Journal of Software Engineering Research and Development, 2025, 13:15, doi: 10.5753/jserd.2025.6090

Analysis of the Role of GitHub Discussions as a Tool for Onboarding Newcomers in Open Source Software

Ana Claudia Maciel [**State University of Maringá** | ***anamaciel@prof.unipar.br***]
Mairieli Wessel [**Radboud University** | ***mairieli.wessel@ru.nl***]
Alexander Serebrenik [**Eindhoven University of Technology** | ***a.serebrenik@tue.nl***]
Igor Wiese [**Federal Technological University of Paraná** | ***igor@utfpr.edu.br***]
Igor Steinmacher [**Northern Arizona University** | ***igor.steinmacher@nau.edu***]

Abstract

The introduction of the GitHub Discussions feature on GitHub provides a new, dedicated space for collaborative communication within open-source software (OSS) projects. This paper investigates the impact of GitHub Discussions on community engagement, focusing on newcomer onboarding and the activity surrounding issues and pull requests. Through a comprehensive empirical analysis of 285 OSS projects, we observe a significant shift in participation patterns, with GitHub Discussions emerging as a more attractive entry point for newcomers compared to traditional mechanisms like issues and pull requests. Our findings suggest that while GitHub Discussions lower the barrier for newcomer participation, it leads to a decrease in traditional contributions such as new issues and pull requests. Additionally, we explore the engagement levels of different contributor roles, demonstrating how GitHub Discussions foster diverse interactions and community involvement. The results provide critical insights into the evolving nature of collaboration on GitHub and highlight the role of GitHub Discussions in shaping the dynamics of OSS project ecosystems.

Keywords: *Open Source, Communication, Discussions, GitHub*

1 Introduction

The collaborative and participatory nature of software development is shaped by the communication channels used by communities of software developers (Storey et al., 2017). Simultaneously, these communities design their own social protocols for using tools, adapting them, and redirecting them over time (Giuffrida and Dittrich, 2013). Software developers employ a variety of communication channels, including discussions during code reviews (Tsay et al., 2014a), interactions on platforms such as Stack Overflow (Beyer et al., 2020), and participation in mailing lists (Guzzi et al., 2013). However, this changed after the dominance of GitHub, as the platform aimed to centralize developer collaboration.

The social coding environment on GitHub is a developer-friendly environment that integrates many features. Among them, GitHub offers an issues tracker focused on supporting coordination and pull requests with code review facilities focusing on supporting cooperation (Zhang et al., 2020). With the introduction of the GitHub Discussions feature, GitHub has filled a critical gap by providing a dedicated space for communication within a project repository. While issues and pull requests have traditionally allowed developers to communicate, they were not designed to support broader conversations that many projects require. GitHub Discussions serve as a central hub where community members can ask and answer questions, engage in open-ended conversations, and share important updates and decisions. This feature represents a significant shift, providing developers with a specific, purpose-built space within the repository to conduct meaningful, ongoing discussions that were previously scattered across issues and pull requests.

By offering a more inclusive and persistent space for interaction, GitHub Discussions create opportunities for developers who are not yet active contributors to become visible to the community and, importantly, to project maintainers. Such visibility is often a prerequisite for contribution acceptance, as maintainers tend to favor contributors they recognize (Ducheneaut, 2005). Identifying capable contributors, however, is time-consuming and challenging for maintainers, particularly in large and active projects (Jin et al., 2022). Simultaneously, developers who are unable to identify relevant tasks or establish communication with the community frequently abandon the project (Steinmacher et al., 2014). Prior research has shown that social interactions, especially those occurring in pull request reviews, influence contribution assessments due to the role of interpersonal relationships and social cues (Tsay et al., 2014a; Marlow et al., 2013). An efficient communication system like GitHub Discussions can help maintainers reduce their workload, support contributors, and lower entry barriers for newcomers. Understanding its impact on the existing GitHub environment is therefore essential to clarifying the role of this new communication mechanism in OSS projects.

Given the recency of the feature, the literature is still in the process of uncovering how GitHub Discussions function and what role they play within the collaborative environment provided by GitHub (Hata et al., 2022). For instance, Hata et al. (2022) examined how developers use the feature and its impact on development processes. Another study (Lima et al., 2023a) proposed a detector (RD-Detector) to identify related posts (i.e., duplicate or near-duplicate entries) within GitHub Discussions. Wang et al. (2023) explored the conversion of discussions into issues and vice versa, highlighting the multiple motivations for such actions.

More recently, Lima et al. (2025) investigated the scope and intentions behind shared links in GitHub Discussions. Their findings indicate that users frequently share both internal links (e.g., issues, pull requests, and other discussions) and external links (e.g., documentation, Stack Overflow, and video platforms such as YouTube) to provide context, clarify ideas, or suggest improvements. Similarly, to ease maintainers' workload, Lima et al. (2023b) proposed a Sentence-BERT-based approach for automatically identifying duplicate discussion posts, achieving high precision (77–100%) and 66% recall.

These studies reveal different facets of how GitHub Discussions are used and managed in OSS communities. Building upon this body of work, our study offers a broader perspective by examining how GitHub Discussions influence newcomer participation and the composition of contributor types in open-source projects. Notably, prior work on GitHub Discussions has examined overall usage patterns and how discussions can lead to project outcomes (e.g., some conversations are eventually converted into work items like issues Wang et al. (2023)), but no studies to date have specifically investigated the role of GitHub Discussions in onboarding newcomers. This leaves an important gap in our understanding: can an integrated discussion forum help lower entry barriers for new contributors, and if so, how?

In this work, we aim to fill this gap by examining GitHub Discussions through the lens of newcomer onboarding. Our study is, to the best of our knowledge, the first empirical investigation of how GitHub's discussion platform supports new OSS contributors. We conducted an empirical study involving 285 OSS projects hosted on GitHub, analyzing trends before and after the adoption of GitHub Discussions to understand how newcomers engage on this platform and how it impacts their pathway into the project. Specifically, we seek to understand what topics are discussed and who participates, and to evaluate whether participating in Discussions helps newcomers integrate into the community. By connecting our findings to the literature on Q&A communities and OSS socialization, we highlight what is novel about this integrated approach to developer communication. Our results provide data-informed evidence of the benefits and limitations of Discussions as an onboarding mechanism, offering guidance for maintainers and contributors to better support newcomers.

In summary, this paper makes the following contributions:

i. it examines how individuals with different roles engage in GitHub Discussions;
ii. it presents evidence on how GitHub Discussions support newcomer onboarding and engagement;
iii. it investigates the impact of GitHub Discussions on issue and pull request activities.

2 Research Questions

This study aims to explore how the GitHub Discussions feature affects different aspects of open-source software (OSS) projects by focusing on three points. First, it seeks to understand how contributors with different roles collaborate within discussions, shedding light on their interactions. Second, it examines the impact of GitHub Discussions on the onboarding process for newcomers, assessing whether this feature has facilitated integration into projects. Finally, it investigates how the introduction of discussions has influenced overall activity in issues and pull requests. To address this, we defined three research questions that we discuss in this section. More details about the data collection and analysis are presented in Section 3.

Research Question 1

How do contributors with different roles in projects collaborate in GitHub Discussions?

By distinguishing between various types of contributions, we can identify users who are particularly active or demonstrate expertise in certain aspects of the project (Furtado et al., 2013). Analyzing both the quantity and quality of these contributions allows us to form detailed contributor profiles, which can reveal whether these contributors are also highly engaged in Discussions, as well as in other project channels. Understanding these engagement patterns is critical for determining which types of contributors are most active in discussions. To address this question, we categorized contributors based on their involvement with issues, pull requests, and discussions, and then conducted an in-depth analysis of their participation in Discussions. These insights explain how contributors collaborate, enhancing our understanding of communication dynamics in open-source projects.

Research Question 2

How did GitHub Discussions change newcomers' onboarding on projects?

As previous research showed (Kafer et al., 2018), to survive and thrive, online communities must provide the benefits and experiences that contributors are looking for. The community's receptiveness to attempts to start conversations is an essential element of the community's success (Arguello et al., 2006). In this context, GitHub Discussions feature may serve as a vital tool by improving communication within projects, potentially easing barriers for newcomers. Therefore, we seek to understand how the introduction of Discussions has affected the onboarding process for new contributors through issues and pull requests. Understanding this impact is crucial for refining community management practices and optimizing project contributions, offering insights into more effective integration of newcomers in open-source software development.

Research Question 3

How did the introduction of GitHub Discussions influence the activity on issues and pull requests?

Prior work has analyzed traditional channels such as commits, issues, pull requests, reviews, and comments to understand how contributors interact and how work progresses in open source software projects (Ait et al., 2022). However, the impact of newly introduced mechanisms like GitHub Discussions remains underexplored. This is worth investigating given that the benefits of conversation (Joyce and Kraut,

2017) and answers (Lampe and Johnston, 2005) are key to keeping contributors engaged. By identifying how GitHub Discussions influence the creation of issues and pull requests, this research can offer insights into how communication tools shape project workflows and contributor dynamics, which can directly impact the sustainability and success of OSS projects.

3 Research Design

This section describes our data collection process and how we answered each research question defined previously. Figure 1 outlines the research method we followed in this study. For replication purposes, we made our data and source code publicly available[1].

