Contributions, Collaborations, and Transitions: Paid and Volunteer Developers in the Rust Community

YUXIA ZHANG[*], Beijing Institute of Technology, China
KLAAS-JAN STOL, University College Cork and Lero, Ireland
MINGHUI ZHOU, Peking University, China
QUNHONG ZENG and MIAN QIN, Beijing Institute of Technology, China
HUI LIU, Beijing Institute of Technology, China

An increasing number of companies are contributing to open source software (OSS) projects by assigning their employees to advance their business objectives. These paid developers collaborate with volunteer contributors, but the differing motivations of these two groups can sometimes lead to conflicts, which might endanger the OSS project's sustainability. This article presents a multi-method comparative study of paid developers and volunteers in Rust, currently one of the most popular open source programming languages. We compare volunteers and paid developers through contribution behavior, social collaboration, and long-term participation. Then, we solicit volunteers' perceptions of paid developers and explore the emotions caused when volunteers transition to paid roles. We find that core paid developers tend to contribute more frequently; peripheral paid developers contribute bigger commits and focus more on implementing features; both core and peripheral paid developers collaborate more with volunteers but less intensively than expected; and being paid correlates positively with becoming a long-term contributor. Our study also reveals existing unfamiliarity and prejudices among volunteers towards paid developers, and that volunteer-to-paid transitions can evoke negative community sentiments. This study suggests that the dichotomous view of paid vs. volunteer developers is too simplistic and that further subgroups could be identified. Contributing organizations should become more sensitive to how OSS communities perceive them when they attempt to get involved and make improvements.

CCS Concepts: • **Software and its engineering** → **Collaboration in software development**; **Open source model**; **Programming teams**.

Additional Key Words and Phrases: Open source software, paid developers, volunteers, sustainability, Rust

ACM Reference Format:
Yuxia Zhang, Klaas-Jan Stol, Minghui Zhou, Qunhong Zeng, Mian Qin, and Hui Liu. 2026. Contributions, Collaborations, and Transitions: Paid and Volunteer Developers in the Rust Community. *ACM Trans. Softw. Eng. Methodol.* 1, 1 (January 2026), 32 pages. https://doi.org/10.1145/3789503

1 Introduction

Open source software (OSS) has become the backbone of society's technical infrastructure. A recent report [88] estimates that up to 96% of commercial codebases contain OSS. Motivated by the huge impact of OSS, a large number of companies have embraced open source to accelerate innovations,

[*]Corresponding author.

Authors' Contact Information: Yuxia Zhang, yuxiazh@bit.edu.cn, Beijing Institute of Technology, China; Klaas-Jan Stol, k.stol@ucc.ie, University College Cork and Lero, Cork, Ireland; Minghui Zhou, zhmh@pku.edu.cn, Peking University, China; Qunhong Zeng, qunhongzeng@bit.edu.cn; Mian Qin, qinmian@bit.edu.cn, Beijing Institute of Technology, China; Hui Liu, liuhui08@bit.edu.cn, Beijing Institute of Technology, China.

Please use nonacm option or ACM Engage class to enable CC licenses
This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1557-7392/2026/1-ART
https://doi.org/10.1145/3789503

new approaches, and best practices [40, 42]. For example, most work in the Linux kernel today is done by paid developers who are hired by companies, emphasizing the important role of paid developers in OSS [22, 93].

Companies participate in OSS projects by tasking their employees to make specific contributions to an OSS project, or by hiring volunteers from the project to do the same. (We use the term 'paid developer' to refer to those who are paid and tasked by a company to make contributions to a specific OSS.) A company's business objectives may affect the direction and scale of how its paid developers make contributions to the OSS project that the company is involved in [109, 116]. Volunteers contribute to open source projects for other reasons, frequently for intrinsic motivations, such as fun, but also to make a better product (to *"scratch an itch"* [77]), rather than pursuing commercial objectives.

Corporate participation in open source is also a source of some concern for several reasons. Contributions from a company may be highly specialized and specific to the company, and be at odds with the project's roadmap; integrating such contributions into the main development branch may not be welcome [7]. Recent work found that dominant corporate participation may be a threat to a project's survival rate [107]. Diverging motivations and behavior that do not comply with a project's norms may lead to conflict within a project. Such conflicts may lead to companies withdrawing their support from an OSS project and may also result in the turnover of individual developers, including both paid developers and volunteers [85, 92].

One popular project where corporate involvement has also led to concerns is Rust, an open source programming language governed by multiple teams [91], and which has been the 'most loved' language for seven years in a row [20]. Hundreds of companies globally use Rust in production, replacing critical systems previously written in C/C++ [91], and since late 2022, Rust can also be used to write Linux kernel components. One report estimated that 28% of commits to Rust were made by paid contributors [70]. There are considerable concerns about corporate involvement in Rust. One Rust member highlighted a lack of responsiveness to community concerns in a blog post [101]:

> "The core team repeatedly dismissed and maligned several members' concerns about the involvement of a company so heavily involved with producing spyware."

Shortly after, the full moderation team that was responsible for upholding the code of conduct of the Rust project, resigned on November 22, 2021 [3, 92]. The moderation team posted that their resignation is *"in protest of the Core Team placing themselves unaccountable to anyone but themselves"* [92]. The resignation of the moderation team and negative perspectives of corporate involvement have gained wide attention in the Rust community [59], which would have led developers in Rust to think more critically about the participation of companies in general, as well as governance issues.

Maintaining sustainable development is crucial to OSS ecosystems, and a clear understanding of the characteristics of both the developers assigned by companies and volunteers in the same OSS project is essential to support this goal. As such, Rust can be considered an intrinsic case study [23], that is, a case study selected on its own merits for its intrinsic features. Thus, the Rust project is a potentially fruitful source to develop a better understanding of the differences and similarities between paid developers and volunteers.

This study is guided by five research questions. First, we seek to compare paid developers and volunteers in terms of contribution frequency, scale, and tasks. Employed developers typically would work full-time on certain tasks, whereas many volunteers would have normal daytime jobs, and contribute in their spare hours. Further, paid developers are more likely to work on specific tasks that they were assigned than volunteers, who would typically self-select tasks to work on. Thus, our first research question (RQ) is:

> RQ1: Do paid developers and volunteers differ in their contribution behavior to the Rust project?

RQ1 compares paid developers and volunteers at an individual granularity. The success of an OSS project relies on collaboration among its contributors [95]. However, the differing motivations and organizational contexts of the two groups may shape distinct interaction patterns. Does it matter whether contributors are paid or not when they collaborate? Do paid developers and volunteers "play nice" together—do they work together on modules or code, or do they work on distinct parts without overlap? Understanding whether and how they collaborate helps reveal the social dynamics of OSS development. Thus, our second question is:

> RQ2: Do paid developers and volunteers collaborate within the Rust project?

Of key importance is an OSS project's continuity and sustainability; long-term contributors (LTC) play a key role in the long-term health of a project [64, 114, 116]. Several factors that affect a contributor's likelihood of becoming an LTC have been studied, such as contribution models, corporate dominance, and social links. However, prior research has not addressed the question of whether becoming a paid developer affects becoming an LTC. Hence, we ask:

> RQ3: Does being paid or not affect the likelihood of a Rust developer becoming a long-term contributor to the Rust project?

Aside from actual contribution, how volunteers *perceive* paid developers is also important, because volunteers' perceptions of their paid 'colleagues' may shape their collaboration with paid developers. As we illustrated above with the events around Amazon's involvement in Rust, volunteer developers may perceive paid developers as having a negative influence. However, no evidence exists on this topic, hence, we ask:

> RQ4: How do volunteers perceive the participation of paid developers in the Rust project?

Volunteers in OSS projects can also become paid developers if they have an outstanding contribution record, and if they need to secure financial support. Developers who transition from volunteers to paid developers are usually key to the development of an OSS project and can have a big impact on its development roadmap. While paid developers *may* share a passion and interest for a project, ultimately they have a different "master to serve." An infamous example of this is the case of the Debian project when a decision to pay two Debian volunteers (the 'Dunc-Tank' experiment) led to a considerable uproar in the community [33, 71], and led some volunteers to reduce their involvement: *"Some people who used to do good work reduced their involvement drastically"* [9]. The literature shows that developers' affective state can have an impact on work performance and team collaboration [37, 68]. Yet, little is known about how such role transitions (from volunteer to paid developer) affect the emotional dynamics within OSS communities. Understanding these emotional responses is crucial, as they may influence community cohesion. Thus, our fifth research question is:

> RQ5: What emotions arise when Rust volunteers are hired by companies to continue their work on Rust?

We conducted a mixed-methods study to address these questions. To answer RQ1, RQ2, and RQ3, we developed five hypotheses and tested these using commit data from Rust's code repository. To answer RQ4, we conducted a survey to gather volunteer developers' perceptions of paid developers; we focused specifically on the five hypotheses mentioned above; we answer RQ5 by conducting emotion analysis of the comments about a Rust volunteer becoming a paid developer. This article extends our previous study [106], which addressed RQ1 (contribution behavior), RQ3 (likelihood of becoming a LTC), and RQ4 (perceptions of paid developers). We extended this initial study by two complementary research questions (RQ2, investigating collaboration between volunteer and paid developers; and RQ5, investigating emotional responses to volunteers being hired by companies). We further refined the analysis of the initial RQs.

The findings show that paid developers and volunteers differ in several ways in how they contribute. For example, core paid developers tend to contribute more frequently than core volunteers. Although paid developers have more collaborations with volunteers, the collaborations are significantly fewer in number than expected. Further, we found that being paid is positively associated with becoming a long-term contributor to Rust. The survey results suggest that most volunteers either have prejudices or are unfamiliar with paid developers. Finally, the results of emotion analysis show that developers changing from Rust volunteers to being paid tend to invoke negative emotions, such as anger, disgust, fear, and sadness. The proportions of emotions vary among the platforms where the posts about the transition of volunteers to paid developers are published. Understanding the differences and similarities between paid developers and volunteers in OSS projects and to what extent we understand the two groups, can aid project leaders in steering their community. Further, companies can refine their OSS engagement strategies by identifying gaps between their contributions and volunteer expectations. Recognizing shared traits between the two groups may promote fairer perceptions of paid developers, fostering more positive, productive, and sustainable collaboration.

2 Hypothesis Development

Initially, open source software was developed by volunteers; as Raymond characterized it, developers (or 'hackers') wrote software to *"scratch an itch"* [77]. As paid developers are now commonly members of open source projects alongside volunteers, conflicts may arise due to diverging views on the future of a project. While paid developers *may* share a passion and interest for a project, ultimately they have a different "master to serve." We review prior literature that compares paid and volunteer open source developers in terms of contribution characteristics, collaboration, and long-term participation, resulting in five hypotheses.

2.1 Contributions of Paid and Volunteer Open Source Developers

Many studies have previously discussed differences between paid developers and volunteers, with varying ways to distinguish these two groups. Several early studies established that some OSS developers were paid for their work [38, 43, 55]. These survey studies indicated that many respondents (38%-55%) contributed during normal working hours. These studies considered open source development *work* that is effectively paid for by companies who support OSS communities—whether these companies are aware of it or not [55]. Riehle et al.'s 2014 study of project repositories considered contributions made from 9 am to 5 pm during weekdays as 'paid work'; based on this heuristic, they estimated that 50% of OSS work is paid for by companies [79]. However, a recent study by Dias et al. of five company-initiated OSS projects found that most contributions happen between 9 am and 5 pm for *both* paid developers and volunteers [25], which casts some doubt on the conclusions in Riehle et al.'s earlier study [79].

The studies cited above do not clearly differentiate between what we label 'purposeful' sponsorship through OSS contributions whereby companies purposefully task developers with contributing to OSS projects, in alignment with corporate strategy, and 'de facto' sponsorship whereby developers contribute during work time but were not explicitly instructed to do so, or without their managers' approval and awareness. However, even early studies of OSS highlighted purposeful sponsorship; for example, German reported in 2002 that Red Hat dedicated six full-time developers to the GNOME project [34, 35]. In the 20 years or so that have passed since these early studies, purposeful sponsorship of OSS projects has become much more prevalent [cf. 21, 47, 70].

Determining whether an OSS developer is paid or a volunteer remains a challenge [5, 15]. Whereas early studies have relied on self-reporting surveys to characterize OSS communities [38, 43, 55],[1] in more recent years researchers have started to infer this information from software repository data using heuristics, such as the time at which commits are made [79], the email address domain associated with commits [16, 70], or the `site_admin` flag that can be set for contributors in GitHub organizations [26]. For example, Dias et al. [26] define internal developers as those who are paid by the organization who open-sourced the project, whereas external developers are not employed by that organization. In their study on effort estimation in OSS, Robles et al. [82] distinguished between full-time and non-full-time developers, implying that those who are full-time are paid by their employer to contribute to OSS projects, whereas those who are non-full-time may be either paid or volunteer. Barcomb et al. argued that from a community's perspective, volunteers may be indistinguishable from non-volunteers [5].

