

# Are You Still Working on This? An Empirical Study on Pull Request Abandonment

Zhixing Li, Yue Yu\*, Tao Wang, Gang Yin, ShanShan Li, and Huaimin Wang

**Abstract**—The great success of numerous community-based open source software (OSS) is based on volunteers continuously submitting contributions, but ensuring sustainability is a persistent challenge in OSS communities. Although the motivations behind and barriers to OSS contributors’ joining and retention have been extensively studied, the impacts of, reasons for and solutions to contribution abandonment at the individual level have not been well studied, especially for pull-based development. To bridge this gap, we present an empirical study on pull request abandonment based on a sizable dataset. We manually examine 321 abandoned pull requests on GitHub and then quantify the manual observations by surveying 710 OSS developers. We find that while the lack of integrators’ responsiveness and the lack of contributors’ time and interest remain the main reasons that deter contributors from participation, limitations during the processes of patch updating and consensus reaching can also cause abandonment. We also show the significant impacts of pull request abandonment on project management and maintenance. Moreover, we elucidate the strategies used by project integrators to cope with abandoned pull requests and highlight the need for a practical handover mechanism. We discuss the actionable suggestions and implications for OSS practitioners and tool builders, which can help to upgrade the infrastructure and optimize the mechanisms of OSS communities.

**Index Terms**—Pull Request Abandonment, Pull-based Development, Open Source Software

## 1 INTRODUCTION

The great success of open source software (OSS) is based on the theory that “many hands make light work” [57], [73]. For a popular community-based OSS project, a large number of volunteers continuously improve the project quality [52], [75] by submitting code patches, reporting bugs, and discussing new features. However, due to the nature of OSS collaboration (*e.g.*, OSS projects are loosely organized as the core-periphery structure [16], [17], and many external contributors are motivated by indirect economic benefits [40], [48]), ensuring sustainability [21] is a persistent challenge in OSS communities.

Prior work in this area has paid considerable attention to: *i)* revealing the motivations behind why developers make contributions to OSS projects to better recruit and encourage contributors [9], [30], [40], [41], [58], [75]; *ii)* uncovering the barriers and challenges faced by developers when contributing to OSS projects to assist the onboarding and participation of developers [15], [19], [29], [64], [66], [74]; and *iii)* investigating the factors that affect OSS developers’ willingness to remain long-term and established contributors to decrease the likelihood of developer disengagement and turnover [41], [43], [47], [54], [59], [78].

In this study, we shed light on OSS contributors’ abandonment, focusing on the individual and single level in pull-based development [23], *i.e.*, every single pull request

that is unfinished but abandoned by the author. We clarify our research motivation as follows. The pull request mechanism [23], [71] provides a synthesized collaboration environment for OSS distributed development by coupling the code repository, modern issue tracker, code review and automatic tools of DevOps. This has unprecedentedly lowered the barrier to entry for potential contributors and simplified the collaborative development process compared to traditional patch-based methods [80]. It is interesting to understand the reasons why there are considerable unfinished pull requests whose contributors have spent time programming, submitting and discussing at the beginning but walk away, leaving their contributions abandoned. As project integrators mentioned in our survey, “We see quite a lot of valuable contributions abandoned by authors, just applying a few demanded changes might make their pull requests good enough to be eventually accepted” [SC224]. Furthermore, pull request abandonment is significant for the health of OSS projects and requires extra project maintenance efforts. Two representative quotes from our survey respondents indicate this: “Opening a PR and walking away is bad form” [SI3], and “It’s frustrating to see a bug almost fixed and then abandoned” [SI6]. A more complete picture of pull request abandonment can help OSS communities obtain great benefits from designing appropriate strategies to overcome avoidable abandonment, building more efficient tools for distributed collaboration, and helping individual participants understand the mindset of abandonment, and find the best ways to contribute.

To address this goal, we conducted a mixed-methods empirical study. We first manually examined the review discussion of 321 abandoned pull requests collected from 5 popular projects hosted on GitHub, which allowed us to obtain a preliminary understanding from an observer’s perspective based on historic trace data. Then, we confirmed

- Zhixing Li, Yue Yu, Tao Wang, Gang Yin, and Huaimin Wang are with the Key Laboratory of Parallel and Distributed Computing, College of Computer, National University of Defense Technology, Changsha, China. E-mail: {lizhixing15, yuyue, taowang2005, yingang, hmwang}@nudt.edu.cn
- ShanShan Li is with the College of Computer, National University of Defense Technology, Changsha, China. E-mail: shanshanli@nudt.edu.cn

\*Corresponding author: Yue Yu, yuyue@nudt.edu.cn

and quantified the manual observation results by large-scale surveys, which received 619 and 91 answers from contributors and integrators, respectively. By analyzing all these data, we answered three research questions:

**RQ1.** *Why do contributors abandon their pull requests?*

We have identified 12 main reasons for pull request abandonment related to the limitations of the collaboration process, personal issues of contributors, and implementations of pull requests *per se*. In addition to the lack of responsiveness from integrators and lack of time from contributors, we observed surprising reasons concerning the pull request updating process, consensus-reaching discussion, and effort investment. Our findings can help OSS communities clarify responsibility, optimize the collaboration process, and focus their efforts on avoidable reasons to prevent abandonment.

**RQ2.** *What are the impacts of pull request abandonment?*

We found that the top-ranked impacts primarily introduce extra maintenance burden on integrators, *e.g.*, cluttering pull request list. Abandonment can also have cascading effects on the community due to technical dependencies among artifacts and common interests among developers. By learning the specific impacts, new mechanisms and tools can be developed to mitigate the undesirable impacts.

**RQ3.** *How do integrators cope with abandoned pull requests?*

We have found several strategies integrators used to promote the completion of abandoned pull requests. However, we also uncovered a novel need for a handover mechanism in social coding platforms, since approximately two-thirds of integrators simply closed abandoned pull requests although most contributors expressed that they were willing to pick up pull requests abandoned by others. Based on our findings, tool builders can be informed to better fulfill developers' information needs and functionality requirements, which are not currently met.

The contributions of the paper are summarized as follows:

- To the best of our knowledge, this is the first in-depth study that systematically explores the phenomenon of pull request abandonment in OSS projects.
- We provide qualitative information and quantitative assessment about the reasons, impacts, and coping strategies for pull request abandonment based on manual observation and surveys.
- We propose actionable recommendations and implications that OSS practitioners and tool builders can take to increase the potential to prevent contributors' abandonment and foster the completion of abandoned pull requests, thus improving the overall development efficiency.

The remainder of the paper is organized as follows. Section 2 presents the background of abandoned pull requests and related work. Section 3 describes the methodology, and Section 4 reports the results with respect to our three research questions. Section 5 presents our main findings and their implications. Section 6 discusses the threats to the validity of our study. The last section concludes the paper.

## 2 BACKGROUND AND RELATED WORK

This section presents the background of abandoned pull requests and points out the main studies in the field.

### 2.1 Abandoned pull requests

Figure 1 illustrates the review process of an individual pull request. When a new pull request is submitted by a contributor, project integrators are responsible for evaluating the changes within the pull request. If integrators do not like the proposed changes, they reject the pull request. However, if integrators are satisfied with the pull request, they would accept it. Nevertheless, pull requests are rarely perfect and ready to merge when submitted. Pull requests are usually iteratively evaluated and updated, until a satisfactory solution has emerged. In each round of review, integrators evaluate the contribution quality and ask the author to fix the identified defects. If the author updates the pull request by submitting new changes as requested, the pull request awaits for another round of review. However, some contributors might vanish without addressing integrators' comments, leaving their pull requests abandoned.

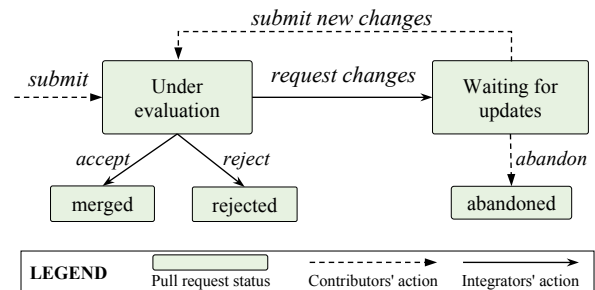


Fig. 1: Review process of an individual pull request.

As an example, Figure 2 describes an abandoned pull request from the project `Rails`. ❶ The author created a commit and submitted a pull request for review. ❷ Developer `dvpr1` reviewed the code and suggested changes. Then the author submitted a new commit addressing the suggestion and pinged integrators to review the code again. ❸ At almost the same time, developer `dvpr2` proposed another suggestion that was approved by the other two reviewers. ❹ However, the author became inactive in the next few weeks, leading integrators to discuss the status and value of the pull request. ❺ For another few months, the author still remained inactive. Developer `dvpr3` posted a prompting comment and tried to convince the author of the benefit of the requested changes. The author was also asked to rebase the pull request. ❻ Unfortunately, the author was still inactive in the next few weeks, and integrators finally closed the pull request.

The fact that the review of the pull request took 5 months, during which 8 developers participated in the discussion generating 12 comments and some kind of testing method was performed to test the code, reveals that pull request abandonment can be unfortunate since valuable human and material resources have been utilized. Moreover, the contributor herself/himself also invested effort and time in creating, submitting and updating the pull request. This motivates us to investigate the reasons behind pull request abandonment and seek inspiration to prevent avoidable abandonment.

