# Investigating the Impact of GitHub Discussions on Maintainers' Workload and Community Dynamics

Ana Maciel
State University of Maringá
Maringá, Brazil
anamaciel@prof.unipar.br

Mairieli Wessel
Radboud University
Nijmegen, Netherlands
mairieli.wessel@ru.nl

Igor Wiese
Federal Technological University of Paraná
Campo Mourão, Brazil
igor@utfpr.edu.br

Igor Steinmacher
Northern Arizona University
Flagstaff, EUA
igor.steinmacher@nau.edu

## ABSTRACT

The introduction of GitHub Discussions, a space for informal communication on GitHub, offers open-source communities a mechanism for engaging contributors beyond traditional issue tracking and code review. While promising, the specific effects of GitHub Discussions on project sustainability, contributor integration, and collaboration dynamics remain underexplored. This study investigates the impact of GitHub Discussions across four high-activity OSS projects by examining (1) changes in maintainers' workload, (2) newcomer participation and progression from GitHub Discussions to technical contribution, and (3) the evolving structure of social and technical roles in project collaboration networks. Using data from the GitHub API, we collected and analyzed issues, pull requests, and discussions over a 48-month period. We applied linear regression and longitudinal social network analysis to evaluate participation trends and role centrality. Our results show that GitHub Discussions had limited impact on overall maintainer workload but served as a valuable entry point for newcomers, with a subset transitioning into impactful technical contributors. Nearly 80% of these newcomers had pull requests accepted, and many became active reviewers. In addition, GitHub Discussions enabled a shift in collaboration patterns, with non-core contributors increasingly occupying central positions in project networks. These findings underscore the strategic role of GitHub Discussions in supporting more inclusive, distributed, and sustainable OSS collaboration. GitHub Discussions help lower entry barriers, foster long-term contributor engagement, and redistribute influence beyond core teams. Future research should explore how discussion dynamics influence project governance, contributor retention, and the integration of non-code contributions into project health metrics.

## KEYWORDS

Open Source, Communication, Discussions, GitHub

## 1 Introduction

Open-source software (OSS) projects thrive on collaboration, relying on a distributed and voluntary community of developers to contribute to the development and maintenance of software [15, 20]. This collaborative model fosters innovation, enables rapid problem-solving, and supports the production of high-quality software. However, the sustainability of OSS communities is continually challenged by the increasing complexity of coordination work and the social dynamics required to manage contributions effectively [6, 38].

Project maintainers, in particular, face a growing burden. They must manage both technical responsibilities—such as reviewing code, triaging issues, and implementing new features—and community-oriented tasks, including welcoming newcomers, answering questions, mentoring contributors, and managing conflicts [30]. These responsibilities often go unrecognized yet are critical to a project's health and continuity.

To support these demands, GitHub introduced Discussions in 2020, a feature designed to foster asynchronous and community-driven communication. Discussions provide a space for users and contributors to ask questions, propose ideas, share experiences, and discuss topics that do not fit neatly into issues or pull requests [7, 12]. By decoupling informal interaction from formal development workflows, Discussions have the potential to enhance inclusivity, reduce entry barriers, and promote sustainable contribution practices [9].

To fill this gap, this study aims to investigate the strategic role of GitHub Discussions in supporting the sustainability of OSS projects. Specifically, we seek to understand how Discussions affect three critical dimensions of OSS communities: (1) the workload of maintainers, (2) the onboarding and progression of newcomers, and (3) the structural evolution of collaboration networks. Thus, our goal is to uncover **how Discussions reshape participation, redistribute responsibilities, and contribute to community resilience**.

One key question is whether the introduction of GitHub Discussions leads to changes in the activity of project maintainers and contributors. It remains unclear whether this feature shifts the nature of interactions within the project or alters maintainers' workload. Some researchers [2, 8] argue that informal tools like GitHub Discussions may complement existing practices by providing a space for non-technical interactions, such as clarifying questions or discussing broader issues, which are often overlooked in traditional issue trackers or pull request systems [28].

Despite their potential, the empirical impact of Discussions on OSS project dynamics remains largely unexplored. Existing research has concentrated on traditional artifacts like pull requests, issues, and commits [14, 17, 31], providing limited insight into how informal tools influence collaboration, contributor engagement, and

the evolution of project communities. This study addresses this gap by examining the role of GitHub Discussions in shaping OSS sustainability through three core lenses: (1) maintainers' workload, (2) the onboarding and progression of newcomers, and (3) structural changes in project collaboration networks. Drawing on longitudinal data from four high-activity OSS projects, we analyze interactions across Discussions, issues, and pull requests to uncover how these channels complement each other and reconfigure participation patterns.

Our findings reveal that while Discussions do not reduce the need for maintainer engagement in traditional channels, they create an accessible space for new contributors to ask questions, build relationships, and gradually transition into more technical roles. We observe a shift in the social structure of these projects, with participants active in Discussions emerging as influential nodes in the collaboration network. These contributors often facilitate knowledge exchange and act as bridges between newcomers and core developers.

By integrating informal communication into the fabric of OSS development, Discussions enable distributed leadership and foster inclusive participation—key attributes of resilient OSS communities. This work contributes novel empirical evidence to the OSS literature and offers actionable guidance for practitioners seeking to manage community engagement, reduce maintainer burnout, and support the contributor pipeline through socially-oriented platform features.

## 2 Related Work

The role of communication tools in improving team productivity and collaboration has been extensively studied. For instance, Vasilescu et al. [32] analyzed collaboration patterns on GitHub and highlighted the importance of informal communication channels in fostering innovation. However, their study predates the introduction of GitHub Discussions and does not focus on the maintainers' experience in balancing technical responsibilities with community engagement.

Hu et al. [13] examined the phenomenon of multi-threaded discussions across GitHub issues, exploring the challenges of managing interconnected conversations in large projects. While their work sheds light on coordination complexity, it does not consider how centralized platforms like GitHub Discussions might help mitigate these challenges.

Storey et al. [29] investigated newcomers' motivations and challenges in OSS projects, emphasizing the importance of supportive environments for onboarding contributors. Although their study does not explicitly examine GitHub Discussions, it highlights the potential of such tools in reducing cognitive and social barriers for new contributors.

More closely aligned with the focus of this paper, Hata et al. [12] conducted an exploratory study on the early adoption of GitHub Discussions, analyzing its usage and perception among developers. Their findings indicate that GitHub Discussions are valuable for community building and knowledge sharing within OSS repositories, but also reveal challenges in choosing the most appropriate channel (e.g., issues versus discussions) for a given topic.

Wang et al. [33] further explored this challenge by empirically characterizing developers' reasons for converting discussion posts into issues and vice versa. Their results suggest that GitHub Discussions often act as an incubator for identifying bugs and technical problems, thereby serving as an effective mechanism for onboarding new contributors.