Studying the differences between paid developers and volunteers provides an understanding of these two categories of developers, who may have different reasons to contribute. Dias et al. found that both internal developers and external developers are rather active: internal developers are responsible for ca. 46% of the pull-requests vs. external developers' ca. 54% [26]. Another study using the same dataset and heuristic, investigated differences between employees and volunteers in terms of their contributions and acceptance rates [73]. Volunteers face considerably more rejections (up to 26 times more rejections of contributions) than employees, and have to wait considerably longer than employees (on average 11 vs. 2 days, respectively). Another study by Dias et al. found that volunteers' contributions focus primarily on refactoring, and that corporate developers (employed by the projects' initiating company) focus more on management (including documentation) [25].

Rather than focusing on *when* contributions are made (work hours vs. spare time), we suggest that a difference between paid developers and volunteers lies in the frequency with which they contribute. Given that paid developers have more time to dedicate to the project, and that a lack of time is a common reason experienced by volunteers, we hypothesize that:

Hypothesis 1 (H1): Paid developers contribute more frequently than volunteers.

Another way in which paid developers and volunteers might differ is the way they contribute code; Pinto et al. previously observed that contributions by paid developers tend to be larger than those from volunteers [73]. For paid developers, who work on behalf of their companies, achieving their set goals may be more important than concerns such as maintainability of the code [66, 84]. Merging code changes to an OSS repository is an onerous and time-consuming process for maintainers [80]; paid developers may want to reduce the overhead of submitting commits and contribute whole features, or complete tasks with as few commits as possible. Volunteers, on the other hand, may have limited time and do not pursue business-specific requirements to make large code changes. For paid developers, these considerations may not exist. Thus, we posit:

Hypothesis 2 (H2): Paid developers contribute larger chunks[2] of code in commits.

Further, companies have specific business objectives when contributing to OSS, such as integrating their own products [104, 109], and paid developers are typically assigned particular tasks that are important to their employer's feature roadmap. Moreover, the effort required to implement new

[1]Even self-reporting can be problematic: while respondents may contribute to OSS during business hours, they may do so without their managers' awareness (as in the case of de facto sponsorship). In that case, are they paid for their open source work?

[2]Inspired by Kolassa et al. [53], we measure a chunk as the sum of two variables: number of lines of code added and number of lines of code removed in a commit.

features is generally greater than that for other development activities [103], and paid developers can dedicate more focused and sustained time to contributing to OSS projects. On the other hand, volunteers tend to contribute to open source projects because of their interest and 'passion' [1, 100]; while motives will vary, volunteers who contribute to a project will want to see that project succeed and improve over time. This means that rather than delivering business-ready features, we argue they are more likely to make improvements through bug fixes and non-functional improvements. Thus, we propose:

Hypothesis 3 (H3): Paid developers are more likely to contribute features than volunteers.

2.2 Collaborations among Paid and Volunteer Open Source Developers

OSS development relies on the collaboration of contributors distributed worldwide. Several studies have investigated individual contributors' collaboration, including collaboration visualization, understanding, and improvement in OSS development [14, 62, 63]. A few studies have explored how companies collaborate in an OSS project. For example, in their study of the OpenStack ecosystem, Zhang et al. found that companies collaborate either intentionally or passively, or choose to work in isolation [110]. Corporations' culture and software development processes differ considerably from volunteers in open source communities [85], which may inhibit any collaboration among paid and volunteer developers. Further, paid developers are a minority in most OSS projects [5, 55], and this is particularly true in the Rust project: over 90% of its contributors are volunteers [**?**]. Companies are more likely to get involved in specific OSS projects that have already established themselves as mature projects and thus can offer considerable business value [50], such as the Linux kernel [93], OpenStack [105, 109], and Eclipse [12]. This suggests that paid developers join an OSS project on their employer's behalf when it is already mature; this in turn may limit their collaboration with key volunteers who have gained a reputation or occupy positions of community leadership. Thus, we suggest that:

Hypothesis 4 (H4): Paid developers tend to collaborate less with volunteers.

2.3 Becoming Long-Term Contributors

A key factor for the sustainability of OSS projects is to attract LTCs [114, 115]. Open source communities attract a variety of developers with varying motivations to contribute. The duration of contributors varies as well; some contributors contribute only once [56, 57]. Lee et al. [57] found that the main reason for peripheral contributors *not* to continue contributing is that they see "nothing else to contribute." Calefato et al. studied open source developers who took a break from contributing [11]. Amongst others, they found that all core developers in the 18 projects they studied have taken a break in activity at least once.

Of particular interest to open source projects is, of course, whether developers continue to contribute, i.e., whether they become LTCs. Several studies have focused on newcomers to open source [13, 87, 90], developer turnover and retention in open source communities [58, 105, 116]. Zhang et al. found a positive association between the diversity of companies' contribution models and the number of volunteers [109] and a negative impact of company domination on the sustainability of OSS projects. Valiev et al. [96] found that the involvement of companies has a significant negative effect on the sustainability of projects in the PyPI ecosystem, with a shared opinion among interviewees that companies' support is not sustainable long term.

Zhou and Mockus studied factors that affect the chances that a contributor becomes an LTC [115]. They found that contributors who get at least one issue reported in the first month to be fixed can double their odds of becoming an LTC; the popularity of projects and low attention from peers can reduce those odds. Lin et al. found that developers are more likely to remain active when they start contributing to a project early, modify rather than create files, and focus primarily on code rather than documentation [58].

Intrinsic motivations may lead to a long-term bond between volunteers and an OSS project. Previous work found that companies may withdraw from OSS projects for various reasons [105], for example, when a company changes its business strategy towards an OSS project. In such a case, paid developers may no longer be tasked to work on an OSS project. Thus, we propose:

Hypothesis 5 (H5): Paid developers are less likely to become long-term contributors.

3 Study Design

We conducted a mixed-method study of the Rust project using quantitative and qualitative methods [28]. We selected Rust as a case study to investigate the tensions that might arise when companies participate in an open source project. Initially created by Mozilla engineer Graydon Hoare in 2006, the project is now managed by the Rust Foundation [91]. Rust is used in production by hundreds of companies worldwide, who also participate in the development of the project. The influence of corporate participation on the project, however, has led to concerns from the community [92, 101]; as mentioned briefly earlier, these tensions led to the resignation of the project's moderation team.

We collected and cleaned the commit data of 4,117 Rust contributors (Sec. 3.1 and 3.2), conducted comparisons to determine the differences and similarities between paid developers and volunteers (Sec. 3.3), built a social network to analyze their collaboration likelihood (Sec. 3.4), and created a statistical model to determine the probability of becoming long-term contributors (Sec. 3.5). We then surveyed volunteer Rust developers (Sec. 3.6) to collect their views on paid Rust developers and conducted an emotion analysis of the caused comments when a Rust volunteer became a paid developer (Sec. 3.7). Figure 1 shows an overview of our methodology. An appendix provides the data, scripts, and other resources [108].

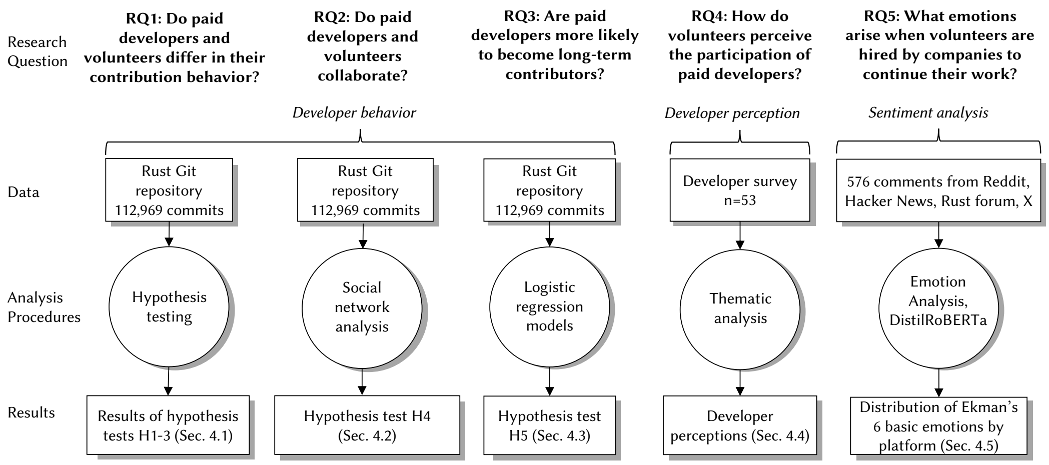

Research Question
RQ1: Do paid developers and volunteers differ in their contribution behavior?
RQ2: Do paid developers and volunteers collaborate?
RQ3: Are paid developers more likely to become long-term contributors?
RQ4: How do volunteers perceive the participation of paid developers?
RQ5: What emotions arise when volunteers are hired by companies to continue their work?
Developer behavior
Developer perception
Sentiment analysis
Data
Rust Git repository 112,969 commits
Rust Git repository 112,969 commits
Rust Git repository 112,969 commits
Developer survey n=53
576 comments from Reddit, Hacker News, Rust forum, X
Analysis Procedures
Hypothesis testing
Social network analysis
Logistic regression models
Thematic analysis
Emotion Analysis, DistilRoBERTa
Results
Results of hypothesis tests H1-3 (Sec. 4.1)
Hypothesis test H4 (Sec. 4.2)
Hypothesis test H5 (Sec. 4.3)
Developer perceptions (Sec. 4.4)
Distribution of Ekman's 6 basic emotions by platform (Sec. 4.5)

Fig. 1. Overview of study design

3.1 Data Collection

Rust uses Git as its version control system. We obtained the commit metadata from GitHub, which hosts the repository of the Rust project [18] by querying GitHub's REST API. The period of the dataset is over 11 years, starting at Rust's creation date (July 7, 2010) until December 16, 2021, containing 114,074 commits.

Each commit includes the name and email of the author, a timestamp, a message description, and the 'diffs' (the raw content of changes between different versions of a file). Previous research found that automated bots, rather than human developers, also submit commits [2, 24]. We identified four bot accounts, i.e., 'bors', 'dependabot-preview[bot]', 'ImgBotApp', and 'dependabot[bot]', which together submitted 138 commits, based on the patterns identified by previous work [2, 24, 94, 109]. We also removed rollback and merge commits, leaving a total of 112,969 commits for analysis. Below, we describe the procedures for cleaning the data.

3.2 Data Cleaning

3.2.1 Merging Multiple Identities. Developers may have multiple accounts when contributing to open source projects, each of which may have a different name and email address [2, 8, 54]. To establish an accurate representation of a developer's activity and contributions, it is necessary to merge multiple identities that belong to the same developer. We addressed this problem by using a rule-based method [117], which augments the developer's name and email address, and has been shown to result in a high level of accuracy (with a precision close to 100%). For example, accounts <John-Smith, johnsmith@gmail.com> and <John Smith, johnsmith@gmail.com> will be merged because of the same email. The rules also apply to accounts with the same names. Using this approach on an initial list of 4,673 author identities, 556 were merged, resulting in a list of 4,117 distinct authors. To assess the accuracy of this identity merge, we performed a manual verification described in Sec. 3.5, through which we established that the accuracy of developers' identities has a 95% confidence interval of [0.99, 1].

3.2.2 Identifying Paid Developers. As mentioned earlier, determining whether a developer is paid or a volunteer is not straightforward because developer affiliations are not directly recorded in Git commits [109], and the Rust community does not have an official record of its contributors' affiliations. We followed approaches used in other studies [16, 70, 116]. We first identified a developer's affiliation at the time of each commit they made to Rust by the domain of their email address. More specifically, if a developer uses an email with a free or general provider domain, such as "gmail.com", we consider them to be a volunteer. We used a list of free email provider domains maintained in GitHub [48], which has been verified and used in other studies [96, 110], to identify volunteers in the Rust project. Similarly, we assumed that every developer using an email registered at a company or organization domain is a paid developer. For example, if a developer used an email that ends with "@mozilla.com", they were classified as a Mozilla employee, i.e., a paid developer in Rust.