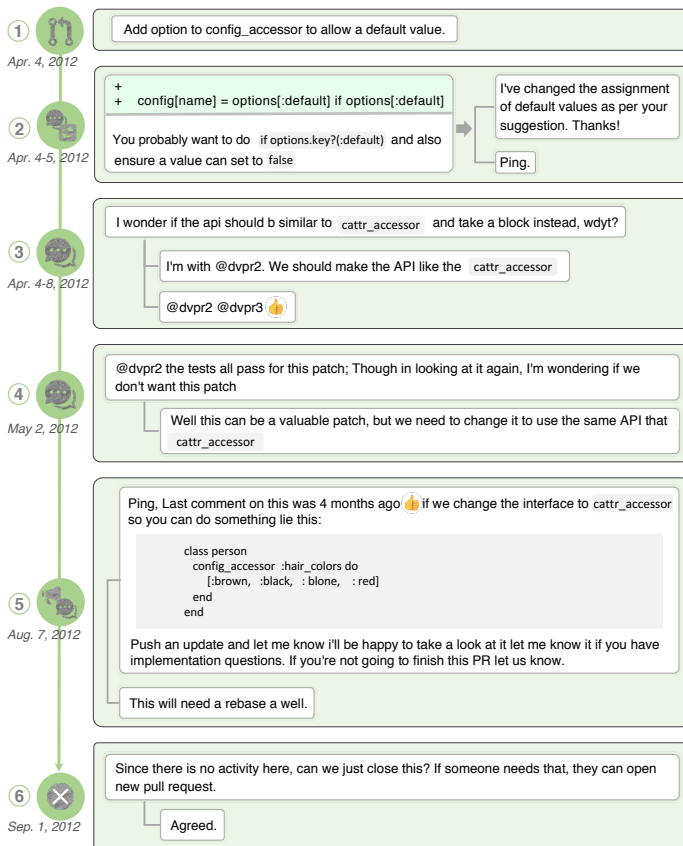


Fig. 2: An example abandoned pull request.

## 2.2 Pull request nonacceptance

Gousios *et al.* [23] manually examined 350 pull requests to explore the reasons for closing a pull request without merging. They observed that far more pull requests are unmerged due to issues related to the distributed development process rather than technical issues. Steinmacher *et al.* [65] presented a study regarding quasi-contributors' perceptions of why their pull requests were not accepted. They found that duplicates and vision mismatch were the most common reasons mentioned by contributors. Additionally, they reported that developers' abandonment (*i.e.*, "Lack of experience/commitment from quasi-contributors" in their study) was cited by some developers as the reason for pull request nonacceptance. Past studies have also extensively investigated the factors affecting integrators' decisions to accept or reject a pull request. For example, Gousios *et al.* [23] showed that the merge decision is mainly influenced by modification hotness. Tsay *et al.* [68] presented that project integrators examine both technical and social factors, and particularly contributors' prior interaction in the project significantly affects pull request acceptance. Recent research has extended previous work by considering the effects of additional factors such as personality traits [34], geographical location [55], [56], employment [39], [51] and gender [67].

The above research studied pull request nonacceptance in general. However, although some of the findings showed that contributors' abandonment is one of the reasons for pull request nonacceptance, to the best of our knowledge, no prior work in the literature has studied this topic in detail.

## 2.3 OSS projects sustainability

The sustainability of OSS projects has attracted great interest from researchers. Previous studies have investigated the motivations behind and barriers to developers' joining and retention in OSS projects.

**Motivations to make OSS contributions.** Developers' motivations to contribute to OSS projects have been extensively studied by prior research [30], [40], [75]. The common motivations include the joy of programming, the identification with a community, career advancement and learning. Furthermore, Roberts *et al.* [58] investigated the interrelationships between the motivations of OSS developers. They found that developers' motivations are not always complementary. Lee *et al.* [41] particularly studied the motivations of one time contributors. They found that the most common motivation is to fix a bug affecting developers. Similarly, Pinto *et al.* [52] showed that "scratch their own itch" was a highly mentioned motivation by casual contributors. Bonaccorsi *et al.* [9] studied the difference in motivations between individual developers and firms and observed that firms are more motivated by economic and technological reasons.

**Barriers to participation in OSS projects.** Steinmacher *et al.* [66] analyzed newcomers' first interactions on a project and found that both the authors and types of received answers affected newcomers' onboarding. Furthermore, they defined a conceptual model composed of 58 barriers [63] that hinder newcomers' first contribution and designed a portal [64] to help newcomers overcome these barriers. Researchers have also investigated the effect of mentoring on onboarding newcomers. For example, the study by Fagerholm *et al.* [19] showed that mentoring increases the chance of developers' active participation. Canfora *et al.* [15] proposed Yoda, a system that identifies and recommends suitable mentors to support newcomers joining a project. In the pull-based model specifically, Gousios *et al.* [24] surveyed developers about the challenges they faced, and found that the most commonly reported challenge is the lack of responsiveness from project integrators.

**Developers' retention in OSS projects.** A study by Zhou *et al.* [78] showed that developers' willingness and participation environment have a significant influence on the probability of developers becoming long-term contributors. Schilling *et al.* [59] revealed that the level of development experience and conversational knowledge significantly affect developers' retention in OSS projects. A study by Lin *et al.* [43] showed that developers who contribute earlier, mainly modify instead of creating files, and mainly write code rather than maintaining documentation have a higher chance of retention. Lee *et al.* [41] found that most of the one-time contributors simply shared the fixes to the bugs impeding their work and had no intention to become long-term contributors. They also observed that time and process are the main reasons that prevent contributors from submitting additional patches to a project. Qiu *et al.* [54] studied the effect of social capital on developers' participation, and found that collaborating with familiar developers is generally beneficial for prolonged engagement. Miller *et al.* [47] studied why established contributors disengage from OSS projects. Their survey with disengaged contributors showed that occupational problems, *e.g.*, changing to a new

job that does not support OSS development work, are the most frequent reasons. Iaffaldano *et al.* [33] conducted an interview with OSS developers to explore what drives them to become temporarily or permanently inactive in a project. The reported reasons were personal (*e.g.*, life events) or project related (*e.g.*, role change and changes in the project).

Existing research has broadly studied each stage of OSS developers’ lifecycle. Some of this research has particularly explored the reasons for developers’ disengagement in OSS projects. However, little is known about developers’ disengagement in individual contributions. Developers who leave a project might have never abandoned any pull request, while developers who have abandoned a pull request might continue to contribute to a project. For OSS projects to sustain, it is important not only to attract and retain developers to make long-term contributions but also to motivate developers to complete each submitted pull request. In this paper, we therefore address this knowledge gap by focusing on pull request abandonment.

### 3 RESEARCH METHODOLOGY

Inspired by prior established guidelines [7], [31], [44], [45], to answer our research questions, we conducted a mixed-methods empirical study using both qualitative and quantitative approaches. At a high level, our research methodology comprised two components, *manual observation* and *online surveys*, as shown in Figure 3. First, we conducted an exploratory qualitative observation by manually examining the public trace data stored in GitHub repositories. This manual observation aimed to obtain a preliminary and broad understanding. We built a taxonomy of the reasons, impacts, and coping strategies for pull request abandonment. Then, the preliminary findings were used as input for surveys which helped us quantify the manual observation results. Combining the results of the observation and the surveys, we derived our main findings.

#### 3.1 Manual observation

The manual observation was based on five popular OSS projects. We first identified a collection of abandoned pull requests through a semi-automatic method from the five projects. Then, by analyzing pull request discussions, we aimed to develop preliminary answers to the three research questions.

##### 3.1.1 Studied projects

In this paper, we accessed five OSS projects hosted on GitHub, *i.e.*, Rails, Kubernetes, Node.js, Cocos2d-x, and Rust. We selected these projects for multiple reasons. First, all five projects are collaboratively developed, following the pull request model and using the integrated code review tool in GitHub. Second, the projects are popular and mature, with years of development history, gaining widespread community attention. Third, they are diverse in terms of programming language and application domain. Table 1 presents the overview of the studied projects. The number of stars and the number of pull requests are proxies for project popularity [10]. For each of the studied projects, we collected all pull requests from the project creation date to the examination date (August 15, 2019) as well as review discussions on pull requests via the GitHub API [2].

TABLE 1: Overview of our studied projects.

Projects	Language	Domain	#Star	#PR
Rails	Ruby	Web framework	36,330	13,820
Kubernetes	Go	Container management	24,965	29,729
Node.js	JS	JavaScript runtime	37,030	18,409
Cocos2d-x	C++	Game engine	10,550	11,915
Rust	Rust	Programming language	22,595	47,522

##### 3.1.2 Identification of abandoned pull requests

Abandoned pull requests are closed but unfinished pull requests in which the authors did not address integrators’ change requests. However, GitHub does not precisely mark abandoned pull requests with a specific status value such as *open*, *closed*, or *merged*. One cannot directly and easily retrieve abandoned pull requests by setting selection criteria when calling the GitHub API or searching any public dataset such as GHTorrent [22]. Consequently, we chose a heuristics-based method.

According to the interaction between integrators and the pull request author after the change request, abandoned pull requests can be classified into four categories: *i)* the author spontaneously reported her/his abandonment decision, *ii)* the author remained inactive and integrators directly closed the pull request, *iii)* the author remained inactive and then reported the abandonment decision when pinged by integrators, and *iv)* the author consistently remained inactive even when pinged by integrators and the pull request was ultimately closed. We can note that pull requests of the last three cases experience an “inactive” period that tends to cause a pull request to receive comments concerning contributor responsiveness, *e.g.*, “closing due to inactivity” and “any update on this?“. These comments usually contain representative keywords that are useful to guide the identification of abandoned pull requests. Our preliminary observation also showed that pull requests of case *i)* were rare. To this end, we first automatically identified candidates for abandoned pull requests by examining the presence of unresponsiveness-related keywords in review comments and then manually verified the identified candidates.