3.1 Data Collection and Cleaning

First, data was collected from issues, pull requests, and discussions, and later, different analysis techniques were applied to the mined data. To select our study sample, we started with a list of 700 projects provided by Hata et al. (2022), who utilized the GitHub API to retrieve the most popular projects based on the number of stargazers. Then, we kept the projects with at least one discussion topic to make sure our analysis included only projects with the feature active and in use. After doing this, we kept a total of 91 projects.

However, this set had limitations in terms of representativeness and scale. To strengthen our analysis and increase the diversity and coverage of projects using the Discussions feature, we complemented the sample by retrieving additional projects through the SEART GitHub search engine (Dabic et al., 2021). To guarantee that we select active projects that have issues and pull requests on GitHub, we kept only those projects with at least 1,000 commits (the last should be in 2023), 1 issue, 1 pull request, and 1 release. We also kept only projects with at least 1,000 stars since we wanted to understand the involvement of the external community with the project, and the number of stars is a proxy for the popularity of the project (Borges et al., 2016). It is important to observe that we only used the number of stars as a filter, not to rank the projects.

We excluded projects that are forks. Lastly, we kept only the projects created before 01/01/2019, guaranteeing that they existed at least one year before GitHub Discussions feature was released. The search resulted in 4,594 projects. We used the GraphQL API[2] to check if the projects had at least 1 discussion topic, and the number went down to 2,057. From these projects, we randomly selected 200 to proceed to the next step. From these 200, we ended up getting errors for 6 of them when collecting data using the API. These errors occurred because these projects stopped using the GitHub Discussions during the data collection period. When a project disables the feature, the discussions are no longer available for collection. At the end, we collected data from 285 projects (91 from Hata et al. (2022) and 194 from our update). To ensure our sample was sufficiently representative, we considered statistical guidelines for determining an adequate sample size. According to Krejcie and Morgan (1970) and Cochran (1977), a sample size of 285 is more than adequate for ensuring statistical representativeness at a 95% confidence level with less than 10% margin of error, as the minimum required sample size under these conditions is only 96 projects. By selecting 285 projects, our sample not only exceeds this threshold but also enhances the precision of our estimates while maintaining a feasible data collection process.

It is important to note that we collected all the data in July 2023 to enable us to run the temporal analysis using Regression Discontinuity Design considering 12 months of history after the GitHub Discussions were adopted by each project.

Issues and pull requests were collected using the GitHub REST API[3]. However, the API still did not support the new GitHub Discussions feature, so to collect the discussion data, we used GraphQL, a query language to use the API. We obtained 1,115,807 users, 1,059,529 commits, 1,726,485 issues, 1,026,043 pull requests, 161,687 discussions posts, and 829,030 discussions comments. These numbers are related to the 285 projects. In addition, we also considered the project age (in days). Figure 2 shows the boxplots for these variables in log scale.

This study encompassed 285 projects developed in different programming languages, reflecting the technological diversity of the open-source ecosystem on GitHub. The most frequent languages were TypeScript (56 projects), JavaScript (39), Python (32), C (24), C++ (23), Go (23), PHP (20), Java (18), C (16), and Rust (10). Other ten languages (including Ruby, Shell, Kotlin, and Swift) appeared less frequently, totaling 24 projects. This distribution shows that the sample covers both widely adopted languages in modern systems (e.g., TypeScript, Go, Rust, Python) and traditional, consolidated ones (e.g., Java, C, C++), which brings robustness and heterogeneity to our analysis. The detailed numbers are also presented in Table 1.

Table 1. Distribution of projects by programming language

Language	#Projects
TypeScript	56
JavaScript	39
Python	32
C#	24
C++	23
Go	23
PHP	20
Java	18
C	16
Rust	10
Other languages (10 languages)	24
Total	285

After collecting the data, we filtered from our dataset activities created by bots, and we identified 246 bots. To do so, we leveraged the combination of two bot identification techniques with the highest precision: (i) "[bot]" suffix detection

[1] https://github.com/anamaciel/DiscussionsNewcomersOSS
[2] https://graphql.org/
[3] https://docs.github.com/en/rest

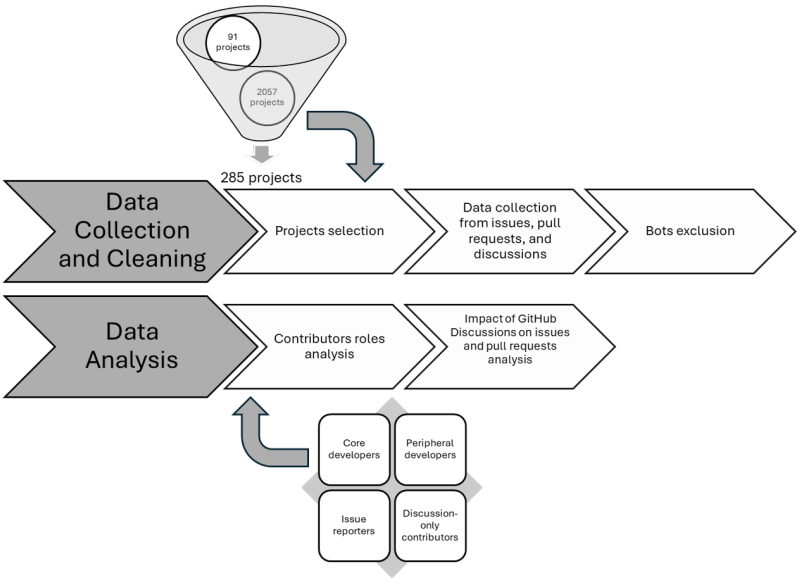

Figure 1. Research Method.

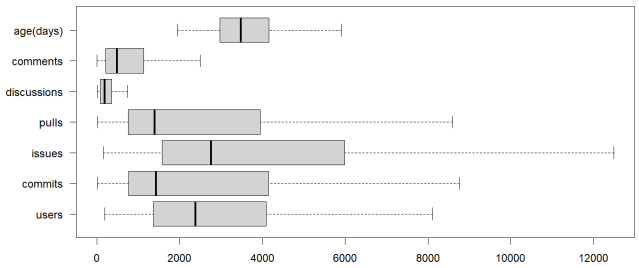

Figure 2. Distribution of the main dimensions of our dataset (with no outliers).

and (ii) a list of well-known bot accounts. We first removed all accounts that had "[bot]" in their username. With this, we identify bots that have an official integration with GitHub and are properly tagged as such in the platform. To improve the detection of bots, we also relied on a list of 385 commonly used bots manually identified by Chidambaram et al. (2023).

3.2 Data Analysis

This section presents the data analysis conducted to address the research questions.

3.2.1 RQ1 – Analyzing contributors roles

For each project, we classified the project contributors according to the roles that they play in the project based on their interaction with the different features on GitHub. To classify the contributors, we considered the following roles:

- **Core developers**: contributors in the group responsible for roughly 80% of the commits in the project (Mockus et al., 2002).
- **Peripheral developers**: contributors who opened pull requests but are not core developers;
- **Issue reporters**: contributors who opened issues but did not contribute with pull requests;
- **Discussion-only contributors**: those who interacted exclusively through Discussions, either by posting, commenting, or replying. In this feature, each discussion begins with a **post** (Figure 3(a)), which can receive **comments** (Figure 3(b)) or replies. For discussions under QA categories, a **comment** can be marked as an **answer** (Figure 3(c)), although not all discussions require one.

Figure 3. Anatomy of a discussion

After defining the roles, we checked where each contributor appeared. We performed statistical tests to compare the engagement of the members of each group opening, commenting, and replying to GitHub Discussions. We used the ANOVA test to test the following hypotheses:

H_0^1 The means of the distributions of **the number of discussions opened** by core developers, peripheral developers, issue reporters, and discussion-only contributors are the same.

H_0^2 The means of the distributions of **the number of comments** by core developers, peripheral developers, issue reporters, and discussion-only contributors who comment on discussions are the same.

To calculate the effect sizes for pairwise comparisons be-

tween the levels of factors in a statistical model, we use the **'eff_size'** function from the **'emmeans'** package in R. Initially, we fit a linear model, for example, **'modelDiscussions'**, and obtain the marginal estimates (adjusted means) for the factor levels using the **'emmeans'** function. These estimates are stored in an object, 'ems', which serves as the basis for subsequent calculations. The **'eff_size'** function is then called with the parameters **'method = "pairwise"'**, **'edf = df.residual(modelDiscussions)'**, and **'sigma = sigma(modelDiscussions)'**, where **'edf'** represents the residual degrees of freedom of the model and 'sigma' the residual standard deviation.

The "pairwise" method specifies that we want to calculate effect sizes for all pairwise comparisons between the factor levels. For each pair of levels, the function computes the difference between the corresponding marginal estimates. This difference is then standardized by dividing by the residual standard deviation of the model (σ). This procedure results in the calculation of Cohen's d for each pairwise comparison, which is a standardized measure of effect magnitude. The residual degrees of freedom (edf) are used to adjust the estimates, improving the precision of the calculated effect sizes.