Lima et al. [17] developed and evaluated an approach to automatically identify duplicate discussion posts using a Sentence-BERT pre-trained general-purpose model, achieving high precision (77–100%) and reasonable recall (66%).

More recently, Lima et al. [18] investigated the scope and intent behind link sharing on GitHub Discussions. Their findings show that users frequently share both internal and external resources, with internal links often referencing issues, pull requests, and other discussions, while external links typically point to project documentation, Stack Overflow, and video platforms such as YouTube.

While prior studies offer valuable insights into the use and impact of GitHub Discussions, most have examined specific use cases—such as content duplication [17], link-sharing behavior [18], or the transition between issues and discussions [33]—or have focused on the feature's early adoption and perceived benefits [12]. However, these works typically emphasize isolated functionalities and user behavior without systematically analyzing how Discussions affect broader project dynamics. We note a lack of a longitudinal perspective on how Discussions influence core aspects of OSS sustainability, such as maintainers' workload, newcomer integration, and shifts in collaborative structures. Our study addresses this gap by examining Discussions as a socio-technical mechanism that reshapes interaction patterns, distributes leadership roles, and facilitates sustained engagement, offering a deeper understanding of how informal communication channels can support resilient OSS communities.

## 3 Research Method

To investigate how GitHub Discussions reshape participation, redistribute responsibilities, and contribute to community resilience in OSS projects, we adopt a mixed-methods approach combining quantitative analysis of project activity logs with social network analysis (SNA). We conducted a longitudinal, repository-level study of four high-activity OSS projects that adopted GitHub Discussions between 2020 and 2022. Our analysis integrates data from GitHub Discussions, issues, and pull requests to track contributor behavior, role evolution, and maintainer activity over time.

This section presents the guiding research questions, followed by details about data collection and analytical procedures used to address them.

### 3.1 Research Questions

GitHub Discussions were introduced to provide a more flexible and inclusive space for asynchronous communication within OSS projects [12]. As a platform-integrated feature, they may reshape collaboration dynamics, reduce cognitive entry barriers for newcomers, and reconfigure how contributors engage with the project and each other. Understanding these effects is essential to assess how GitHub Discussions contribute to the long-term sustainability and governance of OSS communities. We investigate these potential transformations through the following research questions:

**Research Question 1**

How have maintainers' workloads changed after the introduction of GitHub Discussions?

In this first RQ, we examine how maintainers' workload in pull requests and issues has shifted after the introduction of GitHub Discussions. The decision to use pull requests and issues as indicators was made to track potential shifts in the volume of activities that would require maintainer intervention [5, 7], as a higher number of issues and pull requests would likely correspond with an increased workload in terms of comments, reviews, or coordination efforts by the maintainers.

**Research Question 2**

How are newcomers who joined through GitHub Discussions participating in issues and pull requests?

This research question investigates whether GitHub Discussions serve as an effective entry point for newcomers' onboarding. By observing how newcomers transition into formal development tasks—such as participating in issues and pull requests—we assess whether informal engagement leads to deeper, more sustained contributions within the project.

**Research Question 3**

How does the participation dynamic develop among the different contributor roles within the networks formed by GitHub Discussions?

Here, we examine whether GitHub Discussions foster more distributed forms of leadership. By analyzing social interaction patterns, we investigate the emergence of new central figures in the community. The analysis aims to determine if new members have taken on responsibilities—such as facilitating or managing discussions—potentially reducing the workload of project maintainers in GitHub Discussions

## 3.2 Data Collection

To select our study sample, we initially considered a list of 98 projects identified by Hata et al. [12]—excluding seven projects that had discontinued using GitHub Discussions, making the data inaccessible. From the remaining 91 projects, we narrowed our focus to the four (4) projects with the highest volume of discussions during a one-year observation period following their initial adoption of the feature during the data collection period. We selected these projects to ensure the analysis focused on projects with sufficient activity to provide meaningful insights into the impact of GitHub Discussions.

The four GitHub projects studied showcase diverse domains and collaboration dynamics, offering a comprehensive view of open-source development:

- `vercel/next.js`: started using GitHub Discussions on 2020-01-15, next.js is a popular React framework focusing on performance and scalability.
- `ImageMagick/ImageMagic`: With GitHub Discussions adopted on 2020-03-30, the project is an established image-processing library.
- `livewire/livewire`: began using Discussions on 2020-05-15, livewire is a framework that simplifies the integration between the backend and frontend in Laravel applications.
- `fastlane/fastlane`: Discussions were introduced on 2020-07-09, fastlane is a tool for automating mobile app deployments.

These projects exemplify varied collaborative structures across software ecosystems, including differences in governance models (e.g., individual-led versus organization-backed), contributor engagement patterns, and levels of technical complexity and modularity in development workflows. The descriptive statistics for the selected projects are provided in Table 1. The table includes the number of issues, pull requests, total of core developers, discussions, comments in discussions, and unique participants in discussions.

To gather data, we employed a dual approach. Issues and pull requests were collected via the GitHub REST API [1], a well-established interface for retrieving data from GitHub. To collect discussion data, we leveraged GraphQL, a versatile query language that facilitates data extraction from APIs.

## 3.3 Data Analysis

This section presents the data analysis conducted to address the research questions. The analysis is structured to provide insights into changes in maintainers' workload related to issues and pull requests (PRs), as well as the broader implications of introducing GitHub Discussions. Each research question is addressed individually, supported by quantitative metrics and longitudinal data collected over the study period.

*3.3.1 RQ1 – Changes in Maintainers' Workloads After the Introduction of GitHub Discussions* To examine the impact of GitHub Discussions on maintainers' workload, we applied a linear regression model [16, 23]. Linear regression is a statistical technique used to model the relationship between a dependent variable (response) and one or more independent variables (predictors). The objective is to estimate a linear equation describing how the response variable changes with the predictors [22, 25].

The linear regression equation used is given by:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon$$

where $y$ represents the dependent variable, $x_1, x_2, \ldots, x_k$ are the independent variables, $\beta_0$ is the intercept, $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients of the independent variables, and $\epsilon$ is the error term, accounting for variability not explained by the model.

Each project was modeled independently. For each one, we fitted two separate regression models: one with the monthly number of *pull requests* created and another with the monthly number of *issues* created as the dependent variable. Modeling issues and PRs separately enables a clearer interpretation of trends and reflects the distinct dynamics in how contributors engage with these artifacts. The independent variables included *time* (measured in 30-day intervals) and a binary indicator representing the *introduction of GitHub Discussions*. These models allowed us to assess correlations between the adoption of GitHub Discussions and changes in maintainers' engagement with issues and PRs.

