GitHub Discussions: An exploratory study of early adoption

Hideaki HATA

Nicole NOVIELLI

Sebastian BALTES

Raula KULA

Christoph TREUDE
Singapore Management University, ctreude@smu.edu.sg

Citation

GitHub Discussions:
An Exploratory Study of Early Adoption

**Hideaki Hata · Nicole Novielli ·
Sebastian Baltes · Raula Gaikovina
Kula · Christoph Treude**

Abstract Discussions is a new feature of GitHub for asking questions or discussing topics outside of specific Issues or Pull Requests. Before being available to all projects in December 2020, it had been tested on selected open source software projects. To understand how developers use this novel feature, how they perceive it, and how it impacts the development processes, we conducted a mixed-methods study based on early adopters of GitHub discussions from January until July 2020. We found that: (1) errors, unexpected behavior, and code reviews are prevalent discussion categories; (2) there is a positive relationship between project member involvement and discussion frequency; (3) developers consider GitHub Discussions useful but face the problem of topic duplication between Discussions and Issues; (4) Discussions play a crucial role in advancing the development of projects; and (5) positive sentiment in Discussions is more frequent than in Stack Overflow posts. Our findings are a first step towards data-informed guidance for using GitHub Discussions, opening up avenues for future work on this novel communication channel.

Hideaki Hata
Shinshu University
E-mail: hata@shinshu-u.ac.jp

Nicole Novielli
University of Bari
E-mail: nicole.novielli@uniba.it

Sebastian Baltes
QAware GmbH, University of Adelaide
E-mail: research@sbaltes.com

Raula Gaikovina Kula
Nara Institute of Science and Technology
E-mail: raula-k@is.naist.jp

Christoph Treude
University of Melbourne
E-mail: christoph.treude@unimelb.edu.au

Keywords GitHub Discussions · communications · sentiment · empirical study · exploratory study

1 Introduction

As part of the 2020 edition of GitHub's Satellite conference in early May, the platform announced a number of new features, including Codespaces, GitHub Private Instances, and GitHub Discussions. The latter were touted as an alternative to GitHub Issues and Pull Requests, stating that "software communities don't just write code together. They brainstorm feature ideas, help new users get their bearings, and collaborate on best ways to use the software".[1] In their announcement, GitHub argued that the linear format of Issues and Pull Requests—while well suited for merging code—is not suitable for creating a community knowledge base. With the goal of filling this gap, GitHub Discussions was released in May 2020 in a closed beta phase for selected public repositories. In December 2020 during the GitHub Universe conference, the platform announced that GitHub Discussions was released as a public beta feature.[2]

Originally, this feature was requested by software developers back in 2016.[3] As such requests show, there are questions about the effectiveness of the communication channels that developers currently have at their disposal, in particular for open-ended tasks such as brainstorming. Discussions is one of the first attempts at addressing this on the GitHub platform. As previous work reported, innovative services could attract contributors (Hata et al, 2015). GitHub's announcement drew the attention of the software developer community which questioned the role of the new feature in the existing landscape of question-and-answer (Q&A) forums, mailing lists, issue trackers, and others. For example, in a popular Hacker News thread[4] debating the announcement, opinions ranged from "GitHub is slowly eating the surrounding cities [...] now Discussions aims for a piece of Stack Overflow pie" to "This should make Github much more pleasant once it spreads across most projects". In particular the relationship between the new Discussions feature and the Q&A forum Stack Overflow garnered interest, with some users expecting a "reduction" in Stack Overflow and others arguing that Stack Overflow "is literally anti-discussion, the whole point is it is about questions and answers, not discussion [...] That isn't to say discussion isn't useful or valuable, it absolutely is, it is just a completely different thing".

The collaborative and participatory nature of software development is shaped by the communication channels that are used by software developer

---

[1] https://github.blog/2020-05-06-new-from-satellite-2020-github-codespaces-github-discussions-securing-code-in-private-repositories-and-more/#discussions

[2] https://github.blog/2020-12-08-new-from-universe-2020-dark-mode-github-sponsors-for-companies-and-more/#discussions

[3] https://github.com/dear-github/dear-github/issues/44

[4] https://news.ycombinator.com/item?id=22388639

communities of practice (Storey et al, 2016). At the same time, these communities will design their own social protocols of how to use tools and adapt and repurpose them over time (Giuffrida and Dittrich, 2013). Our work follows in a long line of work on how software developers use and appropriate communication channels, ranging from discussions during code review (Tsay et al, 2014) and on Stack Overflow (Beyer et al, 2020) to mailing lists (Guzzi et al, 2013). Here we aim at understanding whether it is worth adding yet another communication channel. Our focus is not on "the feature" but about how it is used for collaborative knowledge-building and how its usage differs from existing channels.