**Automatic identification.** First, we preprocessed the text of a comment by removing code snippets and hyperlinks and converting all the words to lower case. Then, we determined whether any of the following unresponsiveness-related keywords was contained in the processed comment.

```
{ lack of response, lack of activity, lack of feedback, inactive, inactivity, no response, no reply, no activity, no feedback, any update, any news, any activity, unfinished, uncompleted, unresponsiveness }
```

Initially, the keyword list only contained several items extracted from a set of known abandoned pull requests. Subsequently, it was iteratively extended when new representative keywords were found from the newly identified abandoned pull requests. If a comment was successfully matched with at least one of the keywords, the pull request to which it belonged was marked as a candidate for an abandoned pull request. In total, we automatically identified

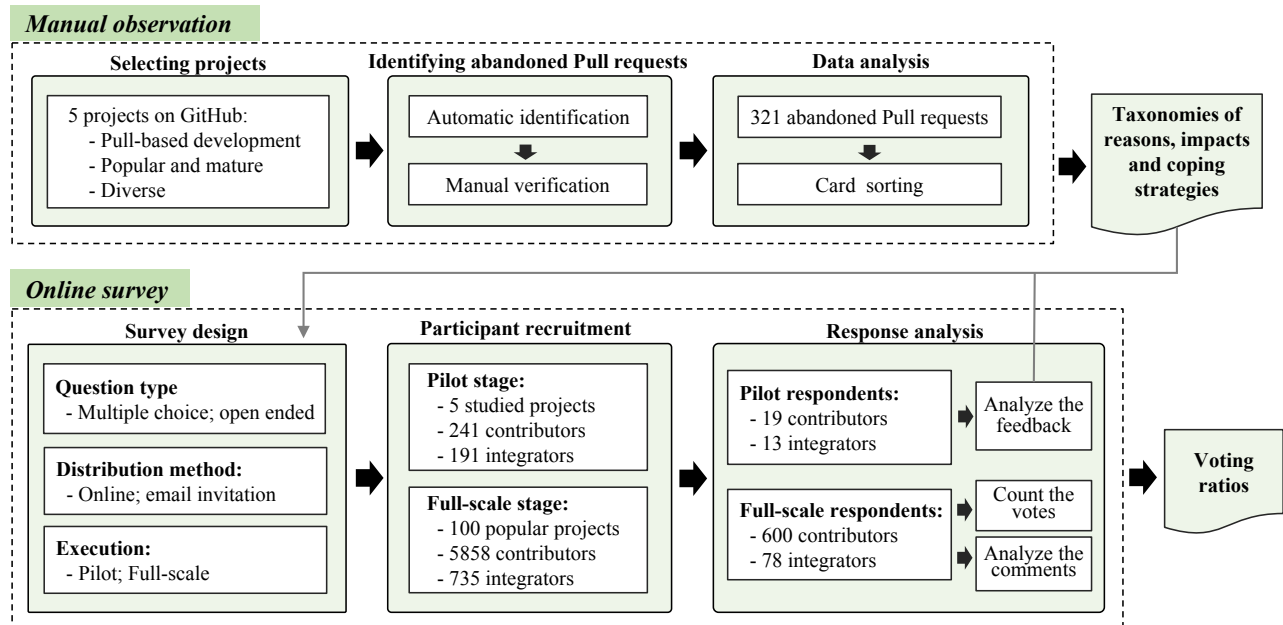


Fig. 3: Overview of methodology design.

1,125 candidates.

**Manual verification.** Unresponsiveness does not always indicate abandonment, *i.e.*, the heuristics used in automatic identification can introduce false positives. For example, inactive authors might respond after they were pinged by integrators. Moreover, the keywords might have been used in a different context other than contributor responsiveness, *e.g.*, the keyword *inactive* can be used to describe the state of a remote service. Currently, there is not a feasible way to increase precision without decreasing recall, *i.e.*, filtering out as many false positives as possible without mistakenly removing true positives simultaneously. Therefore, we manually examined each candidate and verified whether it was really abandoned by the author. At the end of this process, we obtained a total of 321 abandoned pull requests.

### 3.1.3 Data analysis

For each abandoned pull request, we went through its review discussion in full and extracted *i)* the author's explanation for why s/he could not finish the pull request (*e.g.*, "I am very sorry, but I am not interested in this topic anymore" [D30]), and *ii)* integrators' comments and activities around the pull request abandonment (*e.g.*, "This had no followup after a review. Is there anyone that wants to help on this?" [D75]). The extracted data were qualitatively analyzed by following the card sorting procedures [62], [81] to build taxonomies of the reasons, impacts and coping strategies for pull request abandonment. Taking as an example of the emergence of the taxonomy of abandonment reasons, our analysis process proceeded as follows (the other two taxonomies were established following the same process).

**Preparation:** The first three authors were assigned to conduct the card sorting task. They reviewed all the explanation comments and created a "card" for each of them. The cards were created by selecting the key phrases and sentences from the comments.

**Coding:** Instead of coding the cards separately in parallel and checking the consistency of the coding results, the three authors coded the cards together [8], [28]. Agreement was negotiated along the way [14], *i.e.*, when the coders had different opinions, they interrupted the process to discuss the discrepancy before continuing on. As we had no predefined categories, the coders used an open coding approach [81] to analyze the cards, whereby new themes emerged during the coding process. By reading the cards, they grouped them into meaningful categories; each category had a theme describing the reason for abandonment. Although the card sort reached saturation early (*i.e.*, when no new information was gained from new data), we decided to code all cards to increase the completeness of the discovered set of themes. Finally, after all the cards were analyzed, a taxonomy of reasons for pull request abandonment was established.

**Validation:** To minimize the subjectivity of the three coders, all authors discussed the taxonomy for conceptual validation and checked its consistency with the raw data. This discussion refined the taxonomy by reorganizing some categories and rewording the names of some categories, ensuring that the taxonomy was clear and understandable without any misunderstanding. Additionally, to further validate the taxonomy with respect to project popularity, we performed a closed card sorting on additional data collected from 20 less-popular projects [5]. Finally, no new themes emerged in the closed sort.

## 3.2 Survey

In this section, we introduce the design, participant recruitment, and response analysis of the surveys.

### 3.2.1 Survey design

Since pull request contributors and project integrators are involved in different aspects of pull request abandonment,

we designed two different surveys that can be found online [6]. The survey for contributors was mainly for abandonment reasons, while the survey for project integrators was mainly for the impacts and coping strategies of abandonment. Both surveys started with an introduction to the research background and purpose. The questions in the surveys consisted of two parts: *i*) demographic questions that were designed to obtain the participants’ role, experience and work practice in OSS development, and *ii*) main questions that were designed to quantify the insights that we obtained from manual observation. All the questions were multiple-choice questions, most of which included an optional “Other” text field to allow survey participants to provide additional answers that we did not offer. The predefined answers to demographic questions were inspired by prior research [24], [25], and the predefined answers to the main questions were set according to the manual observation findings. Finally, we asked the participants an open-ended question to allow them to freely provide feedback on anything that they thought might help us understand the problem of pull request abandonment. Before deploying the surveys on SurveyMonkey [4], one of the most famous platforms used for online surveys, we discussed them with software engineering researchers with experience in OSS and survey design to ensure that the questions were clear and appropriate.

In addition, each survey was conducted in two rounds: *i*) a pilot stage that targeted a limited number of participants in order to clarify our questions and enrich the emerging themes that we could further validate, and *ii*) a full-scale stage that targeted a broader population in order to vote on answers to each question.

### 3.2.2 Participant recruitment

In the pilot stages, we surveyed developers from the five projects studied in the manual observation. We targeted contributors who submitted abandoned pull requests, and project integrators who reviewed abandoned pull requests, respectively. In the full-scale surveys, we wanted to investigate how frequently the identified themes occurred in a broader populations. To this end, we surveyed developers coming from more projects. We first obtained the list of the top 1,000 popular projects hosted on GitHub, which had the most number of stars. Then, we randomly selected 100 projects from them, which were software projects (identified by programming language) and used the pull-based development model. From each of the selected projects, we selected the 100 most recent closed pull requests. The authors of selected pull requests and project integrators who reviewed these pull requests were selected as candidate participants. For our sample, we targeted developers who were more likely to have fresh experience with pull request abandonment.

We obtained the email addresses of candidate participants by analyzing their GitHub profile pages and commit logs. Duplicate developers were removed by comparing email addresses and user names. In this way, we identified 432 individuals in the pilot surveys, including 241 contributors and 191 integrators, and 6,593 individuals in the full-scale surveys, including 5,858 contributors and 735 integrators. Finally, the surveys were published online and

the web addresses were sent to target participants via email. The invitation message included the number of questions and the estimated time required to complete the survey. All surveys run for two weeks.

### 3.2.3 Responses and analysis

In the pilot surveys (December 20, 2019 - January 3, 2020), we successfully sent 407 invitations and got 32 responses (7.9% response rate), including 19 from contributors and 13 from integrators. Based on the participants’ feedback, we improved the questionnaires to be used in the full-scale surveys by adding specific options to two yes/no questions, supplementing the options set to two questions, and rewording several questions.

In the full-scale surveys (January 14-28, 2020), we successfully sent 6,114 invitations and got 678 responses including 600 from contributors and 78 from integrators, for a total response rate of 11.1%, which is similar to response rates reported by prior software engineering research [24], [49], [82]. For each multiple-choice question, we analyzed its responses in two ways. First, we counted the votes on each of the predefined answer options and computed the voting ratios of each option. Second, we used the closed coding method [81] to classify the textual replies of the “Other” field, if any, into its corresponding taxonomies. If new themes emerged, they were integrated into the existing taxonomy. For the last open-ended questions, we collected meaningful replies and qualitatively analyzed the text content.

When analyzing the quantitative data regarding demographic information from the full-scale surveys, we found that the majority of our respondents were industry developers, and more than half of them had over three years of OSS development experience. We also found that one third of contributors submitted more than three pull requests per month, and approximately half of the integrators managed more than three projects.

TABLE 2: Demographic information of survey respondents.

Question	Contributors	Integrators
	Industry Dev.: 66%	Industry Dev.: 81%
Q1. How would you best describe yourself?	Student: 17%	Student: 5%
	Independent Dev.: 10%	Independent Dev.: 5%
	Academic: 4%	Academic: 5%
Q2. OSS development experience in year?	<1: 15%; 1-3: 30%	<1: 7%; 1-3: 24%
	3-5: 18%; >5: 33%	3-5: 20%; >5: 49%
Q3. Average number of PRs per month? (contributors)	Not sure: 25%; 1-3: 42%	-
	3-5: 10%; >5: 22%	
Q3. Number of maintained OSS projects? (integrators)	-	1-3: 55%; 3-5: 22%
		5-10: 8%; >10: 13%














## 4 RESEARCH RESULTS

In this section, we present our findings from the manual observation. We also report the frequency distributions of answers collected from the full-scale surveys. Representative quotes are presented to support the findings. The source of each quote is noted in square brackets:  $[Dx]$  refers to discussion comments from trace data, and  $[SCx]$  and  $[SIx]$  refer to survey comments from contributors and integrators. The content is organized around our three research questions.