For example, suppose we have two factors, **'factor1'** and **'factor2'**, with levels A and B, and X and Y, respectively. The marginal estimates might be 3.0, 3.5, 4.0, and 4.5 for the combinations A-X, A-Y, B-X, and B-Y. With a residual standard deviation of 0.5, the **'eff_size'** function calculates the differences between all pairwise combinations, such as A-X vs A-Y, resulting in differences like 0.5, 1.0, and 1.5. By dividing these differences by σ, we obtain Cohen's d values of 1.0, 2.0, and 3.0, respectively, for each comparison. Thus, the **'eff_size'** function allows us to assess the magnitude of differences between the factor levels in the model in a standardized manner, adjusted for residual variability and degrees of freedom.

Also, we counted the number of discussions with answers. In addition to that, we checked the number of discussion comments broken down by role. We determine who responds to whom within discussions. For each project in our dataset, we counted all discussions with selected answers, no answers, and no comments. We also counted the answers in discussion topics by role.

3.2.2 RQ2 and RQ3 – Analyzing the impact of GitHub Discussions on issues and pull requests

To answer **RQ2**, we conducted a quantitative analysis of issues, pull requests, and discussions. We collected the date of each contributor's first interaction in the project. The first interaction of a contributor can be (i) creating an issue, (ii) submitting a pull request, (iii) creating a new post discussion, or (iv) commenting on an existing post discussion. In the two last cases, we considered that the contributor was onboarded via discussion. We considered all project history and counted all the contributors who used GitHub Discussions to join the project. To verify if GitHub Discussions changed issues and pull request dynamics, we compared the contributors who entered through issues and pull requests with those who entered through discussions.

To answer **RQ2** and **RQ3**, we collect data for different outcome variables and consider the start of GitHub Discussions with each project in our sample as an intervention. We employed a Regression Discontinuity Design (RDD) (Thistlethwaite and Campbell, 1960; Imbens and Lemieux, 2008), which has been widely applied in the software engineering domain (Zhao et al., 2017; Cassee et al., 2020; Wessel et al., 2020). RDD is a technique for modeling the extent of a discontinuity at the moment of intervention and long after the intervention. The technique is based on the assumption that if the intervention does not affect the outcome, there would be no discontinuity, and the outcome would be continuous over time (Cook and Campbell, 1979). The statistical model behind RDD is:

$$y_i = \alpha + \beta \cdot \mathit{time}_i + \gamma \cdot \mathit{intervention}_i + \delta \cdot \mathit{time\_after\_intervention}_i + \eta \cdot \text{controls}_i + \varepsilon_i$$

where i indicates the observations for a given project.

To model the passage of time as well as the GitHub Discussions adoption, we include three additional variables: *time*, *time after intervention*, and *intervention*. The *time* variable is measured as months at the time j from the start to the end of our observation period for each project (24 months). The *intervention* variable is a binary value used to indicate whether the time j occurs before ($\mathit{intervention} = 0$) or after ($\mathit{intervention} = 1$) adoption event. The *time_after_intervention* variable counts the number of months at a time j since the GitHub Discussions adoption, and the variable is set up to 0 before adoption.

The $\mathit{controls}_i$ variables enable the analysis of GitHub Discussions adoption effects rather than confounding the effects that influence the dependent variables. For observations before the intervention, holding controls constant, the resulting regression line has a slope of β, and after the intervention, it has a slope of $\beta + \delta$. Further, the size of the intervention effect is measured as the difference equal to γ between the two regression values of y_i at the moment of the intervention.

We estimate the model coefficients using the `lmerTest` R package (Kuznetsova et al., 2017). In mixed-effects regression, the model coefficients and the statistical significance of these coefficients are used to determine whether the intervention (GitHub Discussions introduction) immediately impacted the dependent variables and how the one-year trends in these variables changed. We evaluate the model fit using *marginal* (R^2_m) and *conditional* (R^2_c) scores, as described by Nakagawa and Schielzeth (2013). The R^2_m can be interpreted as the variance explained by the fixed effects alone, and R^2_c is the variance explained by the fixed and random effects together.

We aggregated the project data in monthly (30-day) time frames and collected the four variables we expected to be influenced by the introduction of the GitHub Discussions:

- **Number of new contributors (issues):** the number of new unique contributors who created issues for the project.
- **Number of new contributors (pull requests):** the number of new unique contributors who submitted pull requests to the project.
- **Issues per month:** the number of issues opened monthly in the project.

- **Pull requests per month:** the number of contributions (pull requests) received per month in the project.

We modeled eight control variables. To account for the different behaviors that might be observed across projects or programming languages, we follow the previous works (Zhao et al., 2017; Cassee et al., 2020; Wessel et al., 2020) and used *project name* and *programming language* as random effect variables:

- **Project name:** the project's name used to identify the project on GitHub. We considered that different projects could lead to different contribution patterns.
- **Programming language:** The primary project programming language as automatically determined and provided by GitHub. We considered that projects with different programming languages could lead to different activities and contribution patterns.

The other control variables were used according to the context. We used the three variables related to pull requests when the context of interest was pull requests, and we used the other three related to issues when the context was issues:

- **Total number of pull requests:** total number of pull requests before the first discussion date.
- **Total number of issues:** total number of issues before the first discussion date.
- **Total number of pull requests authors:** Total number of contributors who submitted pull requests to the project before the first discussion date.
- **Total number of issues authors:** Total number of contributors who submitted issues to the project before the first discussion date.
- **Time since the first pull request:** in months, computed since the earliest recorded pull request in the project history. We use this to capture the difference in starting using GitHub Discussions earlier or later in the project life cycle after the projects started to use pull requests.
- **Time since the first issue:** in months, computed since the earliest recorded issue in the project history. We use this to capture the difference in starting using GitHub Discussions earlier or later in the project life cycle after the projects started to use issues.

Similar to previous work (Zhao et al., 2017; Wessel et al., 2020; Cassee et al., 2020), we also excluded 30 days around the GitHub Discussions adoption to avoid the influence of instability caused during this period.

4 Results

In the following, we report the results of our study by research question.

4.1 Contributors roles in GitHub Discussions (RQ1)

To investigate our first research question, we distinguish contributors according to their role: core developers, peripheral developers, issue reporters, and discussion-only contributors. Before the analysis, it is important to note that the number of contributors per role is imbalanced. The distribution per role is presented in Figure 4.

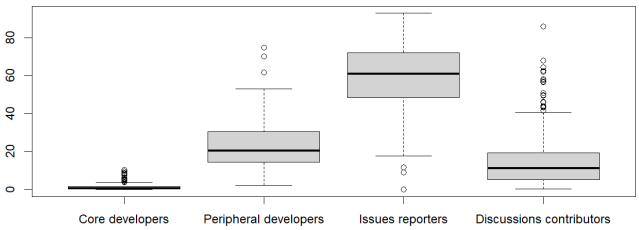

Figure 4. Distribution of the percentage of contributors per role.

Regarding discussions opened, the distribution per role is shown in Figure 5. We conducted a multigroup analysis using ANOVA (F-value=719.25 and $p < 2.2 \times 10^{-16}$). Therefore, we reject H_0^1, since there is a significant difference in the number of discussions opened per group, and this is also quite clear in Figure 5. So, we calculated the effect size to understand the magnitude of the difference between each pair of groups (Sullivan and Feinn, 2012). The results for the ANOVA and effect size are presented in Table 2. The estimated effect sizes reveal significant differences in the adjusted means between various groups of developers. Discussion-only contributors have an extremely higher involvement in opened discussions compared to Peripheral developers ($d = 3.176$) and Core developers ($d = -3.519$). They also show a notable impact relative to Issue reporters (d = 1.994). In contrast, the difference between Core developers and Peripheral developers is small ($d = -0.343$), and the difference between Issue reporters and Peripheral developers is large ($d = 1.181$). These magnitudes suggest that Discussion-only contributors and Issue reporters are significantly more active in open discussion topics or influential compared to Peripheral developers, while Core developers have relatively lower involvement compared to other groups, except with Peripheral developers. Among these contrasts, all large and statistically significant effects are highlighted in bold in Table 2. The Core developers – Peripheral developers difference, however, was not highlighted because, although statistically significant, its magnitude ($d = -0.343$) is below the threshold commonly considered a large effect ($|d| \leq 0.8$).

Table 2. ANOVA results when comparing the number of discussions opened per role.

Contrast	Effect size	SE	*p*-value
Core developers - Discussion-only contributors	**-3.52**	**1.18**	**<.0001**
Core developers - Issue reporters	**-1.53**	**1.18**	**<.0001**
Core developers - Peripheral developers	-0.34	1.18	0.0003
Discussion-only contributors - Issue reporters	**1.99**	**1.18**	**<.0001**
Discussion-only contributors - Peripheral developers	**3.18**	**1.18**	**<.0001**
Issue reporters - Peripheral developers	**1.18**	**1.18**	**<.0001**

Table 3. ANOVA results when comparing the number of discussion comments per role.