While this method works in many cases, this technique is not perfect. Paid developers might use a personal email address to submit commits, even when they do so on their employer's behalf—or, indeed, vice versa. To improve the accuracy of developer affiliations, we conducted additional checks by searching the Internet (using "Rust" and the developer's name as keywords) and inspected the first 20 results. We analyzed pages from LinkedIn or Rust's official website, or the developer's personal homepage if it were accessible, to determine whether a developer is a volunteer or paid to make contributions to Rust. Specifically, we carefully reviewed all content related to their involvement in Rust, with particular attention to their affiliations and the corresponding timeframes. When conflicting signals appeared—for example, a developer committing via a company email but declaring themselves as a volunteer—we chose to respect the developer's self-declaration. We used this process to manually validate the top 500 developers in our dataset, ordered by the number of commits, who together made 100,136 (87.9%) commits. In only 17 cases (3.4%), we identified discrepancies, i.e., developers contributing with a corporate email address, while they specified on their résumé to be a volunteer contributor to Rust. In those cases, we registered their affiliation as "volunteer" and checked all developers with the same domains. Finally, we also performed a manual verification with developers described in Sec. 3.5 and obtained a 94.3% accuracy.

In our dataset, we identified 250 paid developers from 55 companies. Among them, Mozilla employed 54 developers who contributed approximately 23% of all commits, making it the largest corporate contributor to Rust apart from volunteers.

3.3 Comparing Paid and Volunteer Developers

To address RQ1, we conducted a series of analyses that sought to determine whether paid and volunteer contributions to Rust differ, considering three metrics: contribution frequency (*H1*), change size (*H2*), and task categories (*H3*).

3.3.1 Measuring Contribution Performance. Contribution frequency was measured by the number of commits submitted by a developer within a fixed time frame. Following previous studies [69, 96, 97], we set the time frame to be one month, i.e., if a developer has contributed 100 commits during 3 months, their contribution frequency will be 33.3 ($=\frac{100}{3}$). We measure change size by a widely used metric [89, 97, 105]: lines of code (LOC) in a commit, calculated by the sum of added and deleted code lines. For each developer, we took the median of the LOCs of all contributed commits to represent their change size.

To determine the type of work carried out in a commit, we adopted the classification of dos Santos and Figueiredo [27]: 'feature,' corresponding to new feature introduction; 'corrective,' related to fault fixing; 'perfective,' based on system improvements; and 'non-functional,' referring to documentation and non-functional requirements. We used their model, which uses natural language processing, to classify the commits in our dataset. While other models exist, we used this model because it has a high F-measure (91%) and is well documented. The output is a list of commit types. The model only assigned the 'unknown' label to 553 (0.5%) commits, most of which have an uninformative message, such as *"Apply suggestions from code review."* When analyzing developers' task preferences, we did not consider commits labeled as 'unknown.' We validated the performance of the classification model by randomly selecting 150 commits (error margin: 8%, confidence level: 95%) and manually labeling by two of the authors. The results show that 84% of commits are given the same labels from both the classification model and the manual validation. The high consistency demonstrates the fit of the classification model we selected.

3.3.2 Classifying Developer Roles. The distribution of contributions in OSS projects frequently follows the Pareto principle [36, 65, 109]: a relatively small group of developers (the core) drives most of the work, while a larger group of contributors (the periphery) contributes a considerably smaller amount. While the distribution between these two groups varies, Mockus et al., who first observed this, identified a typical 80/20 distribution [65]. Independent of whether developers are paid or volunteers, they may be core or peripheral contributors. Instead of comparing paid and volunteer developers at a superficial level, we answer RQ1 at a finer granularity by comparing paid and volunteer developers for each group, i.e., core and peripheral.

We adopt the widely used core-periphery structure [36, 96] to classify developers' roles based on the commit-based heuristic [17, 46]. Core developers are deemed to account for 80% of the commits in Rust. The remaining developers are classified as peripheral developers.

3.3.3 Comparing Differences. Based on the measures of contribution frequency, change size, and task categories, we explored the three hypotheses posited above to answer RQ1. Given the non-normal distribution of the data, we used the non-parametric Mann–Whitney U test [60] to test for differences between paid developers and volunteers in different role groups (i.e., core and peripheral). To reduce false discoveries brought by multiple hypothesis testing, we adjusted all *p-values* using the Benjamini-Hochberg correction method [6]. We also report the effect size [32], which assesses the strength of the relationship between investigated variables. We used the library `statsmodels` for Python to calculate these statistics.

3.4 Measuring Collaborations with Social Network Analysis

To address RQ2, we investigated whether paid developers and volunteers work in an isolated way or collaborate when contributing to the development of Rust. We applied social network analysis to answer this research question, with developers represented as nodes, and collaborations as edges between those nodes. Prior studies model developer collaborations based on social connections found in software development artifacts, such as commits [110], code reviews [51], and issue reports [61]. As evidenced by Meneely and Williams [62], developer collaborations as measured by commits to the same source code file are well reflected in developer perceptions when compared with other development activities; that is, this measure is consistent with developers' observations. Thus, in this study, we also selected the commit data to model developers' collaborations. Based on the same principle, and following other studies conducting social network analysis [62, 89, 110], we considered the existence of collaboration if two developers changed the same file within a period of one month (not considering any commits that were reverted, as we noted in Sec. 3.1). We used the number of files that were edited by two developers as the weight of the edge connecting those two developers, indicating the frequency of collaboration.

The generated network contained 3,887 nodes (i.e., developers), including 250 paid developers, 3,637 volunteers, and a total of 46,551 edges between these nodes. We measured collaboration by calculating the percentage of paid developers and volunteers in each developer's direct neighbor nodes (i.e., collaborators). The number of collaborations is the sum of weights on the edges of the built social network. We followed the same procedures used to address RQ1, dividing paid developers into core and peripheral groups to provide a detailed comparison of their collaboration likelihood to validate *H4: Paid developers tend to collaborate less with volunteers.* For each group, we employed two complementary approaches to measure paid developers' collaboration likelihood: *macro-level and micro-level measurements.*

At the macro level, we used the Wilcoxon signed-rank test [99] to compare the likelihood of collaboration between paid developers and volunteers versus that among paid developers. This non-parametric test captures overall tendencies across the Rust project, helping us detect whether paid developers generally collaborate more with other paid developers or volunteers.

At the micro level, given the imbalance in the distribution of paid and volunteer developers across the network, we constructed a permutation-based null model [30] to assess whether the observed collaboration with volunteers is stronger or weaker than would be expected by chance. Specifically, randomly shuffled the paid/unpaid labels of developers 1,000 times while preserving the underlying network structure [30, 74]. For each randomized instance, we calculated the proportion of paid developers' collaboration involving volunteers, and compared the observed value to this null distribution using a z-score and a one-sided p-value.

3.5 Modeling Long-Term Contributors

Following Zhou et al.'s [114] definition of LTCs,[3] we characterized a contributor as an LTC if:

- they have contributed to Rust for 3 years or more, and

[3]defined as: "*A participant who stays with the project for at least three years and is productive [114].*"

- they rank in the top 20% in terms of number of commits in at least three years.[4]

Both requirements should be satisfied. We found that 1,508 out of 4,117 developers (36.6%) joined Rust less than three years at the time of the study, and thus, we could not assess whether they would become an LTC (based on the definition above); thus, we removed these developers from the analysis.

After determining whether a developer has been an LTC, we fit logistic regression models [44] to investigate the association between being paid or not, and the likelihood of becoming an LTC, and report the effect size of the independent variable with odds ratio. Previous studies [114, 115] have found that a newcomer's development ability, willingness to participate, and the environment at the time of joining are associated with the likelihood of becoming an LTC. Following previous work [114, 115], we measured a new contributor's willingness and ability by the number and types of tasks (e.g., the willingness and ability to fix bugs are stronger than writing documents [114, 115]). Moreover, we extend the measurement of contributors' willingness and ability to change size (measured by LOC) because submitting large code changes may require huge efforts (what we would call 'strong willingness') and can also convey a developer's ability. Since all developers share the same environment (i.e., Rust community), we excluded the environment factor. The remaining measures are calculated based on the commit data of developers during their first month of participation in Rust, as we sought to investigate whether being paid is linked to the probability of a newly joined contributor becoming an LTC.

3.6 Volunteers' Views on Paid Developers

The quantitative comparison of paid developers and volunteers presented in the previous sections can convey behavioral differences. How OSS volunteers *perceive* paid developers cannot be determined from archival data, but clearly plays a role in whether their collaboration is harmonious or subject to conflict. Therefore, we conducted a survey study to gain insight into how volunteers perceive paid developers in Rust while focusing on the five hypotheses. The survey included both closed and open-ended questions and required approximately five minutes to complete. The remainder of this section outlines the question design, pilot testing, recruitment process, and analysis of responses.

3.6.1 Questions. The survey sought to establish volunteer developers' perspectives on paid developers within Rust. Specifically, we asked whether respondents agreed with the five hypotheses (using a 5-point Likert scale, with anchors 1=Strongly disagree and 5=Strongly agree) [49] and their perspectives with an open-ended question.

As part of the survey, we also conducted a validation of developer identities. Following previous work [109], we adopted a less intrusive approach, i.e., for each unique pair of a developer's identity and affiliation, we randomly selected one commit and recorded the affiliation (or labeled as "volunteer"). We then asked respondents to either confirm or refute that these commits were done by them with the given pair of identity and recorded affiliation.

The lead investigator's institution did not require ethics approval for this study, and so while we did not have a formal study ethics application approved, we followed established best practices. Respondents were informed of the purpose of the study, and of the voluntary nature of participation. No personally identifiable information was collected. The collected data were stored securely with multi-factor authentication. All responses were treated anonymously, and we ensured that no respondent could be identified through our reporting.

[4]The Pareto principle (20/80) phenomenon have been frequently encountered in software engineering [65, 109, 113], and so we deem 20% a reasonable threshold.

3.6.2 Pilot Study. We conducted a pilot with five randomly selected developers first. One of the contacted developers is the Executive Director of the Rust Foundation, who offered to help us by encouraging developers who received the survey invitation to participate. Based on the feedback from the pilot, we added minor changes in some key terms in the hypotheses, such as adding the sentence "(namely, staying for a relatively long time and making significant contributions)" to explain what a long-term contributor is, and refining collaboration to working on the same code modules or files.

3.6.3 Survey Respondents and Responses. We randomly selected 350 developers from those who were ranked in the top 20% by their commit count, with an error margin of 5% and a confidence level of 95%. We did not exclude paid developers at this point, allowing us to check the affiliations of developers, if indeed they were paid. Among the 350 survey respondents, 59 were paid developers. We then sent the revised survey to the 350 selected developers; of those, 122 emails were not delivered.

After two months, we obtained a total of 53 replies (a response rate of 23.2% ($\frac{53}{350-122}$)), of which five responses came from paid developers. Since we sought to understand volunteers' perceptions of paid developers' participation in Rust, only the volunteers' level of agreement on the five hypotheses and further explanations are considered for answering RQ4. The five responses from paid developers were used only to validate their identities and affiliations; one of them indicated a different affiliation that was manually corrected.

3.6.4 Qualitative Analysis. We followed the guidelines by Seaman [86] to code developers' explanations of their agreements with the five hypotheses:

(1) Initially, two investigators thoroughly read the original responses. After getting a comprehensive understanding of the collected open-ended answers, we examined developers' responses sentence by sentence and transformed key phrases into concise labels as initial codes. Table 1 shows a coding example.
(2) Within our established set of codes, we revisited the meaning of each code and merged codes that had overlap. For instance, we discovered that "more working hours" and "full-time job" both suggested that paid developers have more time to make contributions to Rust, then we merged them into one code "more time."
(3) To minimize the impact of personal bias, a series of face-to-face meetings were conducted (approximately 4 sessions, each lasting 20 minutes or more) to discuss the coding results and resolve any variance in coding. Through this process, we found agreement in the final set of categories of responses listed in Tables 4-8.

We provided the coding data and step descriptions in the online appendix [108].

Table 1. Example of open coding developer responses

Response text	Codes
"Volunteers usually develop for fun. Obviously, implementing features is more exciting than fixing bugs. Developers are paid to develop Rust because Rust is critical to their company. So, it is more likely that paid developers fix bugs and do non-functional improvements according to the needs of the company."	Volunteers develop for fun; non-functional improvements; Paid developers follow the company's needs

3.7 Analysis of Developer Emotion when Volunteers are Hired

It is common for companies to hire volunteers who are established contributors in OSS projects that companies are interested in. In our initial investigation of Rust forums, we observed several discussions about this phenomenon. RQ5 explores how people react when organizations hire Rust volunteers. To this end, we first collected comments in the posts that discuss the role transition from volunteers to paid developers; we then conducted an emotion analysis based on these comments. Within the dataset constructed in Sec. 3.1 and 3.2, we identified 30 developers who had role transitions, and among them, 24 had transitioned from volunteers to paid developers. For each developer, we sought to collect background information regarding their transition; using Google's search engine, we used a search string composed of the developer's name, the term 'Rust,' and the company that hired them. We visited the top 20 links to collect any posts and discussions about the hiring of this developer by the given company. We also performed snowball sampling [72] based on the selected links. Specifically, if any comments linked to other posts about Rust volunteers becoming paid developers, we collected those online records too. We stopped retrieving when no new relevant links could be found. In this process, we identified 12 relevant posts, each containing between 6 and 166 comments (see Table 2).