#### 4.1 RQ1: Why do contributors abandon their pull requests?

As shown in Table 3, we identified 12 main reasons for pull request abandonment at three different aspects, *i.e.*, *process and social limitations during collaboration, personal reasons of contributors, and implementations of pull requests per se*. Reasons  $\mathcal{R}1$ - $\mathcal{R}11$  were identified in the manual observation, and reason  $\mathcal{R}12$  was added from the pilot survey with contributors. Table 3 also lists the frequency of reasons voted by respondents in the full-scale survey with contributors. Note that the total frequency is greater than 100% because survey respondents may have selected more than one reason in their answers. In the following, we analyze and discuss each of those reasons in detail.

TABLE 3: Reasons why contributors abandon their pull requests.

Reason	Aspect	Votes(%)
$\mathcal{R}1$ Lack of answer from integrators	Process	42.4 
$\mathcal{R}2$ Lack of time	Personal	36.6 
$\mathcal{R}3$ Pull request is obsolete	Pull request	32.8 
$\mathcal{R}4$ Not treated seriously by integrators	Process	28.6 
$\mathcal{R}5$ Inadequate community demand	Pull request	22.8 
$\mathcal{R}6$ Lack of interest	Personal	22.3 
$\mathcal{R}7$ Lack of knowledge about the requested changes	Personal	20.0 
$\mathcal{R}8$ Integrators do not reach a consensus	Process	17.7 
$\mathcal{R}9$ Tasks of higher priority	Personal	15.8 
$\mathcal{R}10$ It requires more effort than anticipated	Pull request	15.5 
$\mathcal{R}11$ Disagree with integrators' opinions	Process	13.8 
$\mathcal{R}12$ Tedious process of updating the pull request	Process	13.4 
Other	---	1.2 

##### A) Process-related reasons.

$\mathcal{R}1$  *Lack of answer from integrators*. 42.4% of contributors cited a lack of answers from integrators as a reason for their abandonment. Contributors might ask for confirmation of the change detail (*e.g.*, “What do you mean by ‘type in code’” [D18]) or approval of the following work (*e.g.*, “I think I need another core committer to champion this effort or I’m afraid it’ll code-rot again” [D51]). Waiting for integrators’ answers is a blocking step for contributors. If contributors do not receive a reply, they might move away (*e.g.*, “in example trying to get responses from [some OSS project] is frequently like pulling teeth. It has turned me off to working with them” [SC533]).

$\mathcal{R}4$  *Not treated seriously by integrators*. 28.6% of respondents reported that they walked away because they were not treated seriously by integrators. For instance, a typical response has clearly pointed out that “More than once, the feedback from reviewers took months.” [SC299]. Additionally, one respondent declared that he became very upset after several times of re-basing requests without receiving any constructive comment (*e.g.*, “I’ve re-based the PR three times already. I’ve brought it to your attention several times. I’ve received no direction on what’s hold this up.” [D70]). Contributors also suggested that the long latency in waiting for the first review comment demotivated them from replying back (*e.g.*,

“The longer the period between submission and first interaction with reviewer, the more likely I am to abandon if the reviewer requests changes” [SC617]).

$\mathcal{R}8$  *Integrators do not reach a consensus*. 17.7% of contributors abandoned their pull requests due to the lack of consensus among integrators. Sometimes, it can be fairly straightforward for reviewers to reach an agreement on how a pull request should be improved. However, in some cases where more than one solution is available and each one has its advantages and disadvantages, the disagreement among integrators arises during the review process. The extended discussions and inconsistent requirements might hinder contributors’ patience in updating their pull requests (*e.g.*, “I would just need someone with the authority to make the decision which approach (using usable\_size or going with next pot) to go forward with” [D61]).

$\mathcal{R}11$  *Disagree with integrators’ opinions*. 13.8% of respondents cited this reason. Contributors may be far from the direction that project integrators want to take due to their different ideas and visions [32]. In some cases, integrators thought some changes would be necessary to perfect the pull request, while contributors disagreed (*e.g.*, “I’m looking into that now. The issue that we ... which is basically a nop so I don’t think there is any cleanup to do..” [D64]). A prior study [25] reported that it is difficult for integrators to ask for more work from contributors, let alone when contributors disagree with them.

$\mathcal{R}12$  *Tedious process of updating the pull request*. To update a pull request, contributors need to amend existing commits or add new commits and then push the updated branch to GitHub. If other developers have changed the master branch, contributors are expected to rebase their local branch on top of the latest changes and deal with the conflicts, if any. Sometimes, contributors may be asked to squash commits together to cut out the noise in the revision history. 13.4% of contributors complained about the tedious process of updating the pull request. Most of them believed that the tedious updating process does not make it cost-effective to carry out trivial changes, *e.g.*, removing extra space. Others suggested that the git facilities used in the updating process might be sufficiently difficult to deter some inexperienced developers from continuing, especially in regard to newcomers.

##### B) Personal reasons.

$\mathcal{R}2$  *Lack of time*. 36.6% of respondents stated that they did not address the requested changes because they had no time to make additional OSS contributions. Many OSS contributors are driven by enjoyment-based intrinsic motivations [40], [41], [52], instead of real payment. In our study, we found that abandoners’ time might be occupied by a full-time job (*e.g.*, “Sadly with my current job i don’t really have enough time to work on this” [D41]), school tasks (*e.g.*, “little bussy with my move and starting my PhD, [someone] was going to rebase this” [D39]), etc.

$\mathcal{R}6$  *Lack of interest*. Lack of interest as a reason was cited by 22.3% of respondents. Interest is crucial to keep OSS developers contributing since many of them are volunteer contributors. If contributors only maintain a shallow relationship with a project, particularly for newcomers and casual contributors to an OSS project, their interests might

fade when their pull requests suffer from an unexpected or overly long review interaction (e.g., “I am very sorry, but I am not interested in this topic anymore.” [D30]).

**R7 Lack of knowledge about the requested changes.** 20.0% of respondents abandoned their pull requests because they lacked the necessary domain knowledge and programming skills to carry out the requested changes (e.g., “I don’t think I’m familiar enough with the rustc codebase to go through and audit all the uses of `?: vs .` . I’d be okay with closing this” [D44]). Although knowledge transfer [7] and mentoring [78] are common on modern social coding sites, some contributors, especially casual contributors, appeared unwilling to learn the required knowledge.

**R9 Tasks of higher priority.** Developers might work on multiple OSS tasks during a specific period to increase their productivity. Tasks can be scheduled and switched according to their priorities. Task-switching, however, comes at a cost [70]. Developers need to recall the goal and reconstruct the context to switch to another task [11], which might require considerable energy and significantly hinder the current work status. Consequently, 15.8% of respondents said that concentrating on current work was their first priority and they had to give up the pull request (e.g., “Hey, sorry about the delay. Regrettably things keep coming up that are higher priority than this” [D64]).

#### C) Pull request-related reasons.

**R3 Pull request is obsolete.** 32.8% of respondents explained that their pull requests were no longer required or applicable. This reason includes cases in which a similar pull request was submitted and accepted (e.g., “Someone else submitted a pull request that fixed the same issue” [SC299]). In other cases, the problem disappeared because codes of popular projects can be frequently changed and become outdated (e.g., “becomes unnecessary with arrival of ES modules/v8 devtools progress” [D62]).

**R5 Inadequate community demand.** 22.8% of contributors abandoned their pull requests because they did not see enough community demand. Other developers’ approval of the relevance and value of pull requests is an important motivation for contributors to continually improve and complete their work [24], [25]. If contributors do not feel obvious and strong desire for their pull requests, they are less likely to perform the requested changes because they conjecture that their work would ultimately be rejected due to a lack of community demand (e.g., “it seems like reviewers are not interested in the feature I have proposed” [SC267]).

**R10 It requires more effort than anticipated.** 15.5% of respondents indicated that they walked away because the required changes exceeded the effort they planned to devote. In some cases, the pull request turned out to be much more complicated than anticipated and contributors did not want to spend more time (e.g., “it was taking too much of my time” [SC144]). In other cases, contributors just submitted a drive-by pull request, e.g., they implemented a feature for their personal needs and sent it back to the community. At the project level, integrators asked contributors to generalize their solutions to cover a wider range of cases. However, contributors were unwilling to go through the effort of performing the requested changes (e.g., “I’m happy to provide

an initial technical fix that may be limited in scope. But the projects need to do their own work” [SC523]).

#### D) Other reasons.








We also collected several one-off responses from the “Other” field, that did not fit in the above categories. The mentioned reasons include missing GitHub notification, strange and irrelevant test failure, project failure, and inefficient communication mechanisms. Interestingly, one contributor said the abandonment was due to the lack of financial incentives.

**RQ1:** We identified 12 main reasons for pull request abandonment. Reasons related to the collaboration process are generally mentioned by surveyed contributors. The most common reasons were the lack of answers from integrators and the lack of time from contributors. Interestingly, tediousness in performing the updating process, tardiness in reaching consensus on change direction, and more effort than anticipated to get patches accepted can also cause abandonment, although relatively uncommonly.

#### 4.2 RQ2: What are the impacts of pull request abandonment?

Table 4 shows the 7 impacts of pull request abandonment. Impacts  $I2-I7$  were identified in the manual observation, and  $I1$  was added from the pilot survey with integrators. The frequency of impacts gathered from the full-scale survey with integrators is also listed in the table. The remainder of this section discusses each of these impacts in detail.

TABLE 4: Impacts of pull request abandonment.

Impact	Votes (%)
$I1$ Cluttered pull request list	64.0 
$I2$ Wasted review effort	62.7 
$I3$ Additional attention for more careful close	57.3 
$I4$ Delayed landing of inter-dependent pull requests	29.3 
$I5$ Duplicate pull requests	26.7 
$I6$ Disordered milestone	18.7 
$I7$ Bad impression	10.2 

**$I1$  Cluttered pull request list.** The review duration of pull requests can vary from days to months [77]. On busy projects, the volume of incoming pull requests can be large, leading to many open pull requests being reviewed simultaneously. Contributors who abandon pull requests without leaving clear statements will clutter the pull request list. This creates extra work for integrators who have to check on each of the open pull requests for recent progress (e.g., “It’s hard to keep track of things being done (we have 700 open PRs on sklearn) [SI83]”).

**$I2$  Wasted review effort.** Review effort spent on abandoned pull requests is also a concern for integrators. In an OSS project, the number of integrators is usually small [41]. Integrators of popular projects can be very busy in handling numerous tasks, e.g., triaging issues and answering questions. In addition, code review is a complicated process that requires integrators to invest significant effort and time [12].