As the software engineering research community is often criticized for being slow in responding to trends, we monitored the closed beta phase to provide data-driven feedback for early adopters who decide to use the feature now that it is publicly available. In this paper, we report the results of the first comprehensive mixed-methods study on how software developers utilize the GitHub Discussions feature during the closed beta period, what they are using it for and who uses it, using qualitative and quantitative analysis, online surveys, and sentiment analysis. We also investigate how these early adopters were initially planning to use it and how they now perceive its usefulness and its place in the complex landscape of existing communication channels and artefacts. We further study the feature's impact on the development process by analysing Discussions' interplay with GitHub Issues and Pull Requests, and we investigate how GitHub Discussions compares to Stack Overflow threads in terms of their sentiment and toxicity.

Based on analysing 7,116 discussions with 29,136 posts from 92 GitHub projects, we find that software developers use the new feature to discuss errors, discrepancies and reviews, and that most discussions were started by project users. Developers initially envisioned using the new feature for question-answering, idea sharing, and community engagement, and appreciate the integration of discussions into the rest of GitHub while highlighting potential issues related to redundancy. Discussions can drive development by triggering the creation of new Issues and Pull Requests. Compared to Stack Overflow threads, positive sentiment is more frequent in GitHub Discussions.

As GitHub Discussions is now available to all projects hosted on the platform, our findings can guide development teams in their decision on whether and how to adopt the feature by establishing how it is being used by early adopters and reporting the experiences of those using it. Our insights further aim to improve the practices developers follow when contributing to GitHub Discussions and guide GitHub and other platforms on how the feature can be improved, in particular in terms of reducing redundancy. Our work also introduces a novel communication channel to the software engineering literature, opening up venues for future work on GitHub Discussions and their interplay with other channels and artefacts in the complex software development landscape.

The remainder of the paper is organized as follows. In the next section, we introduce the GitHub Discussions feature in detail, before we present our
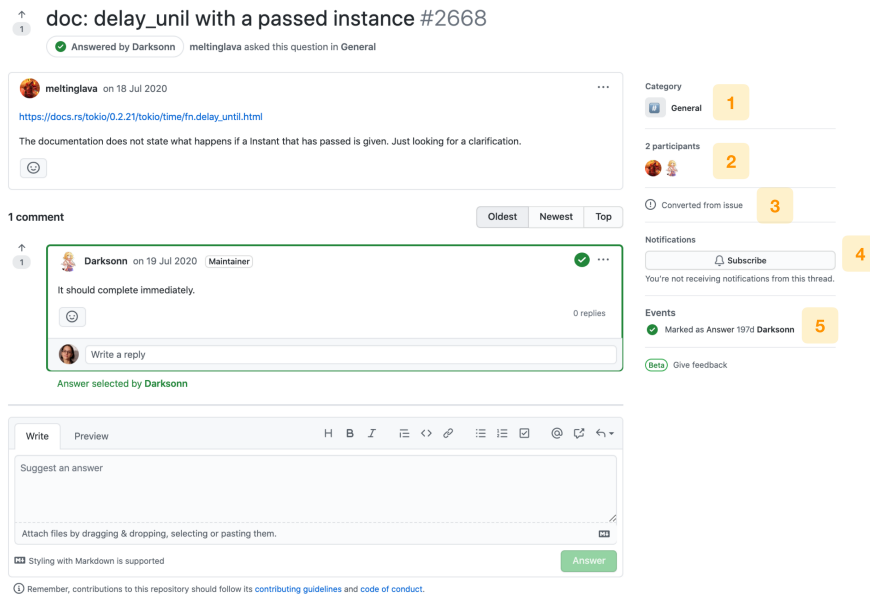
Fig. 1 The GitHub Discussions feature: categories, participants, relations, notifications, and events.

research questions and data collection approach in Sections 3 and 4. In Section 5, we describe the methodology used to address our research questions. Then we present our results in Section 6 and discuss limitations in Section 8. Finally, we position our work in the context of related work (Section 9) and conclude by envisioning possible challenges for