Seven posts were from Reddit,[5] three posts from Hacker News,[6] and two posts published on X, formerly known as Twitter. For each post link, we applied Requests [19], a popular HTTP library for Python, to retrieve the comments of the selected posts. This procedure led to a total of 650 comments.

Following other studies [10, 83], we applied a DistilRoBERTa-based model [39] to classify any emotions within the collected comments. This model has been trained on a combination of six diverse datasets and can predict English text data into Ekman's six basic emotions [29], plus a neutral class: anger, disgust, fear, joy, neutral, sadness, and surprise. Specifically, we leveraged the public API of HuggingFace[7] to detect the emotions. We also validated the accuracy of the classified emotions. Specifically, we randomly selected 100 comments, and two investigators manually labeled any emotions in these comments. The results show that only for 17 comments, human-assigned labels were different from those assigned by the DistilRoBERTa-based model. The consistency rate (83%) indicates the emotion analysis tool is reliable to a certain extent. We provide the collected comments and identified emotions in our online appendix [108].

4 Results

4.1 Contributions of Paid and Volunteer

To compare paid and volunteer contributors, we first divided developers into two groups (see Sec. 3.3): peripheral and core, and then compared paid developers with volunteers on three metrics in each group: contribution frequency, change size, and task preference. Only a small portion of developers that are classified as core, 7.0% (272), is responsible for 80% of commits in Rust. The proportion of paid developers in the core group (20%, n=55) is higher than in the peripheral group (5%, n=195). We analyzed the comparison of contributions between paid and voluntary developers for each group as follows.

4.1.1 Peripheral Developers. We compare the contribution behavior of paid and voluntary developers in the periphery group on contribution frequency, change size, and task preference. The left pair of boxplots in Figure 2 shows that the distribution of contribution frequency of volunteers and paid

[5]https://www.reddit.com/
[6]https://news.ycombinator.com/news
[7]A popular community that helps users build, deploy, and train machine learning models.https://huggingface.co/j-hartmann/emotion-english-distilroberta-base

Table 2. Overview of the 12 posts about volunteers' transition to paid developers

Source	Post subject	No. comments	Illustrative comment	Emotion
Reddit	[Name] joins the Rust team as a full-time developer	6	"Awesome. He had been a super productive contributor already."	Surprise
	AWS hires Rust compiler team co-lead [Name]	6	"Do you work for Amazon? If so, do you know if they hire remote?"	Neutral
	The more things change...	29	"Good luck working at Amazon, I've heard a lot of horror stories so hope [Name] comes out okay."	Fear
	[Name] heading up Facebook's Rust team	70	"I'm really happy to see the significant wave of people joining FAMAG companies to work on Rust full time."	Joy
	[Name] working fulltime on the compiler at AWS	20	"Oh, right, I remember seeing the announcement!"	Surprise
	Starting at PingCAP	7	"PingCAP database looks very nice already. It is great they can hire good profiles like you!"	Joy
	[Name] is stepping back from the Rust core team	9	"Only because he wants to focus more heavily on other Rust things though: ... as lead of the language design team and tech lead of the AWS Rust Platform team."	Neutral
Hacker News	Rust [Name] is now at Apple working on Swift	61	"I have heard rumors that someone (I have no idea who) has been fired from Apple for contributing to Rust without permission a while back. Some employers do not like employees doing things on the side without permission."	Disgust
	AWS hires Rust compiler team co-lead [Name]	166	"It's great that AWS wants to have the Rust team onboard, but in my experience, at least AWS has been good at leeching Open Source, rather than fostering it. Open source isn't part of their culture."	Sadness
	How the AWS team will contribute to Rust's success	72	"No offense to [Name], but this is a completely vacuous post without a single concrete promise. I don't really see the point, except for PR, recruiting or internal signaling."	Anger
X	[Name] joining @Microsoft to work on the @rustlang...	95	"Congratulations!"	Neutral
	[Name] joining @Cloudflare as a systems engineer...	109	"That's awesome congrats and it's nice to see a go heavy shop adopting rust."	Joy

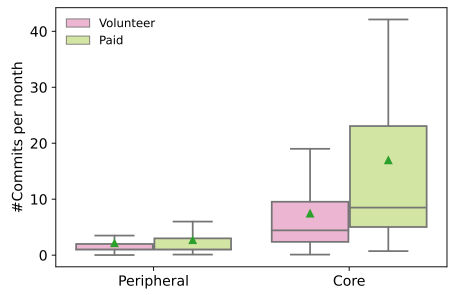


Fig. 2. Contribution frequency distributions of paid developers and volunteers in the two groups.

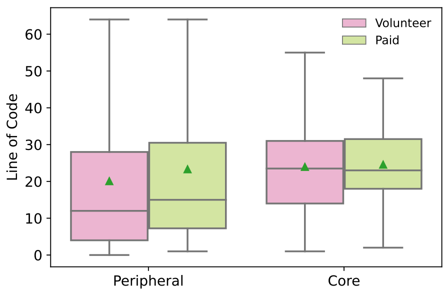


Fig. 3. LOC distributions of paid developers and volunteers in the two groups.

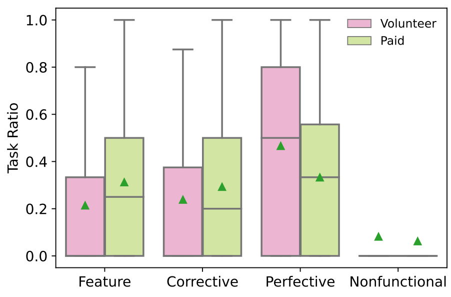


Fig. 4. Task distributions of peripheral developers.

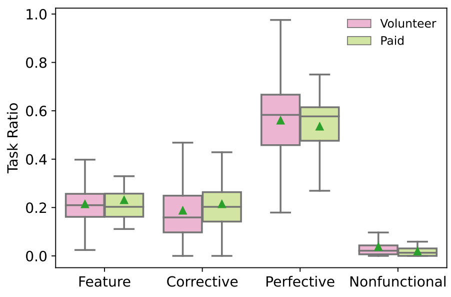


Fig. 5. Task distributions of core developers.

developers is similar. Specifically, the median frequency for both peripheral volunteers and paid developers is 1 commit per month. A Mann-Whitney U test [60] indicates there is no statistically significant difference ($p = .20$). Thus, *H1* is not supported for developers in the periphery.

The first paired boxplots in Figure 3 show the LOC distribution of peripheral volunteers and paid developers. Specifically, the median LOC of commits contributed is 12 for volunteers, and 15 for paid developers in the peripheral group. A Mann-Whitney U test to assess the significance of the difference between peripheral paid developers and volunteers in terms of the LOC distributions shows a statistically significant difference (adjusted $p = .009$), although with a small effect size[8] of .11. The results indicate that peripheral paid developers tend to contribute a bit larger code changes to OSS projects than peripheral volunteers, in support of *H2*.

Figure 4 shows the distribution of four task ratios of voluntary and paid developers in the peripheral group. In the median, 25% of commits contributed by peripheral paid developers implement features (nearly zero for volunteers); we observe statistical differences between the percentage distributions of features contributed by volunteers and paid developers (adjusted $p < .001$ and effect size = .20). Further, we can see that perfective commits has the highest ratio in both paid and voluntary developers: in the median, 33.3% of the paid developers' commits are perfective (50.0% for volunteers). These results indicate that, while developers spend most of their time on

[8]effect size ≥ 0.1 (small) ≥ 0.3 (medium), and ≥ 0.5 (large) [32].

improving code, paid developers tend to contribute slightly more towards features when compared to volunteers in the peripheral group. These results support *H3*.

4.1.2 Core Developers. Among the 272 core developers, 55 are paid developers and 217 are volunteers. The right paired boxplots in Figure 2 present the contribution frequency distribution of core voluntary and paid developers. When compared with peripheral developers, both paid and voluntary developers in the core group have a higher frequency; the median number of contributed commits per month is 4.4 for volunteers and 8.5 for paid developers. This difference is significant (*adjusted* $p < .001$) with a medium effect (= .43), and suggests that paid core developers tend to contribute more frequently than volunteer core developers of Rust. These results lend support to *H1* in the core group.

The second pair of boxplots in Figure 3 shows the LOC distributions of paid and voluntary developers in the core group. Both distributions have similar median and average values. For example, core developers, whether they are paid or volunteer, contribute commits with a median of 23 edited lines of code. The result (p=.63) indicates there is no support for *H2* for core developers.

We also compare the four task distributions of core contributors, as shown in Figure 5. In the median, the most common task is 'perfective' for both paid and volunteers, followed by feature and corrective commits; non-functional tasks account for the smallest proportion of their commits. A Mann–Whitney U test [60] of feature distributions obtains a p=.80, indicating that there is no significant difference between paid and voluntary developers. Thus, *H3* is not supported for core developers.

Summary for RQ1: Paid developers and volunteers differ in their contribution behavior: paid core developers tend to contribute more frequently to Rust than volunteer core developers; peripheral paid developers tend to contribute bigger commits and focus more on implementing features when compared to peripheral volunteers.

4.2 Collaboration between Paid Developers and Volunteers

We use social network analysis to explore whether being paid affects development collaboration in Rust to test *H4*. Similar to how we addressed RQ1 in Sec. 4.1, we divide paid developers into peripheral and core groups, and then compare their collaboration likelihood with volunteers or paid developers in both macro and micro perspectives.

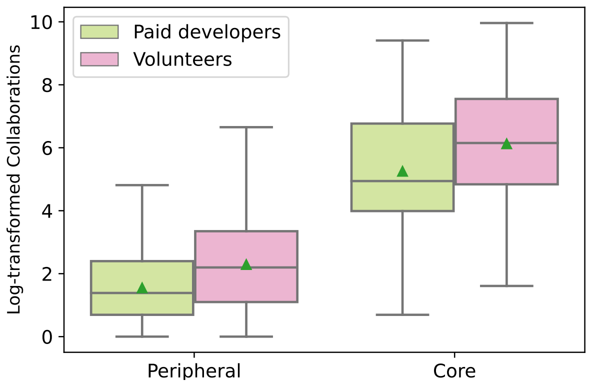


Fig. 6. Paid developers' collaborations with other paid developers (green box) and volunteers (pink box) across two groups: peripheral paid developers and core paid developers.

Based on the collaboration network we built in Sec. 3.4, for each paid developer, we counted the number of collaborations they had with volunteers and paid developers, respectively. Figure 6 shows the collaboration distribution of paid developers with volunteers and other paid developers, organized by peripheral and core paid developers. Across both groups, paid developers tend to collaborate more with volunteers. Specifically, in the median, peripheral paid developers tend to collaborate three times with other paid developers and 8.5 times with volunteers; core paid developers collaborate 3.4 times more with volunteers than with paid developers (468 vs. 139).

At the macro level, we compared the differences between the collaboration of the two groups' paid developers with volunteers and paid developers using the Wilcoxon signed-rank test [99]. The results show that both groups of paid developers are more inclined to collaborate with volunteers when compared with paid developers (all p-values < 0.0005). One possible reason may lie in the uneven distribution of volunteers and paid developers in the Rust community. Specifically, the number of volunteers is 15 times that of paid developers. Thus, we further performed a permutation-based analysis at the micro level. The results confirm that the observed level of paid developers' collaboration with volunteers is significantly lower than expected under the null model. Specifically, core paid developers exhibit an observed collaboration ratio of .73 compared to a randomized mean of .93 ($z = -1.36$, $p = .02$), while peripheral paid developers show an even stronger deviation (observed = .65, randomized mean = .93, $z = -6.31$, $p = .001$). These results indicate that although paid developers appear to collaborate frequently with volunteers overall, their actual preference is weaker than expected under random mixing, especially for peripheral contributors. Based on the results, we conclude that while the macro-level analysis does not support Hypothesis 4, the micro-level analysis provides supporting evidence.