Contrast	Effect size	SE	*p*-value
Core developers - Discussion-only contributors	0.43	1.23	<.0001
Core developers - Issue reporters	**1.41**	**1.23**	**<.0001**
Core developers - Peripheral developers	**1.70**	**1.23**	**<.0001**
Discussion-only contributors - Issue reporters	**1.16**	**1.23**	**<.0001**
Discussion-only contributors - Peripheral developers	**1.09**	**1.23**	**<.0001**
Issue reporters - Peripheral developers	-0.07	1.23	0.22

Regarding the comments per role, the distribution per role

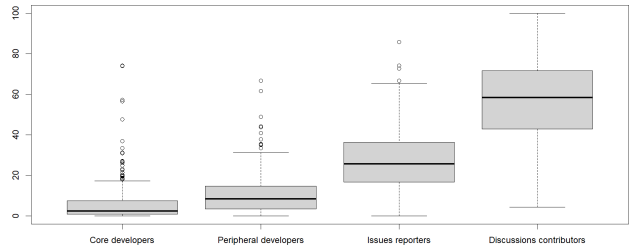

Figure 5. Distribution of the percentage of discussion opened per role per project.

is presented in Figure 6. The ANOVA comparison resulted in F-value = 127.43 and $p < 2.2 \times 10^{-16}$. Thus, we reject H_0^2, since there is a significant difference in the number of comments per group. The results from the statistical tests are presented in Table 3, which also shows the effect sizes.

The results indicate clear differences in the participation and roles of different groups involved in the project development. Core developers emerge as the primary contributors, exhibiting a substantially higher effect size than other groups. The magnitude of the difference between issue reporters (1.407) and peripheral developers (1.700) suggests that core developers have a much more active and distinct contribution, likely involving technical leadership, critical decision-making, and the implementation of significant code changes. In contrast, the difference between core developers and discussion-only contributors is more moderate (0.429), indicating that discussion-only contributors also play a relevant role, though less technical and direct regarding code contributions. Furthermore, there are significant differences between discussion-only contributors and issue reporters (1.159) and peripheral developers (1.088), suggesting that discussion-only contributors are more engaged in guiding and communicating about the project. On the other hand, the near-zero relationship between issue reporters and peripheral developers (-0.071) indicates that these two groups have similar contribution patterns, likely limited to less technical or lower-impact interactions within the project. These findings point to a clear hierarchy of participation, with core developers assuming leadership in discussion comments.

All contrasts with large and statistically significant effects ($|d| \geq 0.8$ and $p < 0.05$) are highlighted in bold in Table 3. The contrasts Core developers – Discussion-only contributors (0.429) and Issue reporters – Peripheral developers (-0.071) are not highlighted because, although the first is statistically significant, its effect size is below the threshold for a large effect, and the latter is not statistically significant. These findings point to a clear hierarchy of participation, with core developers assuming leadership in discussion comments.

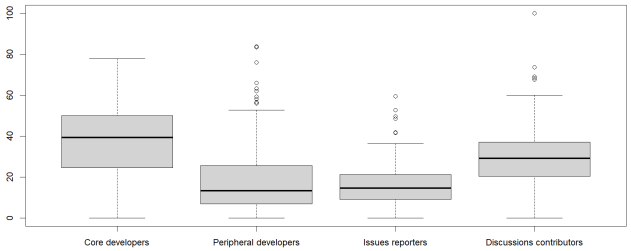

Figure 6. Distribution of the percentage of comments per role per project.

Table 4 shows the distribution of discussion posts that receive answers according to the role of the contributor who opened it. A discussion is considered "answered" when a comment is explicitly marked as an answer. Table 5 complements the information showing the distribution of the answers according to the role of the contributors who wrote them. We further analyzed the posts to determine whether the original authors responded to their own posts. As summarized in Table 6, we found that 142 of the 359 posts with answers (40%) made by core developers were answered by the authors. Similarly, 26% of the posts with answers made by peripheral developers, 21% of the posts made by issue reporters, and 23% of the posts made by discussion-only contributors were answered by their original authors. Another interesting point is that most of the answers come from people who are active only on GitHub Discussions.

Table 4. Discussion that received answers per opener role

Opened by Has answer?	Core developers	Peripheral developers	Issue reporters	Discussions contributors
Yes	359 (10%)	3543 (27%)	11106 (32%)	28596 (35%)
No selected answer	1905 (56%)	6728 (51%)	17661 (51%)	35664 (43%)
No comments	1155 (34%)	2829 (22%)	5973 (17%)	18047 (22%)
Total	3419	13100	34740	82307

Table 5. Role of the respondents of selected answers vs. role of the Discussion opener

Opened by Answered by	Core developers	Peripheral developers	Issue reporters	Discussion contributors
Core developers	349 (97%)	894 (25%)	3077 (28%)	5284 (18%)
Peripheral developers	7 (2%)	2485 (70%)	1265 (11%)	3784 (13%)
Issue reporters	2 (1%)	128 (2%)	6543 (59%)	945 (3%)
Discussion-only contributors	1 (0%)	36 (1%)	221 (2%)	18483 (65%)
Total	359	3543	11106	28596

Table 6. Proportion of posts with answers that were answered by the post authors.

Opened by	Posts with answers	Posts answered by author	Percentage
Core developers	359	142	40%
Peripheral developers	3543	917	26%
Issue reporters	11106	2333	21%
Discussion-only contributors	28596	6574	23%

In addition to checking the number of comments marked as an answer, we collect the number of comments (regardless of whether it was marked as an answer or not) posted by contributors from each role. These numbers are presented in Table 7. We also investigated whether the authors of the posts were commenting on their own posts, and these results are summarized in Table 8. We found that 33% of the comments made by the core developers were from the post authors; accordingly, this happened with 32% of the posts by peripheral developers, 26% by issue reporters, and 27% of discussion-only contributors.

Table 7. All comments by discussion opener.

Commented by \ Opened by	Core developers	Peripheral developers	Issue reporters	Discussions contributors
Core developers	10248 (42%)	15078 (26%)	37698 (26%)	68182 (25%)
Peripheral developers	4789 (19%)	31966 (56%)	24644 (17%)	40270 (15%)
Issue reporters	3038 (12%)	3260 (6%)	61470 (43%)	27851 (10%)
Discussion-only contributors	6618 (27%)	6553 (12%)	19731 (14%)	140471 (50%)
Total	24693	56857	143543	276774

Table 8. Proportion of comments made by the same contributor who opened the post.

Opened by	Core developers	Peripheral developers	Issue reporters	Discussion-only contributors
Total comments	24693	56857	143543	276774
Comments by author	8160	18195	37292	74717
% of comments by authors	33%	32%	26%	27%

Research Question 1

How do contributors with different roles in projects collaborate in GitHub Discussions?

Core developers open fewer discussion topics but are more active in commenting within discussions. In contrast, discussion-only contributors are more prominent in starting new topics, although their commenting activity is less than core developers. The issue reporters and peripheral developers have more balanced contributions than the core and discussion contributor groups.

4.2 Changes in newcomers' onboarding (RQ2)

Our findings indicate the main way that new contributors make their first interaction with the project is through the GitHub Discussions (47%), followed by issues (41%) and then pull requests (12%) (see Figure 7). Additionally, we found that 14.5% of contributors who joined GitHub Discussions later contributed to issues, and 2.64% to pull requests. This is a little below the ratio of those who enter via issue tracker and later open pull requests in these projects (6.14%). To better understand the participation of those who join via GitHub Discussions and later opened pull requests, we further examined their activity: 29.1% only opened discussions, 30.8% only commented on existing ones, and 40.1% both opened and commented on discussions. This distribution indicates that a substantial portion of these newcomers go beyond one-off questions and actively engage in ongoing conversations.

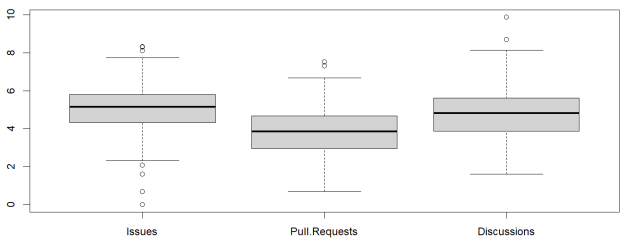

Figure 7. Number of newcomers who started per channel (log scale)

Given this new scenario, in this RQ, we investigated the effects of GitHub Discussions adoption on the number of new contributors on issues and pull requests using RDD models. For these models, the number of new contributors (in issues/pull requests) is the dependent variable. Table 9 summarizes the results of the two RDD models we fitted. In addition to the model coefficients, the table shows the sum of squares, with a variance explained for each variable.

Analyzing the model for new contributors on issues, we found that the fixed-effects part fits the data well ($R^2_m = 0.53$). However, considering $R^2_c = 0.69$, variability also appears from project-to-project and language-to-language. Among the fixed effects, we observe that the total number of issue authors explains the largest amount of variability in the model, indicating that the number of new issue authors per month is strongly associated with the overall number of issue contributors to a project. Moreover, the statistical significance of the *time_after_intervention* coefficients indicate a discontinuity at adoption time, followed by a decrease after the GitHub Discussions introduction. This implies that, on average, fewer new contributors create issues after the introduction of GitHub Discussions to a project.

Similar to the previous model, the fixed-effect part of the new contributors on the pull requests model fits the data well ($R^2_m = 0.40$), even though a considerable amount of variability is explained by random effects ($R^2_c = 0.71$). We also note similar results on fixed effects: projects with more contributors tend to have more new contributors on a monthly basis. Table 9 shows that the effect of GitHub Discussions adoption on new contributors is similar to both issues and pull requests and presents a summary of model predictors, their direction of effect, statistical significance, and substantive interpretation for new contributor activity on Issues and Pull Requests. The results indicate that, controlling for project size and prior contributor activity, the adoption of Discussions is associated with a modest but statistically significant negative slope in the number of new contributors over time ($time\_after\_intervention$). Other predictors, including cumulative numbers of issues and pull requests and prior author participation, remain positively associated with newcomer engagement. Immediate changes following the intervention ($interventionTrue$) are not significant, suggesting that the impact of Discussions manifests gradually rather than abruptly.

Research Question 2

How did GitHub Discussions change newcomers' onboarding on projects?

Our findings indicate that GitHub Discussions change the dynamics of newcomers' onboarding in issues and pull requests. On average, there are fewer new contributors in pull requests and issues after introducing GitHub Discussions.

4.3 The Impact of GitHub Discussions on Issues and Pull Requests (RQ3)

In this RQ, we investigate the effects of GitHub Discussions adoption on the monthly number of incoming issues and pull requests. For these models, the dependent variable is the number of new contributions (issues/pull requests) per month. Table 10 summarizes the results of these two RDD models and shows the predictors, direction of effect, statistical significance, and substantive interpretation for monthly

Table 9. The Effect of GitHub Discussions on new contributors on Issues and Pull Requests. The response is in log scale.

New Contributors on Issues			New Contributors on Pull Requests		
	Coefficients	SS		Coefficients	SS
Intercept	-2.598***		Intercept	-1.319***	
time_since_first_issue	0.010***	10.516	time_since_first_pull	-0.012***	11.199
log(total_number_issue_authors)	0.568***	262.191	log(total_number_pull_authors)	0.519***	258.273
log(total_number_issues)	0.271***	18.852	log(total_number_pulls)	0.096***	5.598
time	0.010***	1.852	**time**	0.006**	0.151
interventionTrue	-0.021	0.400	**interventionTrue**	-0.001	0.023
time_after_intervention	-0.020***	8.221	**time_after_intervention**	-0.012***	2.841
Marginal R^2		0.53			0.40
Conditional R^2		0.69			0.71

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. SS stands for "Sum of Squares".
Time series predictors in **bold**.

issue and pull request activity. Controlling for project size and prior contributor activity, the adoption of Discussions is associated with a significant negative slope over time ($time\_after\_intervention$) for both issues and pull requests, indicating a gradual reduction in activity. Immediate changes following the intervention ($interventionTrue$) are not significant. Other predictors, including cumulative numbers of issues, pull requests, and authors, remain positively associated with monthly activity. In addition to the model coefficients, the table shows the sum of squares, with a variance explained for each variable.

Considering the model of issues per month, we found that the combined fixed-and-random effects ($R^2_c = 0.71$) fit the data better than the fixed effects alone ($R^2_m = 0.40$). This shows that most of the explained variability in the data is associated with project-to-project and language-to-language variability, rather than the fixed effects. Regarding the time series predictors, the model did not detect any statistically significant trend in the number of issues per month before the intervention. However, at the time of the intervention, it grew (positive), and then it dropped (negative), suggesting that the number of monthly issues decreased after the GitHub Discussions introduction to the projects. Additionally, from Table 10, we can also observe that the total number of contributions (issues) explains most of the variability in the results.

Investigating the results of the pull requests per month model, we also found that combined fixed-and-random effects fit the data better. As above, the total number of contributions (pull requests) is responsible for most of the variability encountered in the results. The effect of GitHub Discussions adoption in the number of monthly pull requests is similar to the previous model. The statistical significance of the time series predictors indicates that the introduction of GitHub Discussions affected the trend in the number of pull requests. We note an increasing trend before adoption; a statistically significant discontinuity at the intervention time; and a negative trend after adoption that indicates that the number of pull requests has started to decrease.

However, beyond this general finding, the model also reveals more nuanced effects of GitHub Discussions adoption on pull request activity. Specifically, the time series predictors highlight an upward trend in the number of pull requests before the adoption of GitHub Discussions, suggesting that activity was increasing over time. At the moment of intervention (i.e., when GitHub Discussions was introduced), a statistically significant jump occurred, implying an immediate impact on contributor behavior. Yet, this positive effect was short-lived, as the model detects a negative trend post-adoption, indicating that after this initial boost, the number of pull requests began to decline.

This pattern mirrors the findings from the issues model, implying that while GitHub Discussions may have initially stimulated participation, it ultimately led to a decrease in the frequency of pull requests over time. This suggests a potential shift in how contributors engage with projects, possibly diverting efforts from direct contributions (like pull requests) to more indirect forms of collaboration within the GitHub Discussions feature. Further exploration could investigate whether Discussions became a preferred channel for collaboration, reducing the need for more formal contributions, or if other external factors influenced this decline.

Research Question 3

How did the introduction of GitHub Discussions influence the activity on issues and pull requests?

After the introduction of GitHub Discussions, on average, the monthly numbers of issues and pull request contributions decreased.

5 Discussion

In this section, we provide some more detailed discussion about the results that answered our research questions.

GitHub Discussions as a Mechanism for Legitimate Peripheral Participation? The GitHub Discussions environment involves different profiles of contributors, from outsiders to the project (e.g., users) to those who maintain the project on the technical end (e.g., maintainers, owners). Although all these people are engaged, most (by far) of the discussion threads are opened by people who were not involved in the project before GitHub Discussions. This shows that there was indeed a need for a channel that focuses specifi-

Table 10. The Effect of GitHub Discussions on number of issues and pull requests per month. The response is in log scale.

Issues Per Month			Pull Requests Per Month		
	Coefficients	SS		Coefficients	SS
Intercept	2.659***		Intercept	1.779***	
log(total_number_issues)	0.0001***	135.474	log(total_number_pulls)	0.0001***	162.824
time_since_first_issue	-0.532***	4.157	time_since_first_pull	-0.512***	0.736
log(total_number_issue_authors)	0.394***	26.872	log(total_number_pull_authors)	0.502***	51.979
time	0.015***	0.363	**time**	0.020***	15.098
interventionTrue	0.016	0.363	**interventionTrue**	0.051	0.529
time_after_intervention	-0.032***	20.754	**time_after_intervention**	-0.031***	20.054
Marginal R^2		0.40			0.39
Conditional R^2		0.71			0.72

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. SS stands for "Sum of Squares".
Time series predictors in **bold**.

cally on communication that transcends the previous GitHub features—which focus on project coordination (issue tracker) and cooperation (pull requests). Creating such a channel may encourage the onboarding of new contributors to the project since it lowers the initial barriers to joining an OSS project. This would make GitHub projects closer to a community of practice (Wenger, 2011), in which learning happens as a contextual social phenomenon, achieved through participation in a community (introduced by Lave and Wenger (2001) as Legitimate Periphery Participation, or LPP).

According to the LPP theory, newcomers become acquainted with the tasks, vocabulary, and principles of the community's practitioners by observing and receiving guidance from experienced members. This was also clear in our results, given that core and peripheral developers are the most active. Considering that core developers and peripheral developers have more knowledge about the project, by interacting via discussions they share this knowledge and help to create social ties with the newcomers (who may get comfortable with the project and seek to engage further).

Our findings extend prior work that analyzed how developers use Discussions in general (Hata et al., 2022), by framing this activity performed by newcomers within the theory of LPP. While previous studies have shown that GitHub Discussions can support coordination (e.g., through conversion into issues) or information sharing (e.g., link exchange), none have investigated how this space enables the social learning processes by newcomers. In this sense, our results suggest that GitHub Discussions can support a process similar to Legitimate Peripheral Participation: newcomers often enter at the periphery, observe and interact with more experienced members, and in some cases progress toward more central roles. While we cannot claim that this trajectory applies to all newcomers, our findings provide empirical evidence that GitHub's integrated forum lowers initial barriers and facilitates socialization, extending prior work on Q&A platforms and OSS onboarding.

Qualitative Insights into Unanswered Discussions: While our quantitative analysis revealed that a portion of discussion posts remained without a formally marked answer, a qualitative inspection of a sample of these posts suggests that the binary classification of 'answered' vs. 'unanswered' may not fully capture the richness of community interaction. To better understand the nature of these discussions, we manually examined a sample of threads from the `livewire/livewire` project that were labeled as 'unanswered' in our dataset. Our inspection revealed that many of these posts did, in fact, receive informative responses or technical suggestions, even if no single comment was formally selected as the answer. For instance:

- In **Discussion #1031**, a user reported a CSRF error even when logged out. Although the thread was not marked as resolved, a contributor (`mblarsen`) provided a detailed technical workaround involving a custom JavaScript hook and referenced a related internal issue (#537). This suggests that the discussion was informative and addressed the user's concern, even without formal closure.
- In **Discussion #1101**, a user suggested a JavaScript hook for error handling in modals. The proposal was not only acknowledged by other contributors but also led to the implementation of the `livewire.onError()` feature in a subsequent release. Despite the lack of a marked answer, the discussion directly influenced project development.
- In **Discussion #1244**, a user asked how to display the original filename of an uploaded file. A contributor provided a functional code snippet that solved the problem, and the post received positive reactions (). Although the original author did not return to confirm, the response was complete and actionable.