If a pull request is finally abandoned by the author, integrators can feel disappointed with the wasted effort (e.g., “lost review time that could have been invested in reviewing other PRs and/or writing my own patches” [S147])

**I3 Additional attention for more careful close.** For integrators, it is not always safe to immediately close inactive pull requests because contributors may only be temporarily unresponsive and may return later (e.g., “Sometimes a PR leads to a discussion that reveals significant further development is needed, and not yet done.” [S111]). Mistakenly closing pull requests can hurt contributors’ feelings and make them demotivated to provide additional updates. Therefore, integrators usually add special labels (e.g., “stale” and “needs feedback”) or used the “@” mentions [38] to first remind contributors of their inactive pull requests. Only if integrators are quite sure that the contributor is no longer working on the pull request can they safely close it.

**I4 Delayed landing of inter-dependent pull requests.** Pull requests submitted by different developers might have direct or indirect relationships due to the technical dependency between modules or projects [44]. Consequently, the landing of a pull request might depend on the successful merging of another pull request. If a pull request is abandoned, the progress of the pull request that depends on it is affected (e.g., “Perhaps I’ll prepare a PR for that, since it seems #33036 was abandoned” [D76]).

**I5 Duplicate pull requests.** This impact mostly concerns redundancy with submitting duplicate pull requests [42]. If a pull request is abandoned, the associated issue is left unresolved and can be encountered by other developers. Other contributors might create duplicate pull requests from scratch, which could be avoided if the original pull request was not abandoned (e.g., “They cause well-meaning contributors to open up additional PRs” [S112]).

**I6 Disordered milestone.** OSS projects might use milestones [23] to track progress on developers’ work. All the issues and pull requests associated with a milestone are expected to be completed before a branch for the upcoming version is released. If a pull request is abandoned, integrators have to remove it from the scheduled plan, and postpone it in the next release (e.g., “Think we missed the boat on 1.7 though :/” [D71]).

**I7 Bad impression.** OSS collaborative development is a social process where developers form impressions about others based on a history of activities [13], [46]. Therefore, in the survey for integrators, we additionally asked the participants a question about whether they would have a bad impression of abandoners. 36.7% of them answered “No”, and one mentioned, “Especially if it’s a community driven by voluntary efforts, the author typically gave an honest attempt and abandoned their PR due to lack of time or a slow response from reviewers” [S147]. While 10.2% of integrators answered “Yes” and the others suggested that it depends on the situations, e.g., *i)* the frequency at which this happens, *ii)* the reason for the abandonment, *iii)* where the discussion lead, and *iv)* what the pull request is about.







**RQ2:** Pull request abandonment increases the effort in project

management and maintenance. The most frequently mentioned negative impacts are cluttering the pull request list, wasting review effort and costing additional attention. Other impacts include blocking inter-dependent tasks, causing repeated patches, and disordering the scheduled milestone. Moreover, over 63% of integrators suggested that they would have a bad impression of abandoners directly or indirectly, depending on the corresponding situation.

### 4.3 RQ3: How do integrators cope with abandoned pull requests?

Table 5 lists the strategies project integrators adopted to cope with abandoned pull requests. All the topics were identified by manual observation. The table also presents the frequency of strategies collected from the full-scale survey with integrators. The remainder of this section analyzes each of those strategies in detail.

TABLE 5: Strategies used to cope with abandoned pull requests.

Strategy	Votes (%)
S1 Assign a successor to take it over	66.7 
S2 Close it to clean pull request list	60.2 
S3 Picked up by volunteers spontaneously	42.3 
S4 Advertise it to the community	37.2 
S5 Choose the duplicate as an alternative	23.1 
S6 Merge as it is and iterate over it	12.8 

**S1 Assign a successor to take it over.** Even if a pull request has been abandoned by its submitter, the problem it attempted to solve might still exist. In addition, if a pull request was not rejected at the beginning but was instead requested for improvement, this means that the pull request had a certain value and integrators were inclined to merge it. Therefore, we can notice that 66.7% of integrators assigned successors to take over abandoned pull requests. We took a closer look and asked integrators about their ways of assigning successors. Most integrators completed the abandoned pull requests themselves (e.g., “We adopt the pull request and make the requested changes ourselves,” [S174]), especially when the feature/bugfix was urgent or critical. Integrators also appointed (i.e., using “@” mentions) a successor from the reviewers or developers whose ownership or expertise was related to the pull request (e.g., “Maybe @[developer] can pick this up?” [D72]). Abandoned pull requests could be discussed in an offline internal meeting and certain developers were assigned to own them. A few integrators reported that originators already entrusted another developer to continue the pull request; therefore, integrators did not need to assign another successor.

**S2 Close it to clean pull request list.** 60.2% of respondents said that they had to close the abandoned pull requests to keep the pull request list clean. In our observation, we found that integrators usually closed them in a friendly way and allowed the original authors to reopen the pull requests as they wanted (e.g., “There was no update for a long while, closing this therefore. @[author] please just reopen if you want to continue working on it” [D79]). A few integrators also reported that bots [72] were used to automatically close

inactive pull requests according to predefined rules (e.g., “Our stale bot closed it” [SI68]).

**S3 Picked up by volunteers spontaneously.** Integrators mentioned that some contributors spontaneously picked up abandoned pull requests. To better understand contributors’ willingness, in the full-scale survey for contributors, we asked the participants if they want to pick up pull requests abandoned by others. The responses are shown in Figure 4. Interestingly, only 7.1% of contributors answered “No”; most of them are willing to pick up abandoned pull requests. Among the respondents who expressed willingness, 8.3% of contributors answered “Yes”, while the others suggested that willingness depends on circumstances, such as issue importance and time investment. This finding might also explain why few integrators thought pull request abandonment did not produce serious problems (e.g., “It’s not great, but it’s not THAT big of a problem” [SI77]).

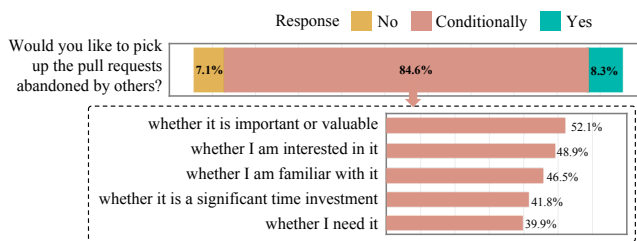


Fig. 4: Contributors’ willingness.

**S4 Advertise it to the community.** If an abandoned pull request is not important enough for integrators to invest their time in finishing it by themselves or seeking a particular successor, they advertise it and expect a volunteer to pick it up. We examined advertising behaviors and found two common ways integrators used to advertise abandoned pull requests. The first approach is *leaving an advertising comment*, i.e., integrators leave a comment on the tracking page of the abandoned pull request or that of the associated issue, stating that everyone is welcome to pick up the pull request (e.g., “That’s unfortunate. This was a good change, just needed to make sure the details were correct. Hopefully someone will pick this up again” [D73]). Another approach is *adding a special label*, i.e., integrators add special labels [35] (e.g., “help wanted”) to mark the abandoned pull requests.

**S5 Choose the duplicate as an alternative.** It is possible that the same issue is encountered by different developers who might unintentionally submit duplicate pull requests [42], [79]. If one contributor is unresponsive to integrators’ comments, integrators can resort to the duplicate pull request, if any (e.g., “Closing due to inactivity and because similar improvements were made to this test in another PR that has landed” [D82]).

**S6 Merge as it is and iterate over it.** In some cases, integrators preferred to merge an abandoned pull request as it was and iterate over it (e.g., “@[integrator] can we merge this as is, and open another issue with the other test?” [D78]). Generally, this kind of pull request passed all the tests and did not contain any serious bugs. The problem was usually that it only solved part of the real issue and more work was needed. Therefore, merging such pull requests would not break the runnable state of the codebase.

**RQ3:** Integrators take different actions to deal with abandoned pull requests, among which assigning successors to take them over and closing them are the most common strategies. Integrators also advertised them to the community, merged them with additional amending, and chose the duplicates. Additionally, some abandoned pull requests were spontaneously picked up by volunteer contributors.

## 5 DISCUSSION

Based on our findings, we provide additional discussion and propose actionable suggestions for OSS practitioners and tool builders.

### 5.1 Main findings

#### 5.1.1 Playing hooky vs. dropping out

We contrasted our findings about pull request abandonment (regarded as *playing hooky*) with earlier findings about project abandonment (regarded as *dropping out*). The first observation we can make is that the reasons that result in pull request abandonment overlap with the reasons that lead developers to quit a project. For example, lack of time, the second most common reason for pull request abandonment ( $\mathcal{R}1$  *Lack of time*), was generally the most cited reason when developers described why they left a project [33], [41], [47]. Other overlapping reasons include reception issues [33], [41], [66], technical hurdles [41], [47], and lack of interest [33], [47]. The overlap of reasons may be an indication that occasional pull request abandonment might be a good prediction of permanent disengagement. Given this implication, project integrators can make more informed decisions with respect to team management. If the abandoner is a newcomer, integrators can concentrate their time and effort on offering active support for others who are more likely to remain in the project since integrators’ time has already been scarce [25]. If the abandoner is a well-established developer, integrators should be cautious that they are going to lose the developer. They need to take actions to prevent the developer’s disengagement, given the fact that turnover of well-established developers would result in significant loss of knowledge and productivity [20], [36].

Despite the overlapping reasons, some reasons leading developers to drop out are not present in the taxonomy of reasons for pull request abandonment. These reasons are mainly related to the developer (e.g., changing to a new job that does not support OSS [47]) or the project (e.g., change of governance model [33]). One possible explanation is that although a developer cannot submit more pull requests to a project due to personal or project-related reasons, s/he would finish the tasks at hand before dropout. Otherwise, the abandoned pull request might hurt the developer’s reputation [37], [75], since even though a developer leaves a project, the developer’s behaviors in the project have been tracked by GitHub and are publicly visible to any GitHub user [18], [46]. Future research is necessary to validate this assumption and examine developers’ perception of pull request abandonment from contributors’ perspective.