Summary for RQ2: Although paid developers appear to collaborate more with volunteers overall, permutation-based analysis reveals that, at the micro level, their collaboration with volunteers is weaker than expected given the skewed distribution of paid and volunteer developers in Rust.

4.3 Becoming Long-Term Contributors

We now address RQ3; in Sec. 2.3, we hypothesized *H5: Paid developers are less likely to become long-term contributors.* To evaluate this, we applied a logistic regression model.

We considered two other factors that could have an impact on the probability of becoming an LTC: new joiners' willingness and ability during the first month of contributing to Rust. We used the commit type with the most commits, the number of contributed commits (i.e., the contribution frequency), and the change size (median LOC in all commits) to measure a developer's willingness and ability. We introduced detailed measurements of these factors in Sec. 3.5. Before fitting the attributes in the model, we calculated correlations but found no evidence of collinearity. Developers who joined Rust after 2018-12-16 were excluded because it would be impossible to determine whether or not they are LTCs at the time of our analysis. The regression is:

$$\text{isLTC} \sim \textit{Being Paid} + \textit{Contribution Frequency} + \textit{LOC} + \textit{Task Type}$$

Table 3 shows the results of the fitted model. The p-value of this model is < .001, indicating the regression is statistically significant [67]. The positive coefficient and p-value of *Being Paid* indicate that paid developers tend to have a higher probability of becoming long-term contributors when compared with volunteers, contradicting our hypothesis. One reason might be that paid developers have a secure income, a lack of which is a common reason for volunteer turnover in OSS projects [31, 45]. The positive relationship indicates an opportunity to cultivate long-term OSS contributors. Further, future studies aiming to predict LTCs should consider whether contributors are being

Table 3. Results of the logistic regression model (n=2,609)

Variable	Coefficient	Standard error	z	Pr(>|z|)
(Intercept)	−4.90	0.41	−11.95	0.00
Being Paid	1.55	0.25	6.29	0.00
Contribution Frequency	0.82	0.12	6.91	0.00
LOC	0.17	0.11	1.59	0.11
Corrective	−0.16	0.34	−0.48	0.63
Perfective	0.08	0.29	0.29	0.77
Nonfunctional	−0.01	0.58	−0.02	0.99

* LLR p-value = 2.96e-23; Pseudo R-square = 0.13.

paid. Another reason may be related to the initiation history of the Rust project: Rust began as a personal project in 2006 by Mozilla employee Graydon Hoare, and Mozilla officially sponsored the Rust project in 2009 and assigned over a dozen engineers to work on Rust full time [98]. Thus, this group of paid developers from Mozilla may perhaps more easily become LTCs, which may explain the positive coefficient and p-value of *Being Paid* and *isLTC*. Developers employed at other companies than Mozilla may not become an LTC as readily.

As expected, *Contribution Frequency* is statistically significant, suggesting that the participation frequency in the first month is an indicator of becoming an LTC in Rust. Contributing more commits demonstrates the willingness and ability of newcomers, which are two factors in becoming LTCs, and is consistent with prior work [115]. Surprisingly, *Task Type* is not statistically significant, suggesting that in the context of commits, the type of contributions does not influence whether newcomers become LTCs.

Summary for RQ3: Being paid is positively associated with becoming a long-term contributor in the Rust project, contrary to our hypothesis.

4.4 Volunteers' Perceptions of Paid Developers

To address RQ4, we solicited volunteers' perspectives on paid developers' contributions in terms of the five hypotheses (see Sec. 2). Tables 4 to 8 present the results, where 'M + N' in the first formula represents the number of respondents who selected 'strongly agree' and 'agree' (and similarly, 'strongly disagree' and 'disagree'), respectively. We group the reasons for 'strongly agree' and 'agree' together, and the same for 'strongly disagree' and 'disagree.' A single response could include multiple reasons, which is why the sum of reasons can be higher than the number of respondents per option.

Table 4 shows the synthesized reasons for respondents' level of agreement with *H1*. More than half (33, 68.8%) of respondents agreed with *H1*, though reasons varied. Only five respondents (10.4%) indicated disagreement towards *H1*, and ten respondents remained neutral. For the responses supporting *H1*, *"having more time"* (including both working hours and spare time) is the most common reason. Further, respondents also pointed out that clear goals and obligations of paid developers force them to contribute more frequently. Developers holding opposing or neutral views indicated that task difficulty, personal choice, passions, and experiences can also affect developers' contribution frequency. The quantitative analysis showed that being paid only has a significant difference when developers are core members of Rust. This may indicate that only core paid developers can fully work on Rust, while this is not the case for most paid developers in the periphery. However, 68.8% of respondents agreed with *H1*, suggesting that most volunteers have a higher expectation of paid developers' contribution frequency.

Table 5 shows the synthesized reasons for the respondents' level of agreement with *H2*. Twenty-one (43.8%) respondents agree with *H2*, the same number of respondents hold a neutral stance, and six (12.5%) respondents disagree with *H2*. The most mentioned supportive reason (n=15) for *H2* is similar to *H1*: paid developers *have more time* to prepare big code changes. Six respondents (four 'agree' and two 'neutral') pointed out that paid developers are assigned to work on specific features, which usually end up with large code dumps. The most common reasons for the 'Neutral' and 'Disagree' options are the same, i.e., the scale of commits is mainly determined by developers' personal style. One respondent indicated that paid developers tend to make more small changes. The quantitative analysis indicated that only peripheral paid developers tend to contribute a bit larger code changes to OSS projects than peripheral volunteers. This indicates that core paid developers are suffering from an undeserved stereotype of contributing large chunks of code from over 40% of volunteers.

Table 6 shows volunteers' perspectives towards *H3*, namely that paid developers focus primarily on adding new features. Most (n=22, 45.8%) respondents remained neutral, 14 (29.2%) respondents indicated agreement, and 12 (25.0%) disagreed with this. For those agreeing, respondents pointed out that implementing features will gain more recognition from an employer than other types of contributions, such as writing documentation, and companies' needs from OSS projects are usually adding specific features. Features usually require considerable time to be designed, tested, and implemented. *"Having more time"* is another reason (3 'Agree' and 1 'Neutral' mentioned this). These reasons may explain the quantitative results for RQ1, i.e., why peripheral paid developers are more inclined to contribute features when compared with volunteers. Eight respondents who selected 'Neutral' or 'Disagree' held the view that, whether or not paid developers prefer implementing features *"varied with assignments"*: some are paid to work on Rust in a way they see fit; others are paid to implement specific features. This reason was also mentioned in their perspectives of *H1* and *H2*. Four volunteers who disagreed with *H3* believe that paid developers tend to *"do boring work nor interesting features,"* which also shows some volunteers' prejudice against paid developers.

We also collected and analyzed developers' perspectives on *H4* related to paid developers' collaborations with volunteers. Most (n=26, 54.2%) respondents disagree with *H4*, i.e., paid developers tend

Table 4. Volunteers' responses to *H1*: Paid developers contribute more frequently than volunteers

Option	Reasons	n
Agree (33=12+21, 68.8%)	Have more time to contribute	22
	Clear goals reduce time to determine tasks	8
	Obligation forces frequent contributions	5
	No explanation	7
Neutral (10, 20.8%)	Depend on various factors	2
	Mostly do proprietary projects	1
	No explanation	7
Disagree (5=1+4, 10.4%)	Mostly do proprietary projects	2
	Varies with assignment type	1
	Personal choice	1
	Depend on task difficulty	1
	Paid developers tend to be more senior, do more management	1

Table 5. Volunteers' responses to H2: Paid developers may contribute larger chunks of code in commits.

Option	Reasons	n
Agree (21=7+14, 43.8%)	Have more time to prepare big code changes	15
	Adding features requires large code changes	4
	No explanation	5
Neutral (21, 43.8%)	Personal choice	7
	Adding features requires large code changes	2
	Have limited time to prepare large changes	1
	Depends on projects	1
	No explanation	11
Disagree (6=1+5, 12.5%)	Personal choice	2
	Varies with assignment type	1
	Do more small changes	1
	No explanation	2

Table 6. Volunteers' responses to H3: Paid developers are more likely to contribute features than volunteers.

Option	Reasons	n
Agree (14=2+12, 29.2%)	Implementing features is more fruitful	4
	Company's needs are new features	4
	Have more time to implement features	3
	No explanation	4
Neutral (22, 45.8%)	Varies with assignment type	5
	Have no preference	3
	Personal choice	3
	Have more time to implement features	1
	No explanation	10
Disagree (12=7+5, 25.0%)	Do boring work nor interesting features	4
	Varies with assignment type	4
	No explanation	5

to collaborate less with volunteers. We categorized the collected reasons into three types: (1) Eight respondents (16.7%) believe that paid developers have no preference for either type of contributor to collaborate with. A developer's techniques and skills may determine who will be their collaborators. (2) Seven respondents indicated that *H4* does not correspond with their experience in Rust because they see volunteers and paid developers collaborate together. (3) Two respondents believe that the Rust community supports collaboration between volunteers and paid developers. Nine respondents did not provide any explanation. Twelve respondents (25%) agreed with *H4*. The most common reasons for agreement are that paid developers prefer to collaborate with developers from the same company, and they have no interest in the type of work that volunteers primarily do in Rust, such as documentation and maintenance. Ten respondents remained neutral; seven of those did not provide any explanations. Combined with the quantitative results indicating that whether paid

Table 7. Volunteers' responses to H4: Paid developers tend to collaborate less with volunteers.

Option	Reasons	n
Agree (12=2+10, 25.0%)	Prefer collaborate with developers from the same company	5
	Not interested in what volunteers mainly do	3
	Work is not attractive to others	1
	Dayjob hampers collaboration	1
	No explanation	2
Neutral (10, 20.8%)	Varies with assignment type	1
	Happen in mentoring situations	1
	See volunteers and paid developers collaborate together	1
	No explanation	7
Disagree (26=16+10, 54.2%)	Have no collaboration preference	8
	See volunteers and paid developers collaborate together	7
	Community is supportive of collaboration	2
	No explanation	9

Table 8. Volunteers' responses to H5: Paid developers are less likely to become long-term contributors.

Option	Reasons	n
Agree (7=0+7, 14.6%)	Lack personal attachment	3
	Withdrawal after goal achievement	2
	Contribute less if unpaid	2
	No explanation	1
Neutral (26, 54.2%)	Becoming LTCs is hard for both	4
	Vary with assignment type	3
	LTCs first then being paid	2
	Contribute less if unpaid	1
	No explanation	16
Disagree (15=9+6, 31.3%)	Being paid ensures long term	4
	LTCs first then being paid	2
	No explanation	9

developers are peripheral or core to Rust, they collaborate more with volunteers, we can observe that certain volunteers in the Rust community exhibit some bias against paid developers.

Table 8 shows respondents' categorized perspectives towards their agreements of *H5*, that is, paid developers are less likely to become long-term contributors. Only seven (14.6%) respondents indicated agreement, and their reasons were in line with our hypothesis: companies may withdraw from OSS projects once their business goal has been achieved, or changes; paid developers may lack personal attachment to the OSS projects and may become less active (or even disappear) when they are no longer paid. More than half of (n=26, 54.2%) respondents held a neutral stance. Fifteen respondents (31.3%) disagreed with *H5*: (1) Seven respondents simply indicated this hypothesis contradicted their experience in Rust (or other OSS projects). (2) Four mentioned that being paid ensures long-term contributions because of secure income. (3) Two volunteers explained that developers are usually long-term contributors before being paid by companies. The results of the regression analysis in RQ3 show that being paid is a significantly positive factor in becoming a long-term contributor. Reasons such as having a secure income or already being a long-term contributor may explain the modeling results (see Table 8). Over half of the neutral responses indicate the need to further study the relationship between being paid and developers' long-term participation.

Summary for RQ4: Hypothesis *H1* is supported by the majority of respondents: almost 70% of survey respondents agree that paid developers contribute more frequently than volunteers. Key reasons are that paid developers have more time, have clear goals, and do so because they are paid. *H2* is not clearly supported, as Ca. 44% of respondents believe that paid developers contribute larger commits, while ca. 44% is unsure, and the remaining 12.5% disagree. *H3* is largely unsupported, with approx. 70% of volunteers were unsure or disagreed that paid developers focus primarily on adding features. The same for *H4* and *H5*: more than half of the respondents believe paid developers and volunteers collaborate well in Rust; Only 14.6% of volunteers agreed that paid developers are *less* likely to become long-term contributors, ca. 30% disagreed, and over 54% were unsure.

