open source software maintenance. Journal of Computer Information Systems 50 (3), 81–90.

[30] Morgan, J. T., Bouterse, S., Walls, H., Stierch, S., 2013. Tea and sympathy: Crafting positive new user experiences on wikipedia. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work. CSCW '13. ACM, New York, NY, USA, pp. 839–848.
URL http://doi.acm.org/10.1145/2441776.2441871

[31] Nagappan, M., Zimmermann, T., Bird, C., 2013. Diversity in software engineering research. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ESEC/FSE 2013. ACM, New York, NY, USA, pp. 466–476.
URL http://doi.acm.org/10.1145/2491411.2491415

[32] Park, Y., Jensen, C., 2009. Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In: Proceedings of the 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis. IEEE, Washington, DC, USA, pp. 3–10.

[33] Qureshi, I., Fang, Y., 2011. Socialization in open source software projects: A growth mixture modeling approach. Organizational Research Methods 14 (1), 208–238.
URL http://orm.sagepub.com/content/14/1/208.abstract

[34] Scacchi, W., Feb 2002. Understanding the requirements for developing open source software systems. IEE Proceedings Software 149 (1), 24–39. URL http://digital-library.theiet.org/content/journals/10.1049/ip-sen_20020202

[35] Schilling, A., Laumer, S., Weitzel, T., 2012. Who will remain? an evaluation of actual person-job and person-team fit to predict developer retention in FLOSS projects. In: 45th Hawaii International Conference on System Sciences. IEEE, Washington, DC, USA, pp. 3446–3455.

[36] Schmidt, F. L., Hunter, J. E., Sep. 1998. The validity and utility of selection methods in personnel psychology: Practical and theoretical implications of 85 years of research findings. Psychological Bulletin 124 (2), 262–274.

[37] Steinmacher, I., Gerosa, M. A., Redmiles, D., 2014. Attracting, onboarding, and retaining newcomer developers in open source software projects. In: Workshop on Global Software Development in a CSCW Perspective held in conjunction with the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW'14). URL http://143.107.45.80/public/papers/15705/NEXGSD.pdf

[38] Steinmacher, I., Wiese, I., Chaves, A. P., Gerosa, M. A., 2013. Why do newcomers abandon open source software projects? In: 6th International Workshop on Cooperative and Human Aspects of Software Engineering. CHASE'13. IEEE, Washington, DC, USA, pp. 25–32.

[39] Steinmacher, I., Wiese, I. S., Gerosa, M. A., 2012. Recommending men-

tors to software project newcomers. In: 3rd International Workshop on Recommendation Systems for Software Engineering. IEEE, Washington, DC, USA, pp. 63–67.

[40] Stol, K.-J., Avgeriou, P., Ali Babar, M., 2010. Identifying architectural patterns used in open source software: approaches and challenges. In: Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering. EASE'10. British Computer Society, Swinton, UK, UK, pp. 91–100.
URL http://dl.acm.org/citation.cfm?id=2227057.2227069

[41] Suh, B., Convertino, G., Chi, E. H., Pirolli, P., 2009. The singularity is not near: Slowing growth of Wikipedia. In: Proceedings of the 5th International Symposium on Wikis and Open Collaboration. WikiSym '09. ACM, New York, NY, USA, pp. 8:1–8:10.
URL http://doi.acm.org/10.1145/1641309.1641322

[42] Von Krogh, G., Spaeth, S., Lakhani, K., 2003. Community, joining, and specialization in open source software innovation: A case study. Research Policy 32 (7), 1217–1241.
URL http://www.sciencedirect.com/science/article/pii/S0048733303000507

[43] Wang, J., Sarma, A., 2011. Which bug should I fix: Helping new developers onboard a new project. In: Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering. CHASE '11. ACM, New York, NY, USA, pp. 76–79.
URL http://doi.acm.org/10.1145/1984642.1984661

[44] Wang, Y.-C., Kraut, R., Levine, J. M., 2012. To stay or leave?: The relationship of emotional and informational support to commitment in online health support groups. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. CSCW '12. ACM, New York, NY, USA, pp. 833–842.
URL http://doi.acm.org/10.1145/2145204.2145329

[45] Weiss, M., Moroiu, G., Zhao, P., 2006. Evolution of open source communities. In: Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G. (Eds.), Open Source Systems. Vol. 203 of IFIP International Federation for Information Processing. Springer Boston, Como, Italy, pp. 21–32.
URL http://dx.doi.org/10.1007/0-387-34226-5_3

[46] Zhou, M., Mockus, A., 2011. Does the initial environment impact the future of developers. In: Proceedings of the 33rd International Conference on Software Engineering. ICSE '11. ACM, New York, NY, USA, pp. 271–280.
URL http://doi.acm.org/10.1145/1985793.1985831

[47] Zhou, M., Mockus, A., 2012. What make long term contributors: Willingness and opportunity in OSS community. In: Proceedings of the 34th International Conference on Software Engineering. ICSE '12. IEEE Press, Piscataway, NJ, USA, pp. 518–528.
URL http://dl.acm.org/citation.cfm?id=2337223.2337284