We did not normalize the data prior to modeling. This decision was motivated by three factors: (i) absolute values for monthly

[1] https://docs.github.com/en/rest

| Project | #Issues | #PRs | #Core Developer* | #Discussions | #Discussions Comments | #Discussions Participants |
|---|---|---|---|---|---|---|
| vercel/next.js | 22k | 26k | 31 | 21k | 68k | 13k |
| ImageMagick/ImageMagick | 4k | 1k | 10 | 4k | 19k | 2k |
| livewire/livewire | 1k | 3k | 17 | 6k | 13k | 2k |
| fastlane/fastlane | 13k | 8k | 34 | 3k | 3k | 1k |

* contributors responsible for roughly 80% of the commits in the project [21]

**Table 1: Numbers of our subject projects.**

issues and PRs are more interpretable for practitioners, (ii) normalization could obscure meaningful magnitude changes, and (iii) intra-project analysis ensures comparability within each dataset.

We conducted a two-phase analysis to capture both short- and long-term impacts. In phase one, we compared maintainer activity during two 12-month periods: before and after the introduction of GitHub Discussions. In phase two, we extended the window to 24 months post-adoption, divided into two successive 12-month periods. This phased comparison helped identify temporal trends and lasting shifts in workload. Figure 1 illustrates the analysis timeline.

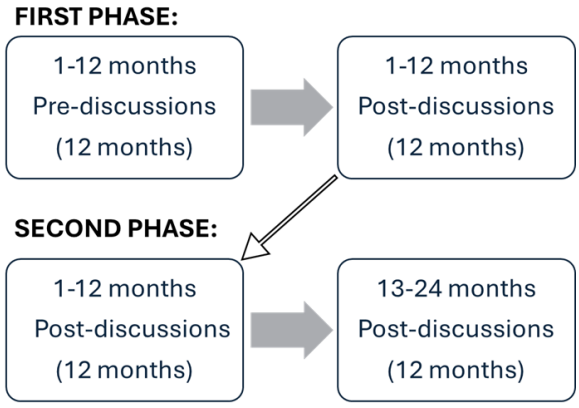



**Figure 1: Timeline of the two-phase analysis.**

To complement the individual project analyses, we conducted a combined analysis using a **linear mixed model (LMM)**, incorporating random intercepts for each project to account for between-project variability while estimating the overall effect of introducing GitHub Discussions on maintainers' workload. This approach allowed us to evaluate the overall impact of introducing GitHub Discussions on maintainers' workload while accounting for variability between projects.

*3.3.2 RQ2 – Participation of New Contributors from Discussions in Issues and Pull Requests* For this question, we identified contributors whose first involvement in the project was through GitHub Discussions, verified by checking that they had no prior recorded activity in commits, issues, or pull requests. To examine contributors who transitioned from social engagement to technical contributions, we focused on those who, at any point after their first interaction, demonstrated sustained technical activity, operationalized as submitting at least three pull requests within a single month. This threshold was selected to capture contributors who transitioned from occasional to sustained technical participation. While somewhat conservative, it is grounded in prior empirical work on contributor retention and workload patterns, ensuring that only contributors showing meaningful engagement were considered by submitting at least three pull requests within a single month. This subset allowed us to analyze contributors' progression in technical tasks. Using timeline visualizations, we tracked their participation in issues and pull requests over 48 months, categorizing their contributions based on their most frequent activity type—such as opening issues, submitting pull requests, conducting reviews, or engaging in comments. This approach allowed us to analyze the progression of contributors from initial engagement to sustained technical involvement.

*3.3.3 RQ3 – Role and Position Changes in GitHub Discussions Over Time* To analyze how GitHub Discussions influence contributor roles and centrality, we constructed monthly interaction networks spanning 48 months. Each network included contributors active in issues, PRs, or GitHub Discussions during that month. Nodes represented contributors, and directed edges captured interactions—specifically, from post authors to commenters within GitHub Discussions.

We computed two standard centrality metrics [1, 10, 34, 36]:

**Closeness Centrality:** Indicates how quickly a contributor can reach others in the network. Higher closeness that a contributor can reach others with fewer intermediary steps, potentially reflecting increased influence or leadership roles.

**Betweenness Centrality:** Measures how often a contributor lies on the shortest path between others, highlighting their role as a bridge in communication flow—facilitating or shaping the flow of ideas and decisions.

After creating the networks, we classified contributors according to their roles, which were reassigned dynamically on a monthly basis. This means a contributor's role could change over time depending on their most recent activity involving commits, pull requests, and issues in that specific month in the project based on their activities involving commits, pull requests, and issues on GitHub. The roles were defined as follows:

- **Core developers**: contributors responsible for roughly 80% of the commits in the project [21].
- **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**: contributors who interacted exclusively via discussions (posts, comments, or replies).

We focused on the top 10 contributors to further analyze dynamics when assessing the centrality metrics. These contributors represent users from any of the above groups who rank among the top 10 for the respective metrics. These sub-graphs provided a focused view of the most influential contributors over time.

To gain a deeper understanding of the dynamic nature of contributor influence, we conducted a longitudinal analysis using a sliding 12-month window [4], a choice guided by prior work in longitudinal network analysis and the need to capture meaningful contributor behavior over extended periods while balancing temporal resolution and computational complexity. This approach, where a fixed-length window moves sequentially across the timeline, allowed us to track changes in engagement patterns and the evolving roles of contributors within the network over 48 months. By advancing the window one month at a time, we identified individuals whose influence was shifting, revealing the degree of stability or fluidity within contributor participation. Figure 2 illustrates this approach, where the window moves sequentially across the timeline, capturing the evolving composition of the top 10 contributors every 12 months.

## 4 Results

In this section, we report our results per research question.

### 4.1 Changes in Maintainers' Workloads After the Introduction of GitHub Discussions (RQ1)

We assessed changes in maintainers' workload by analyzing the monthly number of issues and pull requests (PRs) created before and after the introduction of GitHub Discussions. As outlined in the method, we conducted two analyses. In the first, we compared activity during the 12 months before GitHub Discussions with the 12 months immediately following their adoption. In the second phase, we focused on the long-term effects by comparing two consecutive post-adoption periods: months 1–12 and months 13–24 after GitHub Discussions were introduced.