We also observed that several reasons for pull request abandonment were not found to have ever led developers to discontinue contribution. In general, these reasons were mainly related to the pull request *per se* (e.g., *R1 inadequate community demand*) or the review process (e.g., *R8 integrators do not reach a consensus*). On the one hand, we assume that the micro-level experience concerning one particular pull request does not necessarily impact developers' macro-level willingness to remain in a project. Since many developers contributed to a project to scratch their personal itches [52], if developers who ever abandoned pull requests encounter new issues in the same project, they might continue to submit pull requests to resolve the issues. On the other hand, we speculate that developers might leave a project due to the problems experienced from several individual contributions. For example, a developer might lose interest in OSS and leave a project, after submitting several pull requests that did not receive adequate favor from the community. When the developer was asked for her/his disengagement from the project, s/he, however, might describe the reason from a broad social perspective (i.e., lost interest in OSS) instead of a specific technical perspective (i.e., inadequate community demand for the pull request). The extent to which pull request abandonment affects developers' willingness to contribute is a topic that deserves further investigation.

### 5.1.2 Responsiveness

Lack of responsiveness from integrators is found to be the main reason for pull request abandonment (*R1 Lack of answer from integrators; R4 Not treated seriously by integrators*). As one of our survey respondents suggested "I think there is a golden window when a volunteer and owner are willing to work on a feature together. If there is too sparse of a community or either take too long to respond then likely they have both moved on to other things and don't even remember what the pr was" [SC182]. One of the main reasons for integrators' unresponsiveness is that they are overwhelmed by a huge number of tasks [25], such as answering questions, discussing issues, and reviewing pull requests. However, the outcomes and effects of integrators' delay on different tasks might be different. Since the time and energy of integrators is limited (i.e., response latency is sometimes inevitable), an effective strategy or mechanism for integrators should be considered to balance their workload and schedule different kinds of tasks using different priorities.

Another possible factor that affects integrators' responsiveness is the complex processes projects use for quality assurance. The bureaucracy [65] and overly long CI build [31] in code review might hinder a pull request from being processed as quickly and smoothly as possible. Considering the nature of OSS, integrators of projects with high formality should simplify the participation process in the project and gain a tradeoff between formalization and smoothness of the process.

Furthermore, integrators' responsiveness might be affected by the incoordination among contributors and integrators. For example, OSS developers who are geographically distributed across different time zones [27] tend to face the problem of work-hour asynchronicity. The stretched communication delay [76] might make them seem slow and

inattentive in responding to others. The effect of timezone dispersion on coordination efficiency deserves further analysis.

### 5.1.3 Responsibility

To get their pull requests merged, contributors are expected to follow projects' standards. Nevertheless, we have found that contributors do not always take responsibility for fixing up their pull requests to suit integrators' taste (*R10 It requires more effort than anticipated; R12 Tedious process of updating the pull request*), especially in maintaining the adherence to coding styles and project conventions. This finding aligns with prior studies which found that a part of pull requests were closed due to format issues and incompleteness [24], [83].

Contributors are also expected to clearly state their abandonment decision in a timely manner. Otherwise, it will cost integrators additional effort to keep track of pull requests and keeping the pull request list from growing too long (*I1 Cluttered the pull request list; I3 Additional attention for more careful close*). However, in the full-scale survey with contributors, when we asked the participants to tell us their actions when they were unable to finish their pull requests, more than 65% of them said they just left the pull requests. Meanwhile, a number of contributors vanished because they felt ignored by integrators (*R4 Not treated seriously by integrators*). Specifically, our manual observation found several cases in which integrators' unmindful feedback (e.g., "Maybe I read the code wrong. Just be sure to add a test") drove the contributor away. It might be unreasonable to expect contributors to behave cooperatively when they have unpleasant feelings. Thus, it is very important for OSS communities to mitigate unintentional conflicts and improve the sense of mutual responsibility between external contributors and integrators.

### 5.1.4 Handover

Given the fact that integrators used various ways to make abandoned pull requests handed over to other developers (*S1 Assign a successor to take it over; S4 Advertise it to the community*), and sometimes they even accepted an 'imperfect' patch followed by additional amending work (*S6 Merge as it is and iterate over it*), integrators tried to accept each pull request that could be merged, if possible. However, more than 60% of integrators also said that they had to close the abandoned pull requests to clean the pull request list (*S2 Close it to clean pull request list*). This highlights the need for and feasibility of establishing a practical handover mechanism since most contributors reported that they were willing to pick up pull requests abandoned by others (*S3 Picked up by volunteers spontaneously*). Such a mechanism can efficiently facilitate the completion of abandoned pull requests if they can be properly recommended to the potential successors.

Furthermore, picking up someone else's work might be not easy enough. Even if the original authors cannot continue for various reasons, their knowledge and experience with the the abandoned pull requests can still benefit the follow-up pull requests submitted by the successors. In the full-scale survey with contributors, we asked originators about what kind of help they would like to offer. As one originator said, "I am open to being as involved or out of it as needed/helpful" [SC527]. We found that most originators were willing to provide further assistance by explaining their

ideas more clearly or reviewing the follow-up pull requests (e.g., “at least I believe i would always review the new changes” [SC446]). This indicates that integrators can still invite the originators to pay attention to the follow-up pull requests.

## 5.2 Suggestions for OSS practitioners

We propose a set of suggestions that can help both OSS project integrators and contributors better collaborate in the pull request mechanism.

**Progress awareness.** Even if integrators are too busy to review each pull request in a timely manner, it is helpful to leave a welcome message and manually or rely on a bot [72] to provide an estimated time for review. When the estimated waiting time has exceeded (or is approaching), the bot can comfort the contributors and remind integrators to pay attention to the pull request in the meantime. This can make clear integrators’ status and contributors will not mistakenly feel ignored by integrators ( $\mathcal{R}4$  *Not treated seriously by integrators*).

**Maximizing automation.** Integrators should introduce as much automation support as they can in quality assurance to reduce their own effort and that of contributors. For example, projects that do not provide automatic tools for convention assurance can establish their own checking specifications, which can be used by tools like CheckStyle [1] to locally and automatically check code standards before patch submission. This helps integrators avoid requesting cosmetic and nitpicking changes and frees up the time of casual contributors who might have limited patience to read the standards ( $\mathcal{R}2$  *Lack of time*;  $\mathcal{R}12$  *Tedious process of updating the pull request*).

**Conspicuous policy.** Integrators should establish a policy on their treatments of contributors’ undesired activities. For instance, they can specify the inactive period until a reminder is sent and the inactive period until a pull request is closed. Projects that already have such a policy should notify contributors of the policy in a timely manner, e.g., automatically reminding contributors at pull request submission. With a predefined policy and timely reminder, contributors are less likely to complain or feel hurt by integrators’ treatments; thus, they can collaborate in a less stressful way ( $\mathcal{I}3$  *Additional attention for more careful close*).

**Controlled discussion.** Integrators can set the maximum time needed to reach consensus on the change direction. If a discussion has lasted for a long period that exceeds the threshold, a voting process involving interested stakeholders should be triggered to end the discussion to keep it from bikeshedding [50]. This might help to avoid abandonment caused by  $\mathcal{R}8$  *Integrators do not reach a consensus*.

**Clear declaration.** On the one hand, contributors should clearly tell others their abandonment decisions so that integrators can stop expecting a response from them ( $\mathcal{I}1$  *Clutter the pull request list*;  $\mathcal{I}3$  *Additional attention on more careful close*). If they intend to leave temporarily, it is better to state it and give an estimated period of inactivity. On the other hand, integrators should clearly mark abandoned pull requests, e.g., using special labels, and make them easy for candidate successors to find ( $\mathcal{S}3$  *Picked up by volunteers spontaneously*).

**Nit-picking when landing.** In some projects, the integrators merge pull requests locally using git commands instead of

using the merge button on GitHub. For these projects, the additional small changes (e.g., typos and squashing) that need to be made before a pull request gets merged can be made by integrators when landing the commits. This seems more efficient than requesting and waiting for the pull request author to perform such trivial changes, which might also help to avoid stalling pull requests ( $\mathcal{R}12$  *Tedious process of updating the pull request*).

**Review guidelines.** Projects that do not provide guidelines for new collaborators should at least list some of the best review practices. For example, they can suggest what kind of review comments are considered helpful and insightful ( $\mathcal{R}7$  *Lack of knowledge about the requested changes*), and the way in which a large change or multiple changes should be requested ( $\mathcal{R}10$  *It requires more effort than anticipated*).

**Visible communication channels.** Some projects have multiple channels for communication, e.g., pull request comment, mailing list [28], IRC [60], and Slack [3]. Communication channels outside GitHub should be publicly and apparently described on the project homepage. In particular, contributors should be informed which channel should be used to get quick replies when they have not heard anything for a long time ( $\mathcal{R}1$  *Lack of answer from integrators*).

## 5.3 Implications for tool design

Our findings suggest several implications for tool builders to create new mechanisms and tools and provide developers with more automated and intelligent support.

**Enhanced prioritization.** A prior study [69] proposed a system to help integrators prioritize their work. The features used include the size of change, pull request age, and contributor track record. We argue that developers’ time zones and the elapsed time waiting for integrators’ response in a discussion thread should be applied as prioritization criteria. The waiting time can be calculated from the time when contributors push new commits or post a new comment. This helps integrators focus on at-risk pull requests in a timely manner ( $\mathcal{R}1$  *Lack of answer from integrators*;  $\mathcal{R}4$  *Not treated seriously by integrators*).

**Motivating badges.** To better motivate contributors who have been asked to carry out complicated change requests, integrators can be allowed to assign them a special badge [61] upon pull request acceptance indicating they have ever worked on tough tasks ( $\mathcal{R}6$  *Lack of interest*;  $\mathcal{R}10$  *It requires more effort than anticipated*). This may act as an incentive for contributors to continue in order to build their reputation and gain peer recognition [46], [61].

**Online update.** Platforms can support the updating of pull requests online. For instance, whenever new codes are merged/pushed and no merge conflict is detected, a pull request can be automatically rebased onto the latest base branch. Moreover, developers can be allowed to edit pull requests online. This makes it more cost effective to carry out minor changes that do not require significant effort, e.g., fixing a typo ( $\mathcal{R}12$  *Tedious process of updating the pull request*).