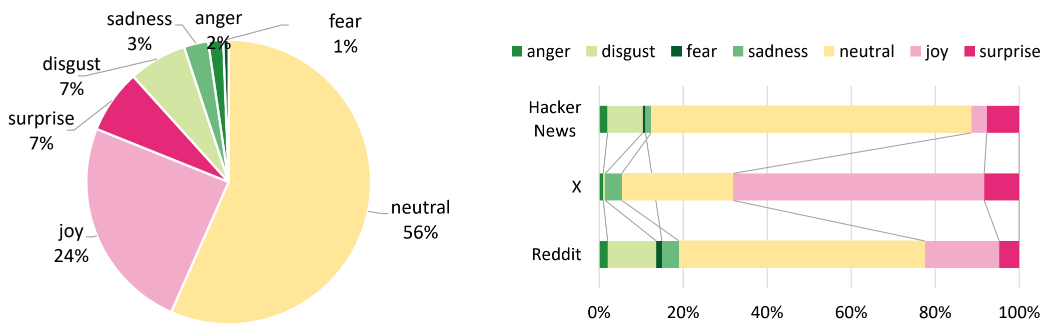
4.5 Community Emotions When Volunteers Become Paid Developers

To explore how the Rust community reacts when volunteers transition to paid developers, we retrieved 650 comments from 12 relevant posts, where the numbers of comments range from 6 to 166, and conducted emotion analysis, as introduced in Sec. 3.7. Figure 7 shows the distribution of the seven identified emotions in the community discussions about Rust volunteers becoming paid developers. As Fig. 7a shows, 'neutral' is the most common emotion (56%). The emotions 'joy' and 'surprise' account for 31% of the comments. In contrast, we found 13% of the comments express negative emotions, including 'disgust' (7%), 'sadness' (3%), 'anger' (2%), and 'fear' (1%). While relatively limited in proportion, this does show that some people are concerned about volunteers being hired by companies to contribute to Rust. The 24 volunteers who subsequently transitioned to paid positions contributed an average of 142 features, far exceeding the overall average of 11 among all Rust contributors, and 22 of them were core developers. Their strong potential to shape the project's roadmap may partly explain the negative emotions observed within the community.

When we consider the distribution of emotions expressed across the three different platforms (i.e., X (formerly Twitter), Reddit, and Hacker News), the results are a bit more nuanced (see Figure 7b). The most common emotions are 'neutral' in the discussions posted on the platforms Reddit and Hacker News. Different from comments in posts published on X, the proportions of negative comments (including 'anger,' 'disgust,' 'fear,' and 'sadness') on Reddit and Hacker News are comparable to the percentages of positive emotions, respectively. Specifically, 19% (28 out of 147) and 12% (37 out of 299) of comments posted on the platforms Reddit and Hacker News express concern about volunteers being hired by companies to contribute to Rust. For example, one person commented:

> "It's great that AWS wants to have the rust team onboard, but in my experience, at least AWS has been good at leeching Open Source, rather than fostering it. Open source is not part of their culture."

On X, the dominant emotion is joy (60% (122 out of 204 comments). Upon further investigation of these comments, we found that the two posts on X are two Rust contributors' announcements of joining companies to work on Rust, and most of the 'joy' comments are like "Welcome to the company" from new colleagues and 'congratulations' with a high spirit, such as *"Tremendously* *exciting, congratulations!"* The emotion differences between X and the other two platforms may lie in their characteristics [4]: X is a social media for sharing short updates, opinions, and news to the general public and has a diverse user base; Reddit and Hacker News are for community-driven discussions across various topics and their users are usually from tech-savvy communities focused on specific interests. Specifically, all seven posts from the Reddit platform are from the 'r/rust' subreddit, which is dedicated to discussing things related to the Rust programming language [76]. In the social media platform, people may be more likely to say something nice, while in the technical forums, same-interest-driven users are more likely to have in-depth discussions.



(a) Overall emotion distribution. (b) Emotion distributions in the three sources.

Fig. 7. Emotion distributions in the discussions when volunteers become being paid in Rust

Summary for RQ5: Overall, the main emotion of comments on Rust volunteers becoming paid by companies is neutral, and approximately 13% of comments express negative emotions, showing that some people have concerns about commercial involvement in Rust. The developers who transitioned to paid roles were highly active and influential contributors, which may explain the negative emotions within the community. However, while most comments on Reddit and Hacker News are neutral, on X emotions are primarily joyful, which may be caused by their social characteristics.

5 Discussion

This study presents a number of key findings. This section discusses the implications of our findings.

5.1 Community Governing Policies and Interface Design

This study presents the first comparative analysis of paid developers and volunteers within a single project at a fine level of granularity. Understanding differences between paid and volunteer contributors can help OSS communities to design better governing policies, and interaction and collaboration interfaces to support the sustainability of OSS projects. For example, OSS communities could conduct real-time measurements of the scale and types of commits contributed by peripheral paid developers. When identifying paid developers who are mainly submitting large features, OSS communities could emphasize the need for long-term maintenance and the value of making other types of contributions. For volunteers, a lack of sustained time and stable income is perhaps the most pressing barrier that keeps them from becoming long-term contributors. Therefore, OSS communities (and/or hosting platforms) could provide an interface for volunteers who are looking for companies to support them financially (or employ them) to make contributions to projects and promote the interface to companies.

5.2 Volunteers' Perspectives on Paid Developers

The combined quantitative and qualitative results suggest that many volunteers might have some 'prejudice' against paid developers, such as *"[they] do boring work since [they're] being paid," "[they] rarely care [for] documentation,"* and *"[they] lack personal attachment."* An unfamiliarity among developers in OSS projects may cause frustration and possibly conflict between paid developers and volunteers, jeopardizing a project's sustainability. Moreover, we also found obvious negative emotions, such as 'disgust,' 'anger,' and 'fear,' when community members see volunteers become paid developers. However, we estimated a panel fixed-effects model on 24 developers who transitioned from volunteer to paid status, most of whom were core contributors. We compared their contribution frequency, change size, task preference, and collaboration likelihood before and after the transition. The results show no statistically significant differences across these measures, suggesting that becoming paid does not systematically alter developers' contribution characteristics.[9] To mitigate possible volunteers' misunderstandings against paid developers, OSS communities could provide dashboards to visualize companies' contributions in real time transparently. The dimensions studied in this article (contribution frequency, change size, task type, collaboration, and likelihood of becoming a LTC) provide a set of indicators that could be measured.

5.3 Harmonizing Company Participation in Open Source Communities

As suggested in prior work [41, 104] as well as the survey presented in this article, there is a perception that companies tend to develop big features in-house that are subsequently contributed to OSS projects. Our quantitative analysis confirms this; peripheral paid developers tend to focus more on implementing features than do volunteers. However, such contributions require considerable effort in terms of peer review, adding workload to maintainers of OSS projects who may already face a considerable workload. Features may directly affect the roadmap of an OSS project. Core volunteer developers may perceive these contributions as a sign of detachment and lack of 'care' for the project, which could be one reason that corporate participation is disliked by some volunteers [109]. Rust, in particular, reportedly had several core developers who left because of the participation of Amazon.com [52, 102]. Thus, companies should become more sensitive to the needs and norms of OSS projects, as well as the volunteers in the project, to avoid being perceived as a commercial 'parasite.' The behavior of even a single paid developer may affect how companies are perceived by volunteers and the wider OSS community. Specifically, the results of RQ2 indicate that although paid developers seem to collaborate more frequently with volunteers at a global level, their micro-level collaboration intensity is weaker than expected. This pattern suggests a latent divide between paid and volunteer contributors. Companies should encourage their employees to actively engage with volunteers beyond organizational boundaries in an OSS ecosystem. The dashboard mentioned above can be used as a mirror to inform their OSS strategies and adjust their contributions.

The proportion of paid developers in the core group is ca. 20%, higher than in the peripheral group, which means companies can be recognized in the Rust project and play a significant role in its development. Assessing whether a company's commercial interests align with an OSS project's roadmap should be the first step before joining. We suggest that hiring volunteers who are already contributing to a project to implement a company's objectives might be a more appropriate and convenient solution because they may be better able to balance the community's long-term concerns and the company's business objectives. Nevertheless, companies should pay attention to public

[9]Detailed results can be found in our online appendix [108]

opinion toward the volunteer-to-paid transitions. The results of RQ5 show that approximately 13% of comments on this phenomenon are negative. Since developers who became paid contributors were among the most active and visible members, transitions of such key figures should be handled carefully to maintain trust and prevent community tension. We suggest companies solicit OSS communities' concerns about their joining and try to address them as much as possible. Moreover, the different emotion distributions of Reddit/Hacker News/X indicate that community reactions vary across platforms. Companies and OSS maintainers should tailor their communication strategies to the norms and expectations of each social space.

5.4 Implication for Research

It is clear that differences exist across different groups of contributors to an OSS project. This study sought to go beyond previous characterizations of paid vs. volunteer developers, offering a more fine-grained analysis. Our results suggest not all paid developers are the same; for example, some are tasked to implement certain required features, and others make contributions as they see fit. The latter group may be similar to volunteers but make more frequent contributions, i.e., core paid developers. This study suggests that the dichotomous characterization of contributors as either paid or volunteer is too simplistic and does not fully match reality, and that further subcategories could be identified. Given the important role of OSS in today's IT landscape, it is imperative to verify these hypotheses across more types of OSS projects, such as single vendor open source projects [78] and from other perspectives. One likely perspective appears to be whether developers are ambitious to pursue a career with the community vs. a career with their company [85]. Increased awareness of subgroups can support OSS projects in building harmonious relationships between different groups of developers.

6 Threats to Validity

The design and execution of the current study are the result of a number of trade-offs [81], which we discuss in this section. These trade-offs should be considered while interpreting the results, and can also be considered in designing future studies.

External validity. This study focused specifically on the Rust programming language project; the scope of the findings is therefore limited to this particular context. We decided to study Rust to allow for an in-depth comparison between paid and volunteer developers who share the same project context, including corporate involvement and programming language. Future work could extend this analysis to additional projects within the Rust ecosystem (e.g., ecosystem tools or popular crates) and across other large-scale OSS systems (e.g., the Linux kernel) to evaluate the generalizability of the observed patterns and uncover potential variations in corporate participation and community dynamics.

Another potential threat is the representativeness of our developer sample. Respondents might be unfamiliar with other paid developers. In the survey, we included a neutral option that provides respondents with the opportunity to express a lack of opinion or indifference. This could explain why 'Neutral' was the most common answer to *H3* and *H4*. The results from the survey might be biased toward what developers think they know about paid developers. Moreover, we found some inconsistencies between quantitative and qualitative responses from developers who remain neutral in terms of our hypotheses. This may reflect nuanced or context-specific views. We prioritized quantitative responses for consistency but acknowledge this as a limitation. Future work could investigate such discrepancies through follow-up interviews or more focused qualitative studies. The resignation of the entire moderation team and some core developers leaving Rust attracted considerable attention, and some companies' dominating involvement has been one of the most discussed causes [52, 92, 102]. Therefore, the issue of paid developers may loom larger in the Rust community than in other OSS projects.

The survey received a total of 53 responses (response rate 23.2%); this number and response rate are similar to other studies of OSS developers [75, 90, 111]. The survey results complement the quantitative results answering RQ1, RQ2, and RQ3. However, the survey goal was not to generalize across the Rust project, and so the findings should not be interpreted as such.

Construct validity. We measured developer contribution through commits, as this represents the key activity in software development. We acknowledge, as did prior studies [105, 109, 116], that different data sources can be used, including issue reports, code reviews, and online discussions. We decided to use commit data only because the cleaning of data and accurately attributing it to the right group (paid, volunteers) is very time-consuming, as it includes a manual check and verification; we obtained an accuracy of 94.3%. Triangulating across other data sources than commit data remains an open challenge for future work. Besides, since the initial experiments were conducted in 2022, we used data up to the end of 2021 to align with the survey period (RQ4). We later extended the dataset to June 30, 2025 and validated *H1–H5*, finding consistent results except for *H1* (contribution frequency) in the core group, likely because of Rust's increasing maturity, which may have naturally slowed developers' contribution pace.

A second potential threat lies in distinguishing paid and volunteer developers, which remains an open research challenge. A developer with a public email address could also be paid. On the other hand, using the same heuristic of checking email address domains, a developer who is paid by a company but makes contributions to Rust on their own, may be classified as a paid developer. However, based on the 53 responses, no developers whom we identified as volunteers, indicated they were paid, and only one paid developer indicated their employer is another company. Thus, we deem this threat would have a limited impact, if at all, on the results.