These examples highlight that GitHub Discussions often function as a space for collaborative problem-solving, where solutions emerge through community interaction rather than authoritative answers. This pattern differs from platforms like Stack Overflow, where accepted answers are central to thread closure. In GitHub Discussions, even threads without a marked answer can accumulate valuable context, workarounds, or references to external resources. This suggests that future studies should consider more nuanced metrics for engagement, such as the presence of informative comments or references to related issues, rather than relying solely on binary answer markers.

Qualitative Mechanisms Behind Workload Changes: Our quantitative results for RQ3 indicated a significant change in issue and PR volume for some projects post-discussions adoption. To understand the mechanisms behind this change, we conducted a qualitative case study on the `livewire/livewire` project. We manually analyzed some posts to identify scenarios where they potentially reduced the need for formal issues or pull requests.

We observed three recurring patterns that acted as a filter or a resolver, thereby impacting issue and PR counts:

- **Collaborative Problem-Solving Within Threads:** In **Discussion #1031**, a user reported a complex CSRF error. Rather than opening an issue, a contributor provided a workaround via a custom JavaScript hook directly within the discussion thread. This suggests that GitHub Discussions can *resolve* certain problems collaboratively and asynchronously, preventing the creation of issues for environment-specific or complex-but-rare problems.
- **Support Channel Replacement: Discussion #986** involved a user asking how to implement a progress bar. The thread evolved into a technical debate where users collaboratively arrived at a solution using XHR events. This demonstrates how GitHub Discussions can absorb "how-to" questions that might otherwise be filed as support-style issues, keeping the issue tracker focused on actionable bug reports and feature requests.
- **Feature Validation Filter:** In **Discussion #995**, a user proposed a new feature (method chaining). The community debated its merits, and a maintainer clarified technical limitations and the project's vision. This discussion served as a *validation filter*, potentially preventing the submission of a formal feature request issue or a PR that would likely be rejected, thus streamlining the maintainers' review workload.

"The more you do, the more you can do." This paraphrases one of the conclusions that Vasilescu et al. (2014) drew from analyzing how social Q&A portals changed the landscape of OSS. They evidenced that developers active in mailing lists and the new Q&A portal (StackExchange) have a higher chance of being answered on StackExchange than other users. In our case, although the percentage of answers from comments from core members is similar to those from GitHub Discussions only contributors, it is worth noticing that the number of core members is significantly lower than the number of discussion-only contributors, with the median difference being around 17 times. As seen in Table 3, core developers are active in commenting. This is also perceived when we analyzed the percentage of comments and selected answers (Tables 7 and 5, respectively). Although in absolute numbers, discussion-only contributors comment and reply to more threads, looking from a relative perspective, core members take a heavier load given that they are numerically disfavored.

The other analyzed goal refers to the number of comments on posts. We observed that at the moment of our data collection, less than 35% of the posts had no answers. While this is a small number, this is in line with a recent study on StackOverflow, which shows that 46% of the questions on that platform are unanswered (Mondal et al., 2021). However, this number was not this big previously, the same paper (Mondal et al., 2021) shows that this number was under 10% until 2013 ("the early days of StackOverflow"). Since GitHub Discussions are in their early days, the number can be considered rather large—even more, so if we consider that the forum is embedded in the environment where the development happens. It would be important to understand the reasons for such a large rate. One hypothesis may be the way that discussions are used since there may be cases in which the post is rather informative than questions. Another possibility is a large number of similar and duplicate posts (Lima et al., 2023a) not dissimilar to duplicate questions on other Q&A platforms (Kamienski et al., 2023; Gao et al., 2022).

Our results extend the findings of Vasilescu et al. (2014) by showing that, in the context of GitHub Discussions, core developers—although fewer in number—carry a relatively higher load of providing answers, while discussion-only contributors participate more in absolute terms. This highlights how integrated Q&A environments on GitHub distribute effort between experienced and peripheral participants, reinforcing the idea that active engagement from a small group of core members supports broader community participation. We recommend that future studies focus on this specific phenomenon. We also see this as an important outcome that may inform GitHub.

GitHub Discussions as a mechanism of onboarding and retention. Regarding the onboarding of newcomers after the introduction of GitHub Discussions, we evidenced that the new feature became, by far, the most attractive way to engage new contributors on the projects' GitHub. This, per se, may be seen as a big win that GitHub Discussions brought to the projects—by bringing the community closer to the project team—and to GitHub—by fostering engagement with the platform. We can say that GitHub Discussions encourage the onboarding of new contributors. This is even more interesting when we observe that some of these contributors join GitHub Discussions and later contribute to other more technical environments (issues and pull requests). An interesting new research direction would be to understand the retention of newcomers who started their journey via GitHub Discussions.

Our findings align with those presented by Silva et al. (2020), who identified and understood what motivates students to participate in Google Summer of Code (GSoC) programs and to continue participating in the projects after the program ends. They suggest that OSS project members should moderate their expectations about gaining long-term contributors. Although GSoC increased participation in OSS projects in general, Silva et al. (2020) found that most OSS projects did not achieve long-term contributors.

Another perspective is the onboarding of contributors with different roles. Although a non-negligible number of people joined the project via discussions and interacted somewhere else later, the vast majority did not "migrate" to other channels. We believe that the GitHub Discussions feature may be attracting people not interested in being part of the development team but want to get involved in another capacity—sometimes only using a channel to have questions answered. In this sense, it would be important to conduct a longer-term

study and investigate the emergence of a subcommunity tied to the project development community.

Overall, our results show that GitHub Discussions provide a structured entry point for newcomers, facilitating initial engagement and, in some cases, enabling transitions to issues and pull requests. This aligns with findings from Silva et al. (2020), who highlight that participation programs like GSoC increase engagement but do not always produce long-term contributors. Similarly, while GitHub Discussions attract newcomers, most do not migrate to other technical channels, suggesting that peripheral participation is a key mode of engagement. Unlike prior work on StackOverflow or Q&A portals (Vasilescu et al., 2014), our findings reveal that integrated discussion spaces within the development environment can foster visibility and low-barrier participation, supporting incremental integration without requiring immediate technical contributions. This underscores the role of GitHub Discussions in shaping newcomer retention and engagement patterns in OSS projects.

How was the project environment impacted? We found a slight decrease in the usage of the more traditional ways of contributing to open-source projects (i.e., through pull requests and issues) after GitHub Discussions were introduced. Understanding the impact of introducing GitHub Discussions is important for practitioners and open-source maintainers, as it helps them anticipate and address potential downsides and decide whether to adopt the feature.

The decrease in issues per month could have happened because GitHub Discussions aimed to accommodate some types of requests that were previously created as issues but are a better fit in the forums. A follow-up study looking specifically at this issue is encouraged to make it possible to understand what exactly changed in the issue reported and how the discussions encompass the topics that were previously posted as issues.

The decrease in the number of issues and pull request after GitHub Discussions align with prior work on task migration in OSS communities (Wang et al., 2023), highlighting that new communication channels can subtly reshape participation patterns. Unlike prior studies that primarily report general usage patterns, our results show that Discussions provide a visible and structured space where newcomers can engage and interact with more experienced contributors, complementing the existing communication channels. Future research should examine whether these effects persist over time and whether similar patterns are observed in smaller or less active projects, which might exhibit different social and technical dynamics.

6 Related Work

This section describes studies related to communication channels in software development, and newcomers in OSS will be briefly discussed.

6.1 Communication Channels in Software Development

The literature presents several studies focusing on communication channels in the context of OSS, including chat, emails, forums, issue trackers, discussion lists, etc. Studies like those conducted by Guzzi et al. (2013) and Panichella et al. (2014) analyzed the behavior of OSS contributions in mailing lists. Guzzi et al. (2013) focused on understanding of development mailing list communication within an OSS project and investigated that implementation details were only discussed in about 35% of topics, in that a number of other topics are discussed. Also, core developers participate in less than 75% of threads. Panichella et al. (2014) studied mailing lists, issue trackers, and IRC chat logs. They looked for an association between the social role of a developer and the evolution of status within a project (newcomers becoming core developers over time).

Some other papers (Linares-Vasquez et al., 2014; Zhu et al., 2022; Zhang et al., 2021) studied Stack Overflow as a communication channel. Stack Overflow is a question-and-answer (Q&A) site widely used by developers to exchange programming-related knowledge through questions, discussions, and answers and provides a means for developers to exchange knowledge. The Q&A process on Stack Overflow creates a crowdsourcing knowledge base that allows developers worldwide to collectively build and improve their knowledge of programming and its related technologies (Zhu et al., 2022).

Linares-Vasquez et al. (2014) investigated how changes that occur in Android APIs trigger questions and activities on Stack Overflow, discovering which topic raises the most questions. Zhu et al. (2022) and Zhang et al. (2021) focused their studies on discussions that occur in Stack Overflow discussion threads, and among the findings, they cite that there is a strong correlation between the number of comments on questions and the response time of the questions, that is, more discussed questions receive answers more slowly; they also found that most comments are informative and therefore can enhance their associated responses from multiple perspectives; and most comment activity occurs after accepting a response. However, none of these cited papers related to Stack Overflow focused on newcomers and/or maintainers during their reviews.

Although different knowledge regarding different communication channels is addressed in the literature, the new resource of GitHub Discussions is a new knowledge base in the OSS community. It is important to observe that, there is a non-negligible literature that refers to discussions on GitHub. These papers to the comment chain on issues and pull requests (Kavaler et al., 2017; Tsay et al., 2014b; Pletea et al., 2014; Kavaler et al., 2019; Viviani et al., 2019) rather than the feature called GitHub Discussions.

Anyway, a few papers focus on GitHub Discussions. Lima et al. (2023a) focused on detecting related posts (duplicate or nearly duplicate posts) in GitHub Discussions. They found that the number of duplicates in non-negligible and that communities would benefit from automatic identification of these messages. Still, Wang et al. (2023) analyzed the phenomenon of converting discussions (on GitHub Discussions) into is-

sues and vice-versa and found that there are multiple reasons to do so.

Hata et al. (2022) presented GitHub Discussions by conducting a study to understand how developers use this new feature and how it impacts development processes. They found that bugs, unexpected behavior, and code reviews are predominant thread categories. There is also a positive relationship between the involvement of project members and the frequency of discussions. While in this paper we also analyze how developers use GitHub Discussions, differently from Hata et al. (2022), we specifically focus on the influence of GitHub Discussions to understand how it modifies the dynamics of issues and pull requests and how it impacts the onboarding of a new way of onboarding of newcomers.

6.2 Newcomers on Open Source Software

OSS communities require developers with specific skills, and accomplishing a task is often a long, multi-step process. Some newcomers may lose motivation and even give up contributing if there are too many hurdles to overcome during this process (Steinmacher et al., 2016). Therefore, newcomers need to learn the social and technical aspects of a project before making a contribution (Steinmacher et al., 2015). They often post their questions and requests for help on forums and mailing lists or send emails to, for example, project maintainers (Park and Jensen, 2009). However, receiving responses that offer no guidance or unpolite responses may result in these newcomers dropping out (Steinmacher et al., 2013). The lack of awareness and guidance during the first steps also discourages new interactions and, consequently, new contributions (Steinmacher et al., 2012).

Previous research has shown that a simple measure of community responsiveness — whether a poster receives a reply — is associated with a higher likelihood of reposting (Joyce and Kraut, 2017) and increased reposting speed (Lampe and Johnston, 2005) and the effect is even stronger for newcomers (Arguello et al., 2006). A response to a newcomer signals acceptance into the group and leads to more committed behavior on the part of the newcomer. Silence is interpreted as rudeness or hostility (Hinds and Kiesler, 2002). Baym (1993) found that responses full of praise are a motivating force for newcomers. Answers signal that the community believes the author of the discussion thread is a valued member worthy of their attention, and the author reciprocates by writing more and responding to other newcomers. Successful conversations turn newcomers into committed contributors who can evolve within projects (Burke et al., 2007).

The success of open software projects depends on the ability to attract new participants. Projects that fail to attract and retain new contributors may not go beyond a few lines of code. Furthermore, as projects progress, the few people who were actively involved are the only ones who understand the full nature of the project. This makes the socialization of new members essential for the long-term survival of open software projects (Zhang et al., 2007).

Given the studies related to newcomers, we notice that communication is key for the onboarding of new members to OSS. A bad reception can make newcomers not return. Our work focuses on GitHub Discussions and how they can influence the development of OSS projects on GitHub, given that this new means of communication can facilitate the onboarding of newcomers.

7 Threats to validity

In this section, we discuss the limitations and potential threats to the validity of our study, their potential impact on the results, and how we have mitigated them (Wohlin et al., 2012). Our selection of projects also limits our results. Therefore, we have a small number of projects with a significant number of discussions, so our results indicate general trends; we recommend running segmented analyses when applying our results to a particular project. Moreover, our findings may not be generalizable to smaller or niche OSS projects, where communication dynamics and contributor interactions can differ significantly from larger or more popular projects.

For replication purposes, we made our data and source code publicly available (Maciel et al., 2025). In the statistical approach, we added several controls that might influence the independent variables to reduce confounding factors. However, in addition to the already identified dependent variables, other factors might influence the activities related to issues, pull requests, and GitHub Discussions. For instance, while RDD is appropriate for analyzing causal effects, we observed declines in issues and pull requests after the adoption of GitHub Discussions. These declines might be influenced by external factors not fully captured in our study, such as project maturity, casual contributions, or other contributor engagement patterns.

One of the constructs in our study is the “first discussion post” as a proxy to the “time of discussions adoption” on a project. A more precise definition of this adoption time would have involved the integration date, which is not provided by the GitHub API. Hence, the validity of the “time of discussions adoption” construct might have been threatened by the definition. We reduce this threat by excluding the period of 15 days immediately before and after adoption from all analyses.

To identify bots accounts in our dataset, we rely on two approaches commonly used in the literature. We first detected official bot integrations that include bot marks visible on GitHub, and then we compared the usernames in our dataset against the list of bots recently curated by Chidambaram et al. (2023). These combined approaches enabled the identification of bot accounts that are very active and, therefore, would distort our findings.

Our study is limited to quantitatively understanding the influence of GitHub Discussions in OSS projects opting for analyzing a broad sample of projects. Another possibility would be going in-depth in the analysis, conducting a focused analysis on specific cases. This could involve analyzing the contents of the discussions and using interviews to further understand the details of specific findings. Our decision was towards understanding the landscape of GitHub Discussions in this first moment, opening avenues for future in-depth research as one can observe in Section 5.

8 Conclusion

Our findings reveal that the introduction of GitHub Discussions on GitHub has brought about notable shifts in project dynamics and contributor engagement. While traditional contributions like issues and pull requests have decreased post-adoption, GitHub Discussions provide an alternative, more informal space for collaboration and knowledge exchange. This shift suggests that contributors, particularly newcomers, are finding value in the interactive and flexible format of GitHub Discussions as a starting point for project involvement.

For open-source software (OSS) maintainers, this change highlights the importance of GitHub Discussions in fostering a more inclusive and interactive community environment. By offering a less formal entry point, GitHub Discussions can enhance the onboarding experience, allowing new contributors to engage with the project in a more approachable manner before transitioning to technical contributions like code and issue tracking. This can lead to more sustainable and meaningful engagement over time.

Future research should explore how GitHub Discussions can be effectively used to attract and onboard contributors, including the potential for increasing diversity within OSS projects. Understanding how different channels, such as issues, pull requests, and GitHub Discussions, contribute to a well-rounded contributor experience can provide OSS leaders with valuable strategies to build healthier, more diverse, and collaborative project ecosystems.

References

Ait, A., Izquierdo, J. L. C., and Cabot, J. (2022). An empirical study on the survival rate of github projects. In *Proceedings of the 19th International Conference on Mining Software Repositories*, page 365–375, New York, NY, USA. Association for Computing Machinery.

Arguello, J., Butler, B. S., Joyce, E., Kraut, R., Ling, K. S., Rosé, C., and Wang, X. (2006). Talk to me: Foundations for successful individual-group interactions in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 959–968, New York, NY, USA. Association for Computing Machinery.

Baym, N. K. (1993). Interpreting soap operas and creating community: Inside a computer-mediated fan culture. *Journal of folklore research*, pages 143–176.

Beyer, S., Macho, C., Di Penta, M., and Pinzger, M. (2020). What kind of questions do developers ask on Stack Overflow? a comparison of automated approaches to classify posts into question categories. *Empirical Software Engineering*, 25:2258–2301.

Borges, H., Hora, A., and Valente, M. T. (2016). Predicting the Popularity of Github Repositories. New York, NY, USA. Association for Computing Machinery.

Burke, M., Joyce, E., Kim, T., Anand, V., and Kraut, R. (2007). Introductions and requests: Rhetorical strategies that elicit response in online communities. In *Communities and Technologies 2007: Proceedings of the Third Communities and Technologies Conference, Michigan State University 2007*, pages 21–39, London, UK. Springer.

Cassee, N., Vasilescu, B., and Serebrenik, A. (2020). The silent helper: The impact of continuous integration on code reviews. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 423–434.

Chidambaram, N., Decan, A., and Mens, T. (2023). A dataset of bot and human activities in github. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pages 465–469.

Cochran, W. (1977). *Sampling Techniques*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Cook, T. and Campbell, D. (1979). *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin.

Dabic, O., Aghajani, E., and Bavota, G. (2021). Sampling projects in GitHub for MSR studies. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 560–564. IEEE.

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

Furtado, A., Andrade, N., Oliveira, N., and Brasileiro, F. (2013). Contributor Profiles, Their Dynamics, and Their Importance in Five Q&A Sites. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, page 1237–1252, New York, NY, USA. Association for Computing Machinery.

Gao, W., Wu, J., and Xu, G. (2022). Detecting duplicate questions in stack overflow via source code modeling. *Int. J. Softw. Eng. Knowl. Eng.*, 32(2):227–255.

Giuffrida, R. and Dittrich, Y. (2013). Empirical studies on the use of social software in global software development – A systematic mapping study. *Information and Software Technology*, 55(7):1143–1164.