**Premium notification.** Currently, GitHub users receive notification emails triggered by updates on the conversations in which they are participating. The notifications can be expedited by enabling the ping tool to send specific kinds of

reminders. For example, contributors can send a “*Request-confirm*” reminder to the corresponding integrator, who is expected to quickly confirm the change detail. The reminder label is added to the head of the email subject, which can distinguish the urgent messages from notification floods and therefore increase their chance of being noticed by integrators (*R1 Lack of answer from integrators*).

**Successor recommendation.** A tool can automatically recommend a list of candidate successors for abandoned pull requests (*S1 Assign a successor to take it over; S3 Picked up by volunteers spontaneously; S4 Advertise it to the community*). The candidates can be grouped together in a visualization panel displaying their profiles. Invitation messages would be automatically sent to the candidates who have been selected by integrators. Moreover, if integrators label the needed changes as trivial, the tool can recommend newcomers to try them.

## 6 THREATS TO VALIDITY

In this section, we discuss our potential threats to validity as follows.

**Construct validity.** The first threat is related to the risk of survey respondents misunderstanding the survey questions. To mitigate this threat, we discussed the questions with experienced researchers and made necessary clarifications of the wording of questions and answers. We also conducted the survey in two rounds; the first pilot survey results were used to improve the design of the final full-scale survey. Second, as with any survey method, to control for sampling bias can be challenging. One threat is selection bias (*i.e.*, developers who do not answer the survey may hold different opinions). To encourage responses, we designed our surveys to be as short as possible and made the questions easy to answer, adhering to survey design recommendations [26], [53]. Another threat is that some of the respondents in the full-scale survey with contributors might have no experience with pull request abandonment and might provide speculative answers. To mitigate this threat, we targeted developers who had pull requests closed recently; they had higher chances of having ever abandoned pull requests than a random sample of GitHub developers. Moreover, at the introduction of the survey questionnaires, we reminded survey participants to answer the questions according to their real experience.

**Internal validity.** The first threat concerns the construction of the taxonomies denoting the reasons, impacts, and coping strategies for pull request abandonment. We may have drawn wrong conclusions in card sorting because the coders may have had preconceptions in interpreting text. To minimize subjectivity and personal bias, the sorting tasks were performed by three of the authors together. Second, as with all qualitative studies, a threat exists concerning the completeness of our taxonomies. We may have missed some abandoned pull requests in the manual observation. To mitigate this threat, we covered the most common cases and included as many keywords as possible for automatic identification. To further minimize the risk of an incomplete taxonomy, we coded all the remaining cards when saturation was reached in card sorting. We also performed a second pass over all cards to ensure that we did not miss any

important information, and invited all the authors to discuss the taxonomies. Additionally, we validated and augmented the taxonomy developed in manual observation with survey responses.

**External validity.** Threats to external validity are related to the generalizability of our conclusions. Our study was conducted on a set of GitHub projects. Although we have used random sampling and the studied projects are with high diversity in terms of programming language and application domain, those projects may not be representative of all OSS projects. Thus, our results may not generalize beyond the OSS projects involved in this study. Additional replication studies on more projects and developers inside and outside of GitHub are needed to generalize our results.

## 7 CONCLUSION

For OSS projects to survive and grow, it is important that project integrators and external contributors work closely to ensure that submitted pull requests are accepted. However, asking contributors to address integrators’ change requests does not always get a response. Through manual observation and surveys with OSS developers, we uncovered 12 main reasons for pull request abandonment, among which the top-ranked ones are lack of responsiveness from integrators and lack of time from contributors. We also identified the specific impacts of pull request abandonment, with the most frequently cited ones being cluttering pull request list, wasting review effort and costing additional attention. Finally, we revealed the common coping strategies used by integrators, such as assigning successors and advertising to the community.

Our findings have several implications for OSS practitioners and researchers. First, mechanisms and tools can be designed to prevent avoidable abandonment, mitigate undesirable impacts of abandonment, and recommend appropriate successors to take over abandoned pull requests. Second, researchers can study further on pull request abandonment from the integrators’ side, *i.e.*, why project integrators ignore pull requests without leaving a comment. Last but not least, a useful complement to this study would be to investigate how project characteristics (*e.g.*, project popularity and formality) affect pull request abandonment in a comprehensive way.

## ACKNOWLEDGMENTS

This work was supported by National Grand R&D Plan (Grant No. 2020AAA0103504) and National Natural Science Foundation of China (Grant No. 61702534).

## REFERENCES

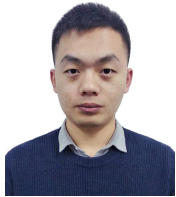
- [1] About checkstyle. <https://github.com/checkstyle/checkstyle>. Accessed: 2020-07-06.
- [2] About github api. <http://developer.github.com/v3/>. Accessed: 2020-07-06.
- [3] About slack. <https://www.slack.com/>. Accessed: 2020-12-06.
- [4] About surveymonkey. <https://www.surveymonkey.com/mp/aboutus>. Accessed: 2020-07-06.
- [5] The list of 20 less-popular projects. [https://github.com/whystar/TSE2021-AbandonedPR/blob/master/resource/observation/20\\_less\\_popular\\_projects.md](https://github.com/whystar/TSE2021-AbandonedPR/blob/master/resource/observation/20_less_popular_projects.md). Accessed: 2021-01-19.
- [6] Surveys. <https://github.com/whystar/TSE2021-AbandonedPR/tree/master/resource/surveys>. Accessed: 2021-01-19.

- [7] Alberto Bacchelli and Christian Bird. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 35th International Conference on Software Engineering*, pages 712–721. IEEE Press, 2013.
- [8] Andrew Begel and Thomas Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*, pages 12–23, 2014.
- [9] Andrea Bonaccorsi and Cristina Rossi. Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology & Policy*, 18(4):40–64, 2006.
- [10] Hudson Borges, Andre Hora, and Marco Tulio Valente. Understanding the factors that impact the popularity of github repositories. In *Proceedings of the 2016 IEEE International Conference on Software Maintenance and Evolution*, pages 334–344. IEEE, 2016.
- [11] Jelmer P Borst, Niels A Taatgen, and Hedderik van Rijn. What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2971–2980. ACM, 2015.
- [12] Amiangshu Bosu, Jeffrey C Carver, Christian Bird, Jonathan Orbeck, and Christopher Chockley. Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *IEEE Transactions on Software Engineering*, 43(1):56–75, 2016.
- [13] Amiangshu Bosu, Jeffrey C. Carver, Christian Bird, Jonathan Orbeck, and Christopher Chockley. Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *IEEE Transactions on Software Engineering*, 43(1):56–75, 2017.
- [14] J. L. Campbell, C. Quincy, J. Osserman, and O. K. Pedersen. Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. *Sociological Methods & Research*, 42(3):294–320, 2013.
- [15] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 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*, pages 1–11, 2012.
- [16] Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, 10(2), 2005.
- [17] Kevin Crowston, Kangning Wei, Qing Li, and James Howison. Core and periphery in free/libre and open source software team communications. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06)*, volume 6, pages 118a–118a. IEEE, 2006.
- [18] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*, pages 1277–1286, 2012.
- [19] Fabian Fagerholm, Alejandro S Guinea, Jürgen Münch, and Jay Borenstein. The role of mentoring and project characteristics for onboarding in open source software projects. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–10, 2014.
- [20] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C Murphy, and Jean-Rémy Falleri. Impact of developer turnover on quality in open-source software. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 829–841, 2015.
- [21] Jonas Gamalielsson and Bjoern Lundell. Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved? *Journal of Systems & Software*, 89(MAR.):128–145, 2014.
- [22] Georgios Gousios. The ghtorrent dataset and tool suite. In *Proceedings of the 10th working conference on mining software repositories*, pages 233–236. IEEE Press, 2013.
- [23] Georgios Gousios, Martin Pinzger, and Arie Van Deursen. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, pages 345–355, 2014.
- [24] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. Work practices and challenges in pull-based development: the contributor’s perspective. In *Proceedings of the 38th International Conference on Software Engineering*, pages 285–296. IEEE, 2016.
- [25] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. Work practices and challenges in pull-based development: the integrator’s perspective. In *Proceedings of the 37th International Conference on Software Engineering*, pages 358–368. IEEE, 2015.
- [26] Robert M Groves, Floyd J Fowler Jr, Mick P Couper, James M Lepkowski, Eleanor Singer, and Roger Tourangeau. *Survey methodology*, volume 561. John Wiley & Sons, 2011.
- [27] Dorina C Gumm. Distribution dimensions in software development projects: A taxonomy. *IEEE software*, 23(5):45–51, 2006.
- [28] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie Van Deursen. Communication in open source software development mailing lists. In *Proceedings of 2013 10th IEEE Working Conference on Mining Software Repositories*, 2013.
- [29] Christoph Hannebauer, Matthias Book, and Volker Gruhn. An exploratory study of contribution barriers experienced by newcomers to open source software projects. In *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering*, pages 11–14, 2014.
- [30] Alexander Hars and Shaosong Qu. Working for free? motivations for participating in open-source projects. *International journal of electronic commerce*, 6(3):25–39, 2002.
- [31] Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-offs in continuous integration: assurance, security, and flexibility. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017.
- [32] Wenjian Huang, Tun Lu, Haiyi Zhu, Guo Li, and Ning Gu. Effectiveness of conflict management strategies in peer review process of online collaboration projects. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, pages 717–728. ACM, 2016.
- [33] Giuseppe Iaffaldano, Igor Steinmacher, Fabio Calefato, Marco Gerosa, and Filippo Lanubile. Why do developers take breaks from contributing to oss projects?: a preliminary analysis. In *Proceedings of the 2nd International Workshop on Software Health*, pages 9–16. IEEE Press, 2019.
- [34] Rahul N Iyer, S Alex Yun, Meiyappan Nagappan, and Jesse Hoey. Effects of personality traits on pull request acceptance. *IEEE Transactions on Software Engineering*, 2019.
- [35] Javier Luis Cánovas Izquierdo, Valerio Cosentino, Belén Rolandi, Alexandre Bergel, and Jordi Cabot. Gila: Github label analyzer. In *Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 479–483. IEEE, 2015.
- [36] Daniel Izquierdo-Cortazar, Gregorio Robles, Felipe Ortega, and Jesus M Gonzalez-Barahona. Using software archaeology to measure knowledge loss in software projects due to developer turnover. In *Proceedings of the 2009 42nd Hawaii International Conference on System Sciences*, pages 1–10. IEEE, 2009.
- [37] Corey Jergensen, Anita Sarma, and Patrick Wagstrom. The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, pages 70–80. ACM, 2011.
- [38] David Kavalier, Premkumar Devanbu, and Vladimir Filkov. Whom are you going to call? determinants of @-mentions in github discussions. *Empirical Software Engineering*, (1):1–29.
- [39] Oleksii Kononenko, Tresa Rose, Olga Baysal, Michael Godfrey, Dennis Theisen, and Bart De Water. Studying pull request merges: a case study of shopify’s active merchant. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 124–133, 2018.
- [40] Karim R Lakhani and Robert G Wolf. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. 2003.
- [41] Amanda Lee, Jeffrey C Carver, and Amiangshu Bosu. Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: a survey. In *Proceedings of the 39th International Conference on Software Engineering*, pages 187–197. IEEE Press, 2017.
- [42] Zhixing Li, Yue Yu, Minghui Zhou, Tao Wang, Gang Yin, Long Lan, and Huaimin Wang. Redundancy, context, and preference: An empirical study of duplicate pull requests in oss projects. *IEEE Transactions on Software Engineering*, 2020.
- [43] Bin Lin, Gregorio Robles, and Alexander Serebrenik. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In *Proceedings of the 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, pages 66–75. IEEE, 2017.
- [44] Wanwangying Ma, Lin Chen, Xiangyu Zhang, Yuming Zhou, and Baowen Xu. How do developers fix cross-project correlated bugs?