Separate analyses were conducted for issues and PRs per project. Table 2 summarizes the results, presenting the $R^2$ and p-values for each of the four projects across both phases. These results highlight how maintainer workload evolved over time, revealing project-specific differences and shifts in activity patterns.

| **ISSUES** | | | | |
|---|---|---|---|---|
| | **First Phase** | | **Second Phase** | |
| **Project** | **$R^2$** | **p** | **$R^2$** | **p** |
| vercel/next.js | 0.067 | 0.480 | 0.332 | $0.014^{*}$ |
| ImageMagick/ImageMagick | 0.138 | 0.211 | 0.102 | 0.321 |
| Livewire/Livewire | 0.486 | $0.001^{***}$ | 0.381 | $0.006^{**}$ |
| fastlane/fastlane | 0.307 | $0.021^{*}$ | 0.504 | $0.001^{***}$ |
| All projects | 0.010 | 0.140 | 0.025 | 0.121 |
| **PULL REQUESTS** | | | | |
| | **First Phase** | | **Second Phase** | |
| **Project** | **$R^2$** | **p** | **$R^2$** | **p** |
| vercel/next.js | 0.050 | 0.585 | 0.350 | $0.011^{*}$ |
| ImageMagick/ImageMagick | 0.157 | 0.167 | 0.129 | 0.237 |
| Livewire/Livewire | 0.393 | $0.005^{**}$ | 0.363 | $0.008^{**}$ |
| fastlane/fastlane | 0.369 | $0.007^{**}$ | 0.457 | $0.001^{***}$ |
| All projects | 0.0493 | 0.081 | 0.029 | 0.465 |

**Table 2: Regression results of discussions and maintainer workload. Significant p-values are indicated as follows: $^{*}$ p < 0.05, $^{**}$ p < 0.01, $^{***}$ p < 0.001.**

*4.1.1 Issues* For issues, two projects—`Livewire` and `fastlane`—exhibited statistically significant changes in both analyses. During the first phase, `Livewire` showed a strong association between the introduction of GitHub Discussions and a change in issue volume ($R^2$ = 0.486, p < 0.001), while `fastlane` presented a moderate but significant effect ($R^2$ = 0.307, $p < 0.05$). In the second phase, these effects strengthened, particularly for `fastlane` ($R^2$ = 0.504, $p <$ 0.001). These findings suggest that Discussions may have helped redistribute or reduce maintainers' engagement with issues over time in these communities.

Interestingly, `vercel/next.js` showed no significant change in the first phase but revealed a delayed effect in the second ($R^2$ = 0.332, $4p < 0.05$), possibly indicating a slower uptake or community adaptation to the feature. In contrast, ImageMagick showed no significant changes in either phase, suggesting a limited or inconsistent impact of Discussions on issue volume. At the aggregate level, the models for all projects combined did not yield statistically significant results, reflecting the importance of analyzing projects individually due to substantial contextual variation.

*4.1.2 Pull Requests (PRs)* Results for pull requests follow a similar pattern. `Livewire` and `fastlane` again showed significant effects in both phases. In Phase 1, `Livewire` exhibited a moderately strong association ($R^2 = 0.393, p < 0.01$), while `fastlane` also demonstrated a robust effect ($R^2 = 0.369, p < 0.01$). These effects persisted and slightly strengthened in Phase 2, with `fastlane` reaching $R^2 = 0.457$ ($p < 0.001$). This suggests that Discussions may have contributed to more coordinated or streamlined PR activity, potentially by providing contributors a space to clarify questions before initiating formal contributions.

`vercel/next.js` again showed a delayed but significant association only in the second phase ($R^2 = 0.350, p < 0.05$), mirroring the pattern seen in issues. `ImageMagick/ImageMagick`, by contrast, remained statistically unchanged across both phases. At the aggregated level, no significant associations were observed for PRs, reaffirming that the impact of GitHub Discussions is highly dependent on individual project dynamics and community behaviors.

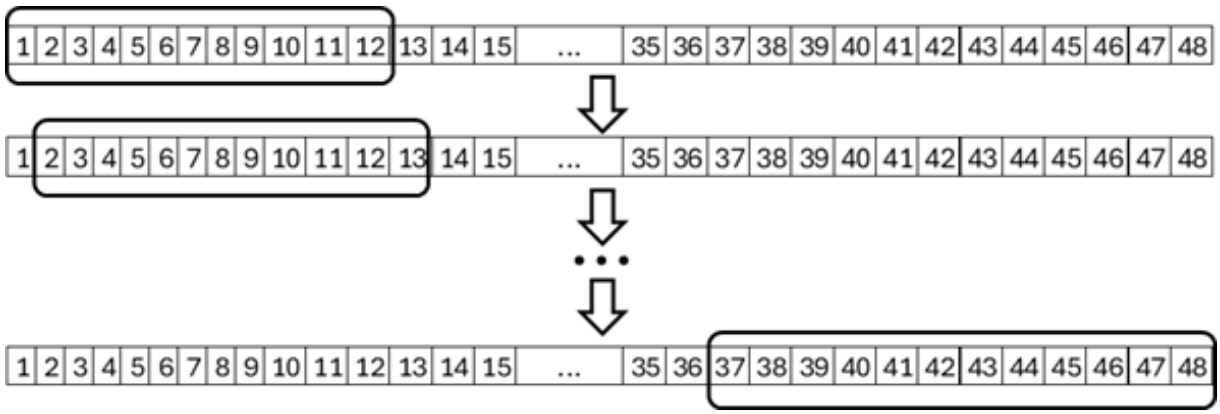



**Figure 2: Example of sliding window technique.**

**Research Question 1**

*How have maintainers' workloads changed after the introduction of GitHub Discussions?*

The impact of GitHub Discussions on maintainers' workload varied across projects. Significant changes in issue and pull request activity were observed in `Livewire/Livewire`, `fastlane/fastlane`, and `vercel/next.js`, especially in the second phase (months 13–24). No significant effects were found in `ImageMagick/ImageMagick`. These results suggest that the influence of GitHub Discussions tends to increase over time and is project-dependent.

## 4.2 Participation of Members Who Joined Through GitHub Discussions (RQ2)

We analyzed the progression of contributors who initially engaged with a project through GitHub Discussions and later transitioned into technical roles. From a dataset of 28,245 users who first interacted via GitHub Discussions, we identified 153 contributors who submitted at least three pull requests within a single month at any point during the observation period—an operational threshold for sustained technical participation (in issues and/or PRs).

The distribution of these contributors across projects varied considerably. `vercel/next.js` and `Livewire/Livewire` accounted for the majority, with 62 and 80 contributors, respectively. In contrast, only 6 contributors in `ImageMagick/ImageMagick` and 5 in `fastlane/fastlane` met the criteria. These differences highlight the varying levels of contributor activity and engagement across projects, which may reflect differences in community size, project complexity, or the nature of contributions.

Among the 153 contributors analyzed, 122 (79.7%) had at least one pull request accepted, resulting in 888 accepted pull requests. This high acceptance rate suggests that contributors who joined through GitHub Discussions and actively participated in pull requests were likely to make successful contributions. Additionally, 47 users (30.7%) reviewed at least one pull request, totaling 597 reviews, and 44 users (28.8%) provided feedback or engaged in discussions on pull requests, generating 704 pull request comments. These figures suggest that contributors entering through GitHub Discussions advance to submitting code and participate in collaborative review and feedback processes.