Another decision was our definition of long-term contributors, which required developers to have contributed at least three years to Rust. This definition was based on prior studies [112, 114]. While this definition required the exclusion of over 1,500 developers (out of 4,117), we argue that the results remain sufficiently representative. Further, there are other factors, such as social capital [75] and the specific features of companies that employ paid developers, which could also affect the likelihood of becoming long-term contributors. In this study, we only considered developers' ability and willingness as control variables, leaving other factors to be studied through future studies.

There are various ways in which developers collaborate in OSS projects. In this study, we adopted co-editing the same file as the primary measure of collaboration, as [62] found that this activity aligns well with developers' own perceptions of collaborative work. To validate the findings of RQ2, we further conducted comparison experiments using a dialogue-based collaboration metric derived from pull request (PR) interactions, where two developers are considered to have collaborated if they both commented on the same PR. The results based on PR comments were consistent with the findings of RQ2, supporting the robustness of our approach. Detailed results and analyses are provided in our online appendix [108].

Besides the five dimensions we have explored, paid developers can also be different from volunteers in other aspects. For instance, are contributions from paid developers more likely to be accepted or rejected in Rust? Future studies could conduct more thorough comparisons between paid developers and volunteers to benefit from a better understanding of the governance framework of OSS contributors.

By addressing RQ5, we aim to convey how the Rust community reacted when volunteers were hired by companies. Although we treat people who make any kind of contributions to Rust as a Rust community member, users giving comments on different platforms may come from other social networks, such as friends, relatives, or colleagues, who may hardly know Rust. Further, given that each platform fosters distinct social norms and audiences, it is plausible that developers adopt different emotional tones depending on the context. Due to the diversity of accounts and behaviors in different platforms, it is extremely difficult to distinguish Rust community members from others or match cross-platform identities. We analyzed the emotion distributions per platform to draw more precise conclusions. Future work can delve into the roles of these commenters toward Rust and explore whether the same developers react differently on different platforms. Besides, the role transition of core developers tends to bring discussions. It indicates that the comments collected and analyzed to address RQ5 may be biased toward core developers and do not represent the views of general volunteers being hired.

7 Conclusion

This article presents an empirical comparison between paid developers and volunteers in the Rust community. We find that paid developers' characteristics differ from volunteers in several dimensions. Peripheral paid developers tend to contribute more frequently than volunteers and also have a stronger focus on features. No matter how senior paid developers are, they tend to collaborate more with volunteers but less than expected. Being paid is a positive factor in becoming a long-term contributor. Some volunteers hold a skeptical attitude toward paid developers' contributions in Rust, which can also be reflected by negative emotions when discussing the volunteer-to-paid transitions. Overall, paid and volunteer developers each have their characteristics, and core paid developers may work beyond their duties.

Acknowledgments

This work is supported by the National Natural Science Foundation of China Grant (62572048, 62202048, 62332001, and 62232003), Research Ireland grant 13/RC/2094-P2 to Lero.

References

[1] Adam Alami, Marisa Leavitt Cohn, and Andrzej Wąsowski. 2019. Why does code review work for open source software communities?. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1073–1083.

[2] Sadika Amreen, Audris Mockus, Russell Zaretzki, Christopher Bogart, and Yuxia Zhang. 2020. ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems. *Empir. Softw. Eng.* 25, 2 (2020), 1136–1167. doi:10.1007/s10664-019-09786-7

[3] Tim Anderson. 2021. Rust dust-up as entire moderation team resigns. Why? They won't really say. https://www.theregister.com/2021/11/23/rust_moderation_team_quits/.

[4] Maurício Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurélio Gerosa. 2018. How modern news aggregators help development communities shape and share knowledge. In *Proceedings of the 40th International conference on software engineering*. 499–510.

[5] Ann Barcomb, Andreas Kaufmann, Dirk Riehle, Klaas-Jan Stol, and Brian Fitzgerald. 2020. Uncovering the Periphery: A Qualitative Survey of Episodic Volunteering in Free/Libre and Open Source Software Communities. *IEEE Transactions on Software Engineering* 46, 9 (2020).

[6] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[7] Evangelia Berdou. 2006. Insiders and outsiders: paid contributors and the dynamics of cooperation in community led F/OS projects. In *IFIP International Conference on Open Source Systems*. Springer, 201–208.

[8] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. 2006. Mining email social networks. In *International Workshop on Mining Software Repositories*. ACM, 137–143.

[9] Matthew Broersma. 2006. Debian Delayed by Fund-Raising Fracas. https://www.cio.com/article/265170/linux-debian-delayed-by-fund-raising-fracas.html.

[10] Sabur Butt, Shakshi Sharma, Rajesh Sharma, Grigori Sidorov, and Alexander Gelbukh. 2022. What goes on inside rumour and non-rumour tweets and their reactions: A psycholinguistic analyses. *Computers in Human Behavior* 135 (2022), 107345.

[11] Fabio Calefato, Marco Aurelio Gerosa, Giuseppe Iaffaldano, Filippo Lanubile, and Igor Steinmacher. 2022. Will you come back to contribute? Investigating the inactivity of OSS core developers in GitHub. *Empirical Software*

Engineering 27, 3 (2022), 76.

[12] Andrea Capiluppi, Klaas-Jan Stol, and Cornelia Boldyreff. 2012. Exploring the role of commercial stakeholders in open source software evolution. In *IFIP International Conference on Open Source Systems*. Springer, 178–200.

[13] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. 2015. Developer onboarding in GitHub: the role of prior social links and language experience. In *10th Joint Meeting on Foundations of Software Engineering*. ACM, 817–828.

[14] Zhifei Chen, Wanwangying Ma, Lin Chen, and Wei Song. 2022. Collaboration in software ecosystems: A study of work groups in open environment. *Information and Software Technology* 145 (2022), 106849.

[15] Maëlick Claes, Mika Mäntylä, Miikka Kuutila, and Umar Farooq. 2018. Towards automatically identifying paid open source developers. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 437–441.

[16] Maëlick Claes, Mika Mäntylä, Miikka Kuutila, and Bram Adams. 2017. Abnormal Working Hours: Effect of Rapid Releases and Implications to Work Content. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 243–247. doi:10.1109/MSR.2017.3

[17] Jailton Coelho, Marco Tulio Valente, Luciana L. Silva, and André Hora. 2018. Why We Engage in FLOSS: Answers from Core Developers. In *11th International Workshop on Cooperative and Human Aspects of Software Engineering* (Gothenburg, Sweden) *(CHASE '18)*. ACM, New York, NY, USA, 114–121. doi:10.1145/3195836.3195848

[18] Rust Community. [n. d.]. The Rust Programming Language. https://github.com/rust-lang/rust.

[19] Requests Community. 2024. Requests: HTTP for Humans. https://requests.readthedocs.io/en/latest/.

[20] Stack Overflow Community. 2022. 2022 Developer Survey. https://survey.stackoverflow.co/2022#section-most-loved-dreaded-and-wanted-programming-scripting-and-markup-languages.

[21] Jonathan Corbet and Greg Kroah-Hartman. 2016. Linux Kernel Development: How fast it is going, who is doing it, what they are doing, and who is sponsoring it? (25th Anniversary Edition). *Linux Foundation Whitepaper* (August 2016).

[22] Jonathan Corbet, Greg Kroah-Hartman, and Amanda McPherson. 2012. Linux kernel development. How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It. *The Linux Foundation* (2012), 1–13.

[23] Sarah Crowe, Kathrin Cresswell, Ann Robertson, Guro Huby, Anthony Avery, and Aziz Sheikh. 2011. The case study approach. *BMC medical research methodology* 11, 1 (2011), 1–9.

[24] Tapajit Dey, Bogdan Vasilescu, and Audris Mockus. 2020. An exploratory study of bot commits. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. ACM, 61–65. doi:10.1145/3387940.3391502

[25] Luiz Felipe Dias, Caio Barbosa, Gustavo Pinto, Igor Steinmacher, Baldoino Fonseca, Márcio Ribeiro, Christoph Treude, and Daniel Alencar da Costa. 2020. Refactoring from 9 to 5? What and When Employees and Volunteers Contribute to OSS. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.

[26] Luis Felipe Dias, Igor Steinmacher, and Gustavo Pinto. 2018. Who drives company-owned OSS projects: internal or external members? *Journal of the Brazilian Computer Society* 24, 1 (Dec. 2018). doi:10.1186/s13173-018-0079-x

[27] Geanderson Esteves dos Santos and Eduardo Figueiredo. 2020. Commit Classification using Natural Language Processing: Experiments over Labeled Datasets. In *CIbSE*.

[28] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*. Springer, 285–311.

[29] Paul Ekman. 1999. Basic emotions. In *Handbook of Cognition and Emotion*, Tim Dalgleish and Mick J. Power (Eds.). 45–60.

[30] Damien R Farine. 2017. A guide to null models for animal social network analysis. *Methods in Ecology and Evolution* 8, 10 (2017), 1309–1320.

[31] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C Murphy, and Jean-Rémy Falleri. 2015. Impact of developer turnover on quality in open-source software. In *2015 10th Joint Meeting on Foundations of Software Engineering*. 829–841.

[32] Catherine O. Fritz, Peter E. Morris, and Jennifer J. Richler. 2012. Effect size estimates: current use, calculations, and interpretation. *Journal of Experimental Psychology: General* 141, 1 (2012), 2–18.

[33] James H. Gerlach, Chorng-Guang Wu, Lawrence F Cunningham, and Clifford E Young. 2016. An Exploratory Study of Conflict over Paying Debian Developers. *International Journal of Open Source Software and Processes* 7, 3 (2016), 20–38.

[34] Daniel M German. 2002. The evolution of the GNOME Project. In *Proceedings of the 2nd Workshop on Open Source Software Engineering*. 20–24.

[35] Daniel M German. 2003. The GNOME project: a case study of open source, global software development. *Software Process: Improvement and Practice* 8, 4 (2003), 201–215.

[36] Mathieu Goeminne and Tom Mens. 2011. Evidence for the pareto principle in open source software activity. In *the Joint Porceedings of the 1st International workshop on Model Driven Software Maintenance and 5th International Workshop on Software Quality and Maintainability*. 74–82.

[37] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un) happy. *Journal of Systems and Software* 140 (2018), 32–47.

[38] Alexander Hars and Shaosong Ou. 2002. Working for Free? Motivations of participating in open source projects. *International Journal of Electronic Commerce* 6, 3 (2002), 25–39.

[39] Jochen Hartmann. 2022. Emotion English DistilRoBERTa-base. https://huggingface.co/j-hartmann/emotion-english-distilroberta-base/.

[40] Øyvind Hauge, Claudia Ayala, and Reidar Conradi. 2010. Adoption of open source software in software-intensive organizations–A systematic literature review. *Information and Software Technology* 52, 11 (2010), 1133–1154.

[41] Joachim Henkel. 2006. Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy* 35, 7 (2006), 953–969.

[42] James Herbsleb, Christian Kästner, and Christopher Bogart. 2016. Intelligently Transparent Software Ecosystems. *IEEE Software* 33, 1 (2016), 89–96.

[43] Guido Hertel, Sven Niedner, and Stefanie Herrmann. 2003. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 32, 7 (2003), 1159 – 1177.

[44] Joseph M. Hilbe. 2015. *Practical Guide to Logistic Regression*. CRC Press.

[45] Giuseppe Iaffaldano, Igor Steinmacher, Fabio Calefato, Marco Gerosa, and Filippo Lanubile. 2019. Why Do Developers Take Breaks from Contributing to OSS Projects? A Preliminary Analysis. In *2nd International Workshop on Software Health* (Montreal, Quebec, Canada) *(SoHeal '19)*. IEEE Press, 9–16. doi:10.1109/SoHeal.2019.00009

[46] Mitchell Joblin, Sven Apel, Claus Hunsen, and Wolfgang Mauerer. 2017. Classifying developers into core and peripheral: An empirical study on count and network metrics. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 164–174.

[47] Corbet Jonathan and Kroah-Hartman Greg. 2017. 2017 Linux Kernel Development Report. https://www.linuxfoundation.org/2017-linux-kernel-report-landing-page/.

[48] Brian T Jones. 2018. free_email_provider_domains.txt. https://gist.github.com/tbrianjones/5992856/, last accessed 20 Aug 2019.

[49] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. 2015. Likert scale: Explored and explained. *British Journal of Applied Science & Technology* 7, 4 (2015), 396.

[50] Nicolas Jullien, Klaas-Jan Stol, and James Herbsleb. 2019. *A Preliminary Theory for Open Source Ecosystem Micro-economics*. Springer, 49–68.

[51] Noureddine Kerzazi and Ikram El Asri. 2016. Who can help to review this piece of code?. In *Collaboration in a Hyperconnected World: 17th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2016, Porto, Portugal, October 3-5, 2016, Proceedings 17*. Springer, 289–301.