- a case study on the github scientific python ecosystem. In *Proceedings of the 39th International Conference on Software Engineering*, pages 381–392, 2017.
- [45] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. An empirical study of practitioners’ perspectives on green software engineering. In *Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering*, pages 237–248. IEEE, 2016.
- [46] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pages 117–128. ACM, 2013.
- [47] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. Why do people give up flossing? a study of contributor disengagement in open source. In *Proceedings of the 2019 International Conference on Open Source Systems*, pages 116–129. Springer, 2019.
- [48] Cassandra Overney, Jens Meinicke, Christian Kästner, and Bogdan Vasilescu. How to not get rich: An empirical study of donations in open source. In *Proceedings of the 2020 International Conference on Software Engineering*, ICSE. ACM, 2020.
- [49] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. Recommending and localizing change requests for mobile apps based on user reviews. In *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering*, pages 106–117. IEEE, 2017.
- [50] Cyril Northcote Parkinson. *Parkinson’s law: The pursuit of progress*. Readers Union [in association with] John Murray, 1959.
- [51] Gustavo Pinto, Luiz Felipe Dias, and Igor Steinmacher. Who gets a patch accepted first? comparing the contributions of employees and volunteers. In *Proceedings of 2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 110–113. IEEE, 2018.
- [52] Gustavo Pinto, Igor Steinmacher, and Marco Aurélio Gerosa. More common than you think: An in-depth study of casual contributors. In *Proceedings of the 23rd International Conference on Software Analysis, Evolution, and Reengineering*, volume 1, pages 112–123. IEEE, 2016.
- [53] Teade Punter, Marcus Ciolkowski, Bernd Freimut, and Isabel John. Conducting on-line surveys in software engineering. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, pages 80–88. IEEE, 2003.
- [54] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. Going farther together: The impact of social capital on sustained participation in open source. In *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 688–699. IEEE, 2019.
- [55] Ayushi Rastogi. Do biases related to geographical location influence work-related decisions in github? In *Proceedings of 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 665–667. IEEE, 2016.
- [56] Ayushi Rastogi, Nachiappan Nagappan, Georgios Gousios, and André van der Hoek. Relationship between geographical location and evaluation of developer contributions in github. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–8, 2018.
- [57] Eric Raymond. The cathedral and the bazaar. *Knowledge Technology & Policy*, 12(3):23–49, 1999.
- [58] Jeffrey A Roberts, Il-Horn Hann, and Sandra A Slaughter. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects. *Management science*, 52(7):984–999, 2006.
- [59] Andreas Schilling, Sven Laumer, and Tim Weitzel. Who will remain? an evaluation of actual person-job and person-team fit to predict developer retention in floss projects. In *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, pages 3446–3455. IEEE, 2012.
- [60] Emad Shihab, Zhen Ming Jiang, and Ahmed E Hassan. Studying the use of developer irc meetings in open source projects. In *2009 IEEE International Conference on Software Maintenance*, pages 147–156. IEEE, 2009.
- [61] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 103–116, 2013.
- [62] Donna Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.
- [63] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer Supported Cooperative Work & Social Computing*, pages 1379–1392. ACM, 2015.
- [64] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. Overcoming open source project entry barriers with a portal for newcomers. In *Proceedings of the 38th International Conference on Software Engineering*, pages 273–284. ACM, 2016.
- [65] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco Aurélio Gerosa. Almost there: A study on quasi-contributors in open-source software projects. In *Proceedings of the 40th International Conference on Software Engineering*, pages 256–266. IEEE, 2018.
- [66] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurélio Gerosa. Why do newcomers abandon open source software projects? In *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 25–32. IEEE, 2013.
- [67] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rinear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*, 3:e111, 2017.
- [68] Jason Tsay, Laura Dabbish, and James Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *Proceedings of the 36th International Conference on Software Engineering*, pages 356–366. ACM, 2014.
- [69] Erik Van Der Veen, Georgios Gousios, and Andy Zaidman. Automatically prioritizing pull requests. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 357–361. IEEE, 2015.
- [70] Bogdan Vasilescu, Kelly Blincoe, Qi Xuan, Casey Casalnuovo, Daniela Damian, Premkumar Devanbu, and Vladimir Filkov. The sky is not the limit: multitasking across github projects. In *Proceedings of the 38th International Conference on Software Engineering*, pages 994–1005. IEEE, 2016.
- [71] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816, 2015.
- [72] Mairieli Wessel, Bruno Mendes De Souza, Igor Steinmacher, Igor S Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A Gerosa. The power of bots: Characterizing and understanding bots in oss projects. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–19, 2018.
- [73] Marty J Wolf, Kevin Bowyer, Don Gotterbarn, and Keith Miller. Open source software: intellectual challenges to the status quo. *ACM SIGCSE Bulletin*, 34(1):317–318, 2002.
- [74] Vincent Wolff-Marting, Christoph Hannebauer, and Volker Gruhn. Patterns for tearing down contribution barriers to floss projects. In *Proceedings of the 2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, pages 9–14. IEEE, 2013.
- [75] Yunwen Ye and Kouichi Kishida. Toward an understanding of the motivation open source software developers. In *Proceedings of the 25th International Conference on Software Engineering*, pages 419–429. IEEE Computer Society, 2003.
- [76] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. Wait for it: Determinants of pull request evaluation latency on github. In *2015 IEEE/ACM 12th working conference on mining software repositories*, pages 367–371. IEEE, 2015.
- [77] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. Reviewer recommendation for pull-requests in github. *Information & Software Technology*, 74(C):204–218, 2016.
- [78] Minghui Zhou and Audris Mockus. What make long term contributors: Willingness and opportunity in oss community. In *Proceedings of the 34th International Conference on Software Engineering*, pages 518–528. IEEE Press, 2012.
- [79] Shurui Zhou, Bogdan Vasilescu, and Christian Kästner. What the fork: a study of inefficient and efficient forking practices in social coding. In *Proceedings of the 27th ACM Joint Meeting on European*

*Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 350–361. ACM, 2019.

- [80] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. Effectiveness of code contribution: From patch-based to pull-request-based tools. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 871–882. ACM, 2016.
- [81] Thomas Zimmermann. Card-sorting: From text to themes. In *Perspectives on Data Science for Software Engineering*, pages 137–141. Elsevier, 2016.
- [82] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach D Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, 2019.
- [83] Weiqin Zou, Jifeng Xuan, Xiaoyuan Xie, Zhenyu Chen, and Baowen Xu. How does code style inconsistency affect pull request integration? an exploratory study on 117 github projects. *Empirical Software Engineering*, 24(6):3871–3903, 2019.



**Zhixing Li** is a Ph.D. candidate in Software Engineering at National University of Defense Technology (NUDT). He received his Master degree in compute science from NUDT in 2017. His research goals are centered around the idea of making the open source collaboration more efficient and effective by investigating the challenges faced by open source communities and designing smarter collaboration mechanisms and tools.



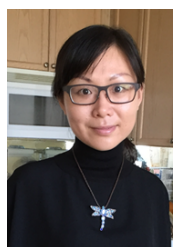
**Yue Yu** is an associate professor in the College of Computer at National University of Defense Technology (NUDT). He received his Ph.D. degree in Computer Science from NUDT in 2016. He has won Outstanding Ph.D. Thesis Award from Hunan Province. His research findings have been published on ICSE, FSE, ASE, TSE, MSR, IST, ICSME, ICDM and ESEM. His current research interests include software engineering, data mining and computer-supported cooperative work.



**Tao Wang** is an associate professor in the College of Computer at National University of Defense Technology (NUDT). He received his Ph.D. degree in Computer Science from NUDT in 2015. His work interests include open source software engineering, machine learning, data mining, and knowledge discovering in open source software.



**Gang Yin** is an associate professor in the College of Computer at National University of Defense Technology (NUDT). He received his Ph.D. degree in Computer Science from NUDT in 2006. He has published more than 60 research papers in international conferences and journals. His current research interests include distributed computing, information security, software engineering, and machine learning.



**ShanShan Li** is a professor in the Department of Computer Science of National University of Defense Technology (NUDT). She received her Ph.D. degree from NUDT in 2007. Her main research areas includes empirical software engineering and intelligent software development. She has published more than 70 papers on FSE, ASE, ICSE, ICPC, SANERD, TPDS et al.



**Huaimin Wang** received his Ph.D. in Computer Science from National University of Defense Technology (NUDT) in 1992. He has been awarded the “Chang Jiang Scholars Program” professor and the Distinct Young Scholar, *etc.* He has published more than 100 research papers in peer-reviewed international conferences and journals. His current research interests include middleware, software agent, and trustworthy computing.