Guzzi, A., Bacchelli, A., Lanza, M., Pinzger, M., and Deursen, A. v. (2013). Communication in Open Source Software Development Mailing Lists. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, page 277–286, San Francisco, CA, USA. IEEE Press.

Hata, H., Novielli, N., Baltes, S., Kula, R. G., and Treude, C. (2022). Github discussions: An exploratory study of early adoption. *Empirical Softw. Engg.*, 27(1).

Hinds, P. J. and Kiesler, S. (2002). Attribution in Distributed Work Groups.

Imbens, G. W. and Lemieux, T. (2008). Regression discontinuity designs: A guide to practice. *Journal of Econometrics*, 142(2):615–635. The regression discontinuity design: Theory and applications.

Jin, Y., Bai, Y., Zhu, Y., Sun, Y., and Wang, W. (2022). Code recommendation for open source software developers. *arXiv preprint arXiv:2210.08332*.

Joyce, E. and Kraut, R. E. (2017). Predicting Continued Participation in Newsgroups. *Journal of Computer-Mediated Communication*, 11(3):723–747.

Kafer, V., Graziotin, D., Bogicevic, I., Wagner, S., and Ramadani, J. (2018). Communication in Open-Source

Projects-End of the e-Mail Era? In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, page 242–243, New York, NY, USA. Association for Computing Machinery.

Kamienski, A. V., Hindle, A., and Bezemer, C. (2023). Analyzing techniques for duplicate question detection on q&a websites for game developers. *Empir. Softw. Eng.*, 28(1):17.

Kavaler, D., Devanbu, P., and Filkov, V. (2019). Whom are you going to call? determinants of@-mentions in github discussions. *Empirical Software Engineering*, 24:3904–3932.

Kavaler, D., Sirovica, S., Hellendoorn, V., Aranovich, R., and Filkov, V. (2017). Perceived language complexity in github issue discussions and their effect on issue resolution. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 72–83. IEEE.

Krejcie, R. V. and Morgan, D. W. (1970). Determining sample size for research activities. *Educational and Psychological Measurement*, 30(3):607–610.

Kuznetsova, A., Brockhoff, P. B., and Christensen, R. H. B. (2017). lmertest package: tests in linear mixed effects models. *Journal of Statistical Software*, 82(13).

Lampe, C. and Johnston, E. (2005). Follow the (slash) dot: Effects of feedback on new members in an online community. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '05, page 11–20, New York, NY, USA. Association for Computing Machinery.

Lave, J. and Wenger, E. (2001). Legitimate peripheral participation in communities of practice. In *Supporting lifelong learning*, pages 121–136. Routledge.

Lima, M., Steinmacher, I., Ford, D., Liu, E., Vorreuter, G., Conte, T., and Gadelha, B. (2023a). Looking for related discussions on github discussions. *PeerJ*.

Lima, M., Steinmacher, I., Ford, D., Liu, E., Vorreuter, G., Conte, T., and Gadelha, B. (2023b). Looking for related posts on github discussions. *PeerJ Computer Science*, 9:e1567.

Lima, M., Steinmacher, I., Ford, D., Vorreuter, G., Gonçalves, L., Conte, T., and Gadelha, B. (2025). How are discussions linked? a link analysis study on github discussions. *Journal of Systems and Software*, 219:112196.

Linares-Vasquez, M., Bavota, G., Di Penta, M., Oliveto, R., and Poshyvanyk, D. (2014). How do api changes trigger stack overflow discussions? a study on the android sdk. In *Proceedings of the 22nd International Conference on Program Comprehension*, page 83–94, New York, NY, USA. Association for Computing Machinery.

Maciel, A., Wessel, M., Serebrenik, A., Wiese, I., and Steinmacher, I. (2025). Replication package for this paper. `https://github.com/anamaciel/DiscussionsNewcomersOSS`.

Marlow, J., Dabbish, L., and Herbsleb, J. (2013). Impression formation in online peer production: Activity traces and personal profiles in github. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, page 117–128, New York, NY, USA. Association for Computing Machinery.

Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309–346.

Mondal, S., Saifullah, C. K., Bhattacharjee, A., Rahman, M. M., and Roy, C. K. (2021). Early detection and guidelines to improve unanswered questions on stack overflow. In *14th Innovations in software engineering conference (formerly known as India software engineering conference)*, pages 1–11.

Nakagawa, S. and Schielzeth, H. (2013). A general and simple method for obtaining r2 from generalized linear mixed-effects models. *Methods in ecology and evolution*, 4(2):133–142.

Panichella, S., Bavota, G., Penta, M. D., Canfora, G., and Antoniol, G. (2014). How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 251–260, Victoria, British Columbia, Canada. IEEE.

Park, Y. and Jensen, C. (2009). Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers. In *2009 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 3–10, Edmonton, AB, Canada. IEEE.

Pletea, D., Vasilescu, B., and Serebrenik, A. (2014). Security and emotion: sentiment analysis of security discussions on github. In *Proceedings of the 11th working conference on mining software repositories*, pages 348–351.

Silva, J. O., Wiese, I., German, D. M., Treude, C., Gerosa, M. A., and Steinmacher, I. (2020). Google summer of code: Student motivations and contributions. *Journal of Systems and Software*, 162:110487.

Steinmacher, I., Chaves, A. P., Conte, T. U., and Gerosa, M. A. (2014). Preliminary Empirical Identification of Barriers Faced by Newcomers to Open Source Software Projects. In *Proceedings of the 2014 Ninth International Conference on Availability, Reliability and Security*, page 51–60, USA. IEEE Computer Society.

Steinmacher, I., Conte, T., Gerosa, M. A., and Redmiles, D. (2015). Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work; Social Computing*, CSCW '15, page 1379–1392, New York, NY, USA. Association for Computing Machinery.

Steinmacher, I., Conte, T. U., Treude, C., and Gerosa, M. A. (2016). Overcoming Open Source Project Entry Barriers with a Portal for Newcomers. In *Proceedings of the 38th International Conference on Software Engineering*, page 273–284, New York, NY, USA. Association for Computing Machinery.

Steinmacher, I., Wiese, I., Chaves, A. P., and Gerosa, M. A. (2013). Why Do Newcomers Abandon Open Source Software Projects? In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 25–32, San Francisco, CA, USA. IEEE.

Steinmacher, I., Wiese, I. S., and Gerosa, M. A. (2012). Recommending Mentors to Software Project Newcomers. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, page 63–67, Zurich, Switzerland. IEEE Press.

Storey, M.-A., Zagalsky, A., Filho, F. F., Singer, L., and German, D. M. (2017). How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering*, 43(2):185–204.

Sullivan, G. and Feinn, R. (2012). Using effect size—or why the p value is not enough. *Journal of graduate medical education*, 4:279–82.

Thistlethwaite, D. L. and Campbell, D. T. (1960). Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational psychology*, 51(6):309.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014a). Let's Talk about It: Evaluating Contributions through Discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, page 144–154, New York, NY, USA. Association for Computing Machinery.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014b). Let's talk about it: evaluating contributions through discussion in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 144–154.

Vasilescu, B., Serebrenik, A., Devanbu, P. T., and Filkov, V. (2014). How social q&a sites are changing knowledge sharing in open source software communities. In Fussell, S. R., Lutters, W. G., Morris, M. R., and Reddy, M. C., editors, *Computer Supported Cooperative Work, CSCW '14, Baltimore, MD, USA, February 15-19, 2014*, pages 342–354. ACM.

Viviani, G., Famelis, M., Xia, X., Janik-Jones, C., and Murphy, G. C. (2019). Locating latent design information in developer discussions: A study on pull requests. *IEEE Transactions on Software Engineering*, 47(7):1402–1413.

Wang, D., Kondo, M., Kamei, Y., Kula, R. G., and Ubayashi, N. (2023). When conversations turn into work: A taxonomy of converted discussions and issues in github. *Empirical Software Engineering (EMSE)*.

Wenger, E. (2011). Communities of practice: A brief introduction.

Wessel, M., Serebrenik, A., Wiese, I., Steinmacher, I., and Gerosa, M. A. (2020). Effects of adopting code review bots on pull requests to oss projects. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 1–11.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Zhang, H., Wang, S., Chen, T.-H., and Hassan, A. E. (2021). Reading answers on stack overflow: Not enough! *IEEE Transactions on Software Engineering*, 47(11):2520–2533.

Zhang, J., Ackerman, M. S., and Adamic, L. (2007). Expertise Networks in Online Communities: Structure and Algorithms. In *Proceedings of the 16th International Conference on World Wide Web*, page 221–230, New York, NY, USA. Association for Computing Machinery.

Zhang, Y., Wang, H., Wu, Y., Hu, D., and Wang, T. (2020). Github's milestone tool: A mixed-methods analysis on its use. *Journal of Software: Evolution and Process*, 32(4):e2229.

Zhao, Y., Serebrenik, A., Zhou, Y., Filkov, V., and Vasilescu, B. (2017). The impact of continuous integration on other software development practices: a large-scale empirical study. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 60–71. IEEE Press.

Zhu, W., Zhang, H., Hassan, A. E., and Godfrey, M. W. (2022). An Empirical Study of Question Discussions on Stack Overflow. *Empirical Softw. Engg.*, 27(6).