[52] Steve Klabnik. 2021. I refuse to let Amazon define Rust. https://twitter.com/steveklabnik/status/1437441662998007809.

[53] Carsten Kolassa, Dirk Riehle, and Michel A. Salim. 2013. A Model of the Commit Size Distribution of Open Source. In *SOFSEM 2013: Theory and Practice of Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 52–66.

[54] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark GJ van den Brand. 2012. Who's who in Gnome: Using LSA to merge software repository identities. In *28th IEEE International Conference on Software Maintenance*. 592–595.

[55] Karim Lakhani and Robert Wolf. 2005. *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*. MIT Press, Cambridge.

[56] Amanda Lee and Jeffery C Carver. 2017. Are One-Time Contributors Different? A Comparison to Core and Periphery Developers in FLOSS Repositories. In *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 1–10.

[57] Amanda Lee, Jeffrey C Carver, and Amiangshu Bosu. 2017. Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 187–197.

[58] Bin Lin, Gregorio Robles, and Alexander Serebrenik. 2017. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In *IEEE 12th International Conference on Global Software Engineering*. 66–75.

[59] Matthieum. 2022. Moderation Team Resignation. https://www.reddit.com/r/rust/comments/qzme1z/moderation_team_resignation/.

[60] Patrick E. McKnight and Julius Najab. 2010. *Mann-Whitney U Test*. John Wiley and Sons, Ltd, 1–1. doi:10.1002/9780470479216.corpsy0524

[61] Andrew Meneely, Mackenzie Corcoran, and Laurie Williams. 2010. Improving developer activity metrics with issue tracking annotations. In *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics*. 75–80.

[62] Andrew Meneely and Laurie Williams. 2011. Socio-technical developer networks: Should we trust our measurements?. In *Proceedings of the 33rd international conference on software engineering*. 281–290.

[63] Andrew Meneely and Laurie A. Williams. 2009. Secure open source collaboration: an empirical study of Linus' Law. In *ACM 2009 Conference on Computer and Communications Security*.

[64] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why do people give up flossing? a study of contributor disengagement in open source. In *Open Source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019, Montreal, QC, Canada, May 26–27, 2019, Proceedings 15*. Springer, 116–129.

[65] Audris Mockus, Roy T Fielding, and James D Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *Acm Transactions on Software Engineering Methodology* 11, 3 (2002), 309–346.

[66] E. Moon and J. Howison. 2014. Modularity and organizational dynamics in open source software (OSS) production. *20th Americas Conference on Information Systems, AMCIS 2014* (01 2014).

[67] Geoffrey Stewart Morrison. 2013. Tutorial on logistic-regression calibration and fusion:converting a score to a likelihood ratio. *Australian Journal of Forensic Sciences* 45, 2 (2013), 173–197. doi:10.1080/00450618.2012.733025

[68] Nicole Novielli and Alexander Serebrenik. 2019. Sentiment and emotion in software engineering. *IEEE Software* 36, 5 (2019), 6–23.

[69] Cassandra Overney, Jens Meinicke, Christian Kästner, and Bogdan Vasilescu. 2020. How to not get rich: An empirical study of donations in open source. In *ACM/IEEE 42nd international conference on software engineering*. 1209–1221.

[70] Mathieu O'Neil, Xiaolan Cai, Laure Muselli, Fred Pailler, and Stefano Zacchiroli. 2021. *The coproduction of open source software by volunteers and big tech firms*. News and Media Research Centre.

[71] Mathieu O'neil, Laure Muselli, Mahin Raissi, and Stefano Zacchiroli. 2021. 'Open source has won and lost the war': Legitimising commercial–communal hybridisation in a FOSS project. *New Media & Society* 23, 5 (2021), 1157–1180.

[72] Charlie Parker, Sam Scott, and Alistair Geddes. 2019. Snowball sampling. *SAGE research methods foundations* (2019).

[73] Gustavo Pinto, Luiz Felipe Dias, and Igor Steinmacher. 2018. Who gets a patch accepted first?: comparing the contributions of employees and volunteers. In *11th International Workshop on Cooperative and Human Aspects of Software Engineering*. ACM, 110–113.

[74] Oleksandra Poquet, Liubov Tupikina, and Marc Santolini. 2020. Are forum networks social networks? A methodological perspective. In *Proceedings of the tenth international conference on learning analytics & knowledge*. 366–375.

[75] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. 2019. Going farther together: The impact of social capital on sustained participation in open source. In *IEEE/ACM 41st International Conference on Software Engineering*. IEEE, 688–699.

[76] Raddit. 2024. The Rust Programming Language. https://www.reddit.com/r/rust/.

[77] Eric Raymond. 1999. The cathedral and the bazaar. *Knowledge, Technology & Policy* 12, 3 (1999), 23–49.

[78] Dirk Riehle. 2020. Single-vendor open source firms. *Computer* 53, 4 (2020), 68–72.

[79] Dirk Riehle, Philipp Riemer, Carsten Kolassa, and Michael Schmidt. 2014. Paid vs. Volunteer Work in Open Source. In *2014 47th Hawaii International Conference on System Sciences*. 3286–3295. doi:10.1109/HICSS.2014.407

[80] Peter C. Rigby, Daniel M. German, and Margaret-Anne Storey. 2008. Open Source Software Peer Review Practices: A Case Study of the Apache Server. In *30th International Conference on Software Engineering* (Leipzig, Germany) *(ICSE '08)*. ACM, New York, NY, USA, 541–550. doi:10.1145/1368088.1368162

[81] Martin Robillard, Deeksha Arya, Neil Ernst, Jin L.C. Guo, Maxime Lamothe, Mathieu Nassif, Nicole Novielli, Alexander Serebrenik, Igor Steinmacher, and Klaas-Jan Stol. 2024. Communicating Study Design Trade-offs in Software Engineering. *ACM Transactions on Software Engineering and Methodology* in press (2024).

[82] Gregorio Robles, Andrea Capiluppi, Jesus M Gonzalez-Barahona, Björn Lundell, and Jonas Gamalielsson. 2022. Development effort estimation in free/open source software from activity in version control systems. *Empirical Software Engineering* 27, 6 (2022), 135.

[83] David Rozado, Ruth Hughes, and Jamin Halberstadt. 2022. Longitudinal analysis of sentiment and emotion in news media headlines using automated labelling with Transformer language models. *Plos one* 17, 10 (2022), e0276367.

[84] Santonu Sarkar, Shubha Ramachandran, G. Sathish Kumar, Madhu K. Iyengar, K. Rangarajan, and Saravanan Sivagnanam. 2009. Modularization of a Large-Scale Business Application: A Case Study. *IEEE Software* 26, 2 (2009), 28–35.

[85] Mario Schaarschmidt, Klaas-Jan Stol, and Brian Fitzgerald. 2025. The Insider's Dilemma: Employed Open Source Developers' Identification Imbalance and Intentions to Leave. *European Journal on Information Systems* 34, 5 (2025), 873–892.

[86] Carolyn B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering* 25, 4 (1999), 557–572.

[87] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In *18th ACM conference on Computer Supported Cooperative Work & Social Computing*. ACM, 1379–1392.

[88] Synopsys. 2024. Open Source Security and Risk Analysis. https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis/thankyou.html#UXoverview.

[89] Xin Tan, Minghui Zhou, and Brian Fitzgerald. 2020. Scaling Open Source Communities: An Empirical Study of the Linux Kernel. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 1222–1234.

[90] Xin Tan, Minghui Zhou, and Zeyu Sun. 2020. A first look at good first issues on GitHub. In *28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 398–409.

[91] Rust Team. 2024. Rust, A language empowering everyone to build reliable and efficient software. https://www.rust-lang.org/.

[92] Rust Moderation Team. 2021. Mod team resignation. https://github.com/rust-lang/team/pull/671.

[93] The Linux Foundation. 2022. Participating in Open Source Communities. https://www.linuxfoundation.org/tools/participating-in-open-source-communities/.

[94] Yingchen Tian, Yuxia Zhang, Klaas-Jan Stol, Lin Jiang, and Hui Liu. 2022. What Makes a Good Commit Message?. In *44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. ACM, New York, NY, USA, 2389–2401. doi:10.1145/3510003.3510205

[95] Yuriy Tymchuk, Andrea Mocci, and Michele Lanza. 2014. Collaboration in open-source projects: Myth or reality?. In *Proceedings of the 11th working conference on mining software repositories*. 304–307.

[96] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. 2018. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In *2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 644–655.

[97] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. ACM, New York, NY, USA, 805–816. doi:10.1145/2786805.2786850

[98] Wikipedia. 2024. Rust (programming language). https://en.wikipedia.org/wiki/Rust_(programming_language).

[99] Robert F Woolson. 2007. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials* (2007), 1–3.

[100] Chorng-Guang Wu, James H. Gerlach, and Clifford E. Young. 2007. An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management* 44, 3 (2007), 253 – 262. doi:10.1016/j.im.2006.12.006

[101] @XAMPPRocky. 2021. The Core Team Is Toxic. https://hackmd.io/@XAMPPRocky/r1HT-Z6_t

[102] Asterisk Youre. 2021. Steve Klabnik is no longer involved with The Rust Programming Language. https://www.reddit.com/r/rust/comments/q2zgni/steve_klabnik_is_no_longer_involved_with_the_rust/.

[103] Qunhong Zeng, Yuxia Zhang, Zeyu Sun, Yujie Guo, and Hui Liu. 2024. COLARE: Commit Classification via Fine-grained Context-aware Representation of Code Changes. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 752–763. doi:10.1109/SANER60148.2024.00082

[104] Yuxia Zhang, Hao He, and Minghui Zhou. 2022. Commercial Participation in OpenStack: Two Sides of a Coin. *Computer* 55, 2 (2022), 78–84. doi:10.1109/MC.2021.3133052

[105] Yuxia Zhang, Hui Liu, Xin Tan, Minghui Zhou, Zhi Jin, and Jiaxin Zhu. 2022. Turnover of Companies in OpenStack: Prevalence and Rationale. *ACM Trans. Softw. Eng. Methodol.* 31, 4, Article 75 (jul 2022), 24 pages. doi:10.1145/3510849

[106] Yuxia Zhang, Mian Qin, Klaas-Jan Stol, Minghui Zhou, and Hui Liu. 2024. How Are Paid and Volunteer Open Source Developers Different? A Study of the Rust Project. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[107] Yuxia Zhang, Klaas-Jan Stol, Hui Liu, and Minghui Zhou. 2022. Corporate dominance in open source ecosystems: a case study of OpenStack. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1048–1060.

[108] Yuxia Zhang, Klaas-Jan Stol, Minghui Zhou, Qunhong Zeng, Mian Qin, and Hui Liu. 2026. Data and code for paper "Contributions, Collaborations, and Transitions: Paid and Volunteer Developers in the Rust Community". https://doi.org/10.5281/zenodo.18219218

[109] Yuxia Zhang, Minghui Zhou, Audris Mockus, and Zhi Jin. 2021. Companies' Participation in OSS Development–An Empirical Study of OpenStack. *IEEE Transactions on Software Engineering* 47, 10 (2021), 2242–2259. doi:10.1109/TSE.2019.2946156

[110] Yuxia Zhang, Minghui Zhou, Klaas-Jan Stol, Jianyu Wu, and Zhi Jin. 2020. How Do Companies Collaborate in Open Source Ecosystems? An Empirical Study of OpenStack *(ICSE '20)*. ACM, New York, NY, USA, 1196–1208. doi:10.1145/3377811.3380376

[111] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. 2017. The impact of continuous integration on other software development practices: a large-scale empirical study. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 60–71.

[112] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A C-LSTM neural network for text classification. *arXiv preprint arXiv:1511.08630* (2015).

[113] Minghui Zhou, Qingying Chen, Audris Mockus, and Fengguang Wu. 2017. On the Scalability of Linux Kernel Maintainers' Work. In *2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 27–37.

[114] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: Willingness and opportunity in OSS community. In *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, 518–528.

[115] Minghui Zhou and Audris Mockus. 2015. Who Will Stay in the FLOSS Community? Modeling Participant's Initial Behavior. *IEEE Transactions on Software Engineering* 41, 1 (Jan 2015), 82–99. doi:10.1109/TSE.2014.2349496

[116] Minghui Zhou, Audris Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. 2016. Inflow and retention in oss communities with commercial involvement: A case study of three hybrid projects. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 25, 2 (2016), 1–29.

[117] Jiaxin Zhu and Jun Wei. 2019. An empirical study of multiple names and email addresses in oss version control repositories. In *IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 409–420.