Figure 3 shows the distribution of PRs opened by these contributors. Most contributors opened fewer than 10 PRs, with the highest concentration between 1 and 5. However, the distribution features a long tail, with a few contributors submitting 50 or more PRs. These outliers indicate that GitHub Discussions may serve as a gateway for casual contributions [27] but also for onboarding highly active participants who assume central roles in project development.

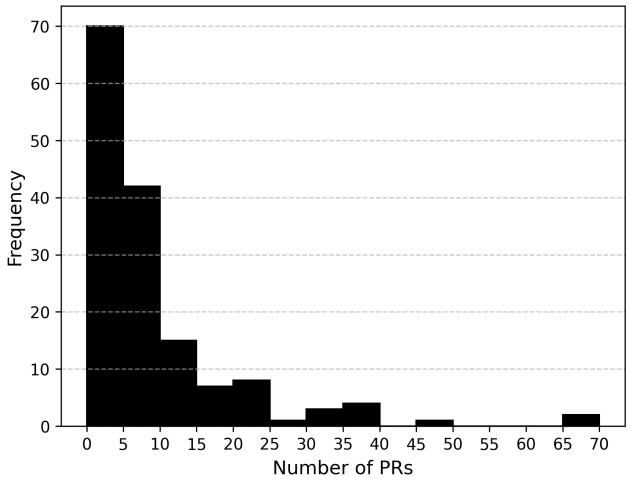



**Figure 3: Distribution of Pull Requests (PRs) opened by contributors entering through GitHub Discussions.**

Overall, the results demonstrate that GitHub Discussions can function as an effective entry point for technical contribution. While most contributors make only a few PRs, a substantial proportion reaches higher levels of engagement. The presence of reviewers and commenters among this cohort further supports the role of GitHub Discussions in facilitating meaningful and sustained participation in OSS projects.

**Research Question 2**

*How are newcomers who joined through GitHub Discussions participating in issues and pull requests?*

While most contributors who joined through GitHub Discussions made only a few technical contributions, a subset demonstrated sustained engagement, including high PR activity, reviews, and feedback. Nearly 80% had at least one PR accepted, and some became highly active contributors. These findings indicate that GitHub Discussions can serve as an effective entry point for newcomers who go on to play technical and collaborative roles in the project.

## 4.3 Evolution of Contributor Roles and Network Positions in GitHub Discussions (RQ3)

To understand how contributor roles evolve within networks shaped by GitHub Discussions, we analyzed the top 10 contributors in terms of betweenness and closeness centrality values over 48 months using a sliding window approach. Contributors were categorized by role: *core developers*, *peripheral developers*, *issue reporters*, and *discussion-only contributors*. Figure 4 presents the number of contributors per role within the top 10 for each centrality metric across 37 sliding windows (from months 1-12 to months 37-48). Below, we detail the observed trends per project.

**`vercel/next.js`.** *Discussion-only contributors* were consistently present among the top contributors in both betweenness and closeness centrality. Their role became especially prominent after window 30, where they frequently occupied up to four top-10 positions in betweenness and more than five in closeness. This sustained presence suggests a growing and stabilizing influence in the project's communication network. Notably, several discussion contributors reappeared across multiple windows, reinforcing their importance in maintaining network cohesion. Meanwhile, *core developers* showed a marked decline in centrality after window 20, indicating a redistribution of influence. Peripheral and issue reporters remained peripheral, with occasional entries into top rankings.

**`ImageMagick/ImageMagick`.** *Discussion-only contributors* dominated both centrality metrics throughout most of the timeline. In closeness centrality, they consistently held more than four top positions from window 13 onward. Although *core developers* had minimal presence, one individual repeatedly appeared in the top 10, indicating localized but persistent influence. A temporary surge of *issue reporters* between windows 13–24 briefly shifted dynamics, but their presence was not sustained. The project's network appears to be largely driven by stable, recurring discussion participants.

**`livewire/livewire`.** *Peripheral developers* were the dominant role throughout much of the timeline, frequently occupying 6 to 9 positions in betweenness centrality between windows 10–30. This suggests an unusually active layer of contributors bridging between core and occasional participants. Two *core developers* appeared consistently but occupied fewer positions, often remaining static figures. From window 26 onward, a visible decline in peripheral dominance was followed by the rise of *discussion-only contributors*, who began to appear more prominently and consistently in both centrality metrics. This reflects a shift from established intermediaries to emerging voices gaining network importance.

**`fastlane/fastlane`.** Initially, *core developers* featured prominently in betweenness centrality during windows 1–14, and resurged again after window 17, maintaining more than four top positions. In closeness centrality, core contributors maintained a moderate early presence. From window 20 onward, *discussion-only contributors* steadily gained ground, eventually reaching 4 to 7 of the top 10 positions by the end of the timeline. The presence of repeating discussion contributors highlights a transfer of connective and mediating influence away from the core team and toward engaged community members.

### 4.3.1 *Summary of Trends*

- **Core developers**: showed stability with a slightly declining presence (in terms of connectivity) in top centrality positions, with influence often limited to one or two recurring individuals.
- **Discussion-only contributors**: maintained strong and increasing influence, especially in closeness centrality. Many reappeared across windows, suggesting sustained engagement and growing network centrality.
- **Peripheral developers**: showed consistent stability in their connections and central influence; played a prominent role primarily in `livewire`, serving as important intermediaries.
- **Issue reporters**: appeared sporadically, with brief spikes in specific time windows, especially in `ImageMagick`.

These results highlight the evolution of roles in the context of using GitHub Discussions and their implications for collaboration and engagement within projects.

**Research Question 3**

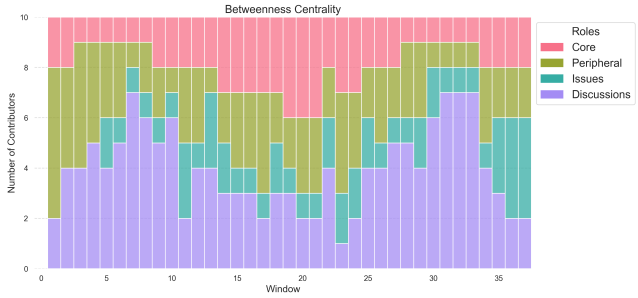
*How does the participation dynamic develop among the different contributor roles within the networks formed by GitHub Discussions?*

Participation dynamics in GitHub Discussions-based networks evolved over time, with *discussion-only contributors* increasingly assuming central roles. They appeared consistently in the top 10 for betweenness and closeness centrality—often with the same individuals reappearing across windows—highlighting their sustained influence. Meanwhile, *core developers* retained some influence but became less prominent, with only a few recurring individuals maintaining central positions. These patterns reflect a redistribution of coordination and connectivity roles, suggesting that GitHub Discussions foster more distributed and stable leadership in OSS communities.

# 5 Discussion

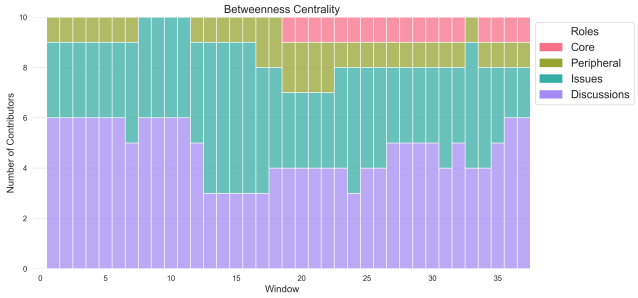
This section discusses the findings that address our research questions, highlighting our results' practical and theoretical implications.

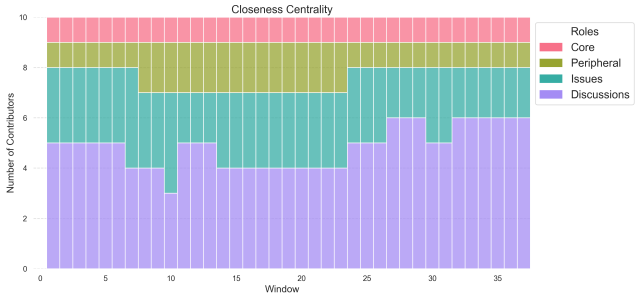
**How Did Maintainers' Practices Adapt to GitHub Discussions?** Our analysis indicates that the introduction of GitHub Discussions has influenced maintainers' interactions within issues and pull requests, as evidenced by an increased variance ($R^2$) between



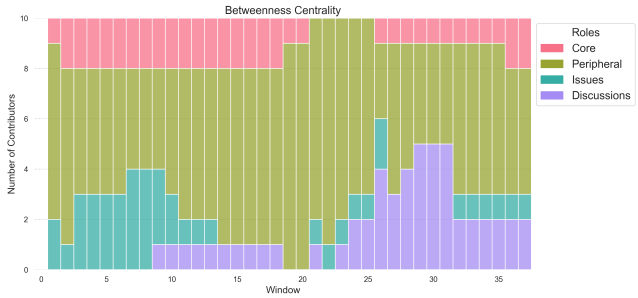


**(a) vercel/next.js**



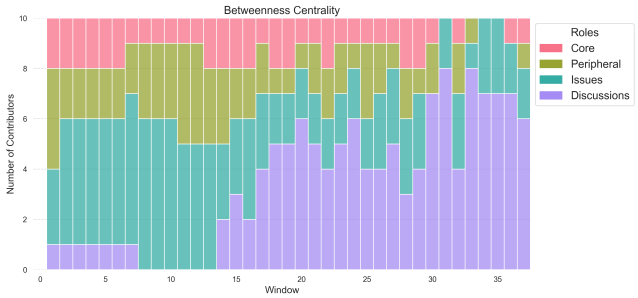




**(b) ImageMagick/ImageMagick**





**(c) livewire/livewire**





**(d) fastlane/fastlane**

**Figure 4: Contributors per role in the top 10 of Betweenness and Closeness centrality across 48-month sliding windows.**

phases. This suggests that discussions became a more relevant factor for maintainer workload over time. However, the total number of issues and pull requests opened remained relatively stable, indicating that the shift primarily affected the nature of engagement rather than its volume. However, maintainers also engaged in GitHub Discussions, which suggests that this feature may have introduced an additional channel for communication. While GitHub Discussions could help structure informal or preparatory conversations, they may also represent another space that maintainers need to monitor and manage alongside issues and pull requests.

Rather than replacing existing collaboration mechanisms, GitHub Discussions seem to have introduced an additional space for engagement. Maintainers might leverage discussions to address peripheral topics, such as user inquiries, brainstorming ideas, or resolving

community disputes, while issues and pull requests remain central to technical contributions. This aligns with previous studies suggesting that informal communication tools tend to supplement rather than replace technical tasks [12, 29].

Other factors, such as shifts in project priorities or variations in contributor activity levels, may have also influenced this growth, making it difficult to isolate the direct impact of GitHub Discussions on maintainers' workload. A more granular analysis focusing on activities such as comments and reviews could provide further insights into how maintainers balance their technical responsibilities with community management duties.

**From Dialogue to Development: Unlocking Newcomer Potential Through GitHub Discussions** GitHub Discussions have emerged as an informal communication tool that can facilitate onboarding newcomers to open-source projects. Our analysis revealed that while some contributors who initially engaged through GitHub Discussions eventually transitioned to technical activities—such as opening issues, submitting pull requests, and reviewing code—this transition was less frequent than suggested by previous studies [13, 24]. Only a subset of new contributors who joined through Discussions engaged in technical activities, with an average transition time of approximately 3.5 months after initial participation.

These findings suggest that GitHub Discussions can serve as an entry point for new contributors, but their effectiveness as a technical onboarding mechanism remains limited. The transition from GitHub Discussions to technical contributions appears to depend on additional factors, such as the availability of mentorship or low-complexity tasks that help newcomers gain confidence in contributing. Future research could explore ways to facilitate this transition, potentially through mentorship programs integrated into GitHub Discussions or the creation of targeted beginner-friendly tasks.

Our results show that GitHub Discussions support contributor onboarding by offering an initial entry channel into projects, enabling some participants to transition into technical roles and actively engage in development. This engagement is reflected in participation metrics, such as pull request acceptance rates and involvement in reviews and comments. The findings highlight the value of fostering Discussions as a gateway to meaningful participation and long-term engagement in software projects.

Overall, these insights illustrate the dual role of GitHub Discussions as both a social entry point and a bridge to technical contributions, lowering barriers for newcomers and empowering them to play an active role in project development. By nurturing this progression, Discussions contribute to open-source projects' long-term sustainability and collaborative success.

**Roles and Network Dynamics in GitHub Discussions: Analysis and Implications** The analysis of centrality metrics over 48 months revealed interesting changes in the dynamics of collaboration within the projects. Core developers maintained a consistent leadership position in the early stages of the projects, but their presence became less consistent over time, particularly in later windows. This decline in consistency suggests a redistribution of influence within the community as other contributor groups, such as discussion-only contributors, increasingly take on more active roles in mediating and facilitating interactions. The growing prominence of discussion-only contributors highlights their rising importance in shaping the network dynamics, while the reduced consistency of core developers indicates a shift in the traditional leadership structure of the projects.

Discussions-only contributors demonstrated a significant presence in closeness and betweenness centrality metrics, highlighting their role as dialogue facilitators and connectors within the community. Meanwhile, peripheral developers maintained a stable presence in centrality metrics, indicating that they continue to play an important role in the project's connectivity. On the other hand, issue reporters showed increased influence, particularly in-betweenness, suggesting that they are becoming more integrated into the collaboration network—identifying issues and acting as mediators and connectors. This redistribution of roles suggests that GitHub Discussions may function as a leveling mechanism, enabling contributors with different experience levels and expertise to assume influential positions within the community. This dynamic could strengthen the inclusivity and resilience of projects, aligning with the core principles of open source.

## 5.1 Implications for OSS Communities

Our findings suggest that GitHub Discussions can play a strategic role in improving project sustainability by enabling more distributed forms of leadership and participation. As discussion-only contributors increasingly assume central positions in the GitHub Discussions collaboration network, communities can become less reliant on a small core team to take care of this new channel. This redistribution of influence can reduce burnout among maintainers and foster a more inclusive, participatory environment [30]. OSS communities should consider investing in practices that acknowledge and support non-code contributions as critical elements of project health [37].

## 5.2 Implications for Newcomers

GitHub Discussions offer a low-barrier entry point for newcomers to engage with OSS projects. While only a subset transition into technical contributors, those who do often demonstrate high impact through accepted pull requests and participation in reviews. This highlights the value of fostering social pathways into technical work [26]. Projects seeking to improve newcomer retention could benefit from integrating mentorship and structured onboarding support into their discussion spaces to increase the likelihood of sustained contributions [3].

## 5.3 Implications for Researchers

This study illustrates the importance of analyzing socio-technical dynamics beyond traditional artifacts like commits and pull requests. GitHub Discussions offer a rich, underexplored layer of participation where contributors can influence a project without code contributions [11]. Future research should investigate the interplay between social participation and technical outcomes, explore contributor trajectories over longer periods, and examine how platform features can be designed to better support sustainable and equitable collaboration [28].

## 6 Threats to validity

In this section, we discuss potential threats to validity, categorized into four main areas: internal validity, external validity, construct validity, and conclusion validity [35].

**Internal Validity.** As an observational study, we cannot claim causal relationships between the introduction of GitHub Discussions and changes in contributor behavior. While our two-phase design (before and after GitHub Discussions adoption) provides comparative insights, other factors such as changes in project governance, external events, or shifts in contributor interest may have influenced the trends observed. To mitigate these effects, we employed a longitudinal design across 48 months and analyzed both short-term and long-term trends across multiple projects. However, we acknowledge that some unobserved confounding variables may persist.

**External Validity.** Our sample includes four OSS projects selected for their early adoption and active use of GitHub Discussions. While these projects represent a variation in domain and size, they may not generalize to all OSS contexts or communities using alternative tools for asynchronous collaboration (e.g., mailing lists, Slack, Discord). Moreover, the impact of GitHub Discussions may differ in smaller or less mature projects, which could exhibit different onboarding or interaction dynamics. Future studies with broader samples are needed to verify the generalizability of our findings.

**Construct Validity.** We relied on observable GitHub activity—issues, pull requests, and interactions within GitHub Discussions —to approximate contributor engagement and influence. While grounded in prior literature, these indicators do not fully capture contributors' intent, the quality of contributions, or the content of interactions. Centrality metrics emphasize structural roles but overlook the nuance of individual influence, such as mentoring or moderating behavior that may not manifest through code contributions. Our classification of contributor roles—core, peripheral, issue reporter, and discussion-only—relies on heuristics tied to commit frequency and artifact interactions, which may oversimplify complex participation patterns or exclude contributions not visible on GitHub.

The threshold of three pull requests in a month was chosen to identify sustained contributors, but may exclude long-term, lower-frequency contributors with high-value input. Likewise, the 12-month sliding window smooths variation but may obscure short-term fluctuations in engagement. While these decisions were grounded in empirical precedent, a mixed-methods approach incorporating qualitative data could improve role attribution and better capture social dynamics.

**Conclusion Validity.** We employed linear regression and network analysis to explore associations between GitHub Discussions and observable project activity. Although statistically significant results were found in multiple projects, the explanatory power of some models was modest, and variance explained varied across contexts. We chose not to normalize data across projects to preserve interpretability within each context, which limits cross-project comparison but supports intra-project relevance. Repetition of individuals in centrality metrics supports the robustness of certain trends, but causal conclusions cannot be drawn.

Our method is grounded in established empirical techniques and supported by multiple layers of analysis. Nonetheless, we acknowledge the limitations inherent to studying socio-technical dynamics in real-world OSS environments and encourage further work to validate and extend our findings.

## 7 Conclusion

This study explored the role of GitHub Discussions in shaping open-source project dynamics. Our findings show that while GitHub Discussions did not substantially alter maintainers' workload, they provided an effective entry point for newcomers and enabled more distributed collaboration.

Contributors who began through GitHub Discussions often transitioned into technical roles, with many submitting pull requests, reviewing code, and engaging in feedback. These contributors not only integrated into the project but also sustained their participation, highlighting the onboarding value of GitHub Discussions.

We also observed a shift in network centrality, with *discussion-only contributors* increasingly occupying influential positions. This redistribution of influence suggests that GitHub Discussions support broader participation and reduce dependency on core developers.

In summary, GitHub Discussions contribute to OSS sustainability by enabling informal engagement, fostering inclusive collaboration, and supporting long-term contributor integration. Future work should examine how these dynamics evolve over time and their impact on project resilience and growth. Potential directions include integrating discussion metrics into project health dashboards, exploring the role of discussion contributors in mentoring and community building, and assessing how these engagement patterns influence project governance and decision-making processes.

## ARTIFACT AVAILABILITY

For replication purposes, we made our data and source code used for collection and analysis publicly available [19].

## ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Igor Wiese, thank you for the support of the Fundação Araucária grant *PRD*2023361000043.

## REFERENCES

[1] Thomas Bock, Nils Alznauer, Mitchell Joblin, and Sven Apel. 2023. Automatic Core-Developer Identification on GitHub: A Validation Study. *ACM Trans. Softw. Eng. Methodol.* 32, 6, Article 138 (Sept. 2023), 29 pages. doi:10.1145/3593803

[2] Bernd Bruegge, Allen H. Dutoit, and Timo Wolf. 2006. Sysiphus: Enabling informal collaboration in global software development. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE '06)*. IEEE Computer Society, USA, 139–148.

[3] Jenny Chen and Helena D Cooper-Thomas. 2023. Finding one's own way: how newcomers who differ stay well. In *Evidence-based HRM: a Global Forum for Empirical Scholarship*, Vol. 11. Emerald Publishing Limited, 143–157.

[4] Chia-Shang James Chu. 1995. Time series segmentation: A sliding window approach. *Information Sciences* 85, 1 (1995), 147–173. doi:10.1016/0020-0255(95)00021-G

[5] Kevin Crowston, James Howison, and Hala Annabi. 2006. Information systems success in free and open source software development: Theory and measures. *Software Process: Improvement and Practice* 11, 2 (2006), 123–148.

[6] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. 1277–1286.

[7] Edson Dias, Paulo Meirelles, Fernando Castor, Igor Steinmacher, Igor Wiese, and Gustavo Pinto. 2021. What Makes a Great Maintainer of Open Source Projects?. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, Madrid, ES, 982–994. doi:10.1109/ICSE43902.2021.00093

[8] Yvonne Dittrich and Rosalba Giuffrida. 2011. Exploring the Role of Instant Messaging in a Global Software Development Project. In *2011 IEEE Sixth International Conference on Global Software Engineering*. 103–112. doi:10.1109/ICGSE.2011.21

[9] Nadia Eghbal. 2020. *Working in public: the making and maintenance of open source software.* Stripe Press San Francisco, San Francisco, CA, USA.

[10] Linton C. Freeman. 1978. Centrality in social networks conceptual clarification. *Social Networks* 1, 3 (1978), 215–239. doi:10.1016/0378-8733(78)90021-7

[11] R Stuart Geiger, Dorothy Howard, and Lilly Irani. 2021. The labor of maintaining and scaling free and open-source software projects. *Proceedings of the ACM on human-computer interaction* 5, CSCW1 (2021), 1–28.

[12] Hideaki Hata, Nicole Novielli, Sebastian Baltes, Raula Gaikovina Kula, and Christoph Treude. 2022. GitHub Discussions: An Exploratory Study of Early Adoption. *Empirical Softw. Engg.* 27, 1 (jan 2022), 32 pages. doi:10.1007/s10664-021-10058-6

[13] Dongyang Hu, Tao Wang, Junsheng Chang, Gang Yin, and Yang Zhang. 2018. Multi-Discussing across Issues in GitHub: A Preliminary Study. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. 406–415. doi:10.1109/APSEC.2018.00055

[14] Jack Jamieson, Eureka Foong, and Naomi Yamashita. 2022. Maintaining Values: Navigating Diverse Perspectives in Value-Charged Discussions in Open Source Development. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 449 (Nov. 2022), 28 pages. doi:10.1145/3555550

[15] Yiqiao Jin, Yunsheng Bai, Yanqiao Zhu, Yizhou Sun, and Wei Wang. 2023. Code Recommendation for Open Source Software Developers. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23)*. Association for Computing Machinery, New York, NY, USA, 1324–1333. doi:10.1145/3543507.3583503

[16] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 92–101. doi:10.1145/2597073.2597074

[17] Marcia Lima, Igor Steinmacher, Denae Ford, Evangeline Liu, Grace Vorreuter, Tayana Conte, and Bruno Gadelha. 2023. Looking for related posts on GitHub discussions. *PeerJ Computer Science* 9 (2023), e1567.

[18] Márcia Lima, Igor Steinmacher, Denae Ford, Grace Vorreuter, Ludimila Gonçalves, Tayana Conte, and Bruno Gadelha. 2025. How are discussions linked? A link analysis study on GitHub Discussions. *Journal of Systems and Software* 219 (2025), 112196.

[19] Ana Maciel, Mairieli Wessel, Igor Wiese, and Igor Steinmacher. 2025. Replication package for this paper. https://figshare.com/s/04b0b90e0b3c4efb8d2b

[20] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 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* (San Antonio, Texas, USA). Association for Computing Machinery, New York, NY, USA, 117–128.

[21] 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 and Methodology (TOSEM)* 11, 3 (2002), 309–346.

[22] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. 2021. *Introduction to linear regression analysis.* John Wiley & Sons.

[23] Nachiappan Nagappan, Thomas Ball, and Andreas Zeller. 2006. Mining metrics to predict component failures. In *Proceedings of the 28th International Conference on Software Engineering* (Shanghai, China) *(ICSE '06)*. Association for Computing Machinery, New York, NY, USA, 452–461. doi:10.1145/1134285.1134349

[24] Ifraz Rehman, Dong Wang, Raula Gaikovina Kula, Takashi Ishio, and Kenichi Matsumoto. 2020. Newcomer Candidate: Characterizing Contributions of a Novice Developer to GitHub. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 855–855. doi:10.1109/ICSME46990.2020.00110

[25] George AF Seber and Alan J Lee. 2012. *Linear regression analysis.* John Wiley & Sons.

[26] 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 *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work; Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 1379–1392. doi:10.1145/2675133.2675215

[27] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco Aurélio Gerosa. 2018. Almost There: A Study on Quasi-Contributors in Open-Source Software Projects. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 256–266. doi:10.1145/3180155.3180208

[28] Margaret-Anne Storey, Christoph Treude, Arie van Deursen, and Li-Te Cheng. 2010. The impact of social media on software engineering practices and tools. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (Santa Fe, New Mexico, USA) *(FoSER '10)*. Association for Computing Machinery, New York, NY, USA, 359–364. doi:10.1145/1882362.1882435

[29] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M. German. 2017. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering* 43, 2 (2017), 185–204. doi:10.1109/TSE.2016.2584053

[30] Bianca Trinkenreich, Mariam Guizani, Igor Wiese, Anita Sarma, and Igor Steinmacher. 2020. Hidden Figures: Roles and Pathways of Successful OSS Contributors. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2, Article 180 (2020), 22 pages. doi:10.1145/3415251

[31] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. 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*. 144–154.

[32] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. 2015. Perceptions of Diversity on Git Hub: A User Survey. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. 50–56. doi:10.1109/CHASE.2015.14

[33] Dong Wang, Masanari Kondo, Yasutaka Kamei, Raula Gaikovina Kula, and Naoyasu Ubayashi. 2023. When conversations turn into work: a taxonomy of converted discussions and issues in GitHub. *Empirical Software Engineering* 28, 6 (2023), 138.

[34] Stanley Wasserman and Katherine Faust. 1994. Social network analysis: Methods and applications. (1994).

[35] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering.* Springer Science & Business Media.

[36] Jiang Wu, Xiao Huang, and Bin Wang. 2023. Social-technical network effects in open source software communities: understanding the impacts of dependency networks on project success. *Information Technology & People* 36, 2 (2023), 895–915.

[37] Jean-Gabriel Young, Amanda Casari, Katie McLaughlin, Milo Z Trujillo, Laurent Hébert-Dufresne, and James P Bagrow. 2021. Which contributions count? Analysis of attribution in open source. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 242–253.

[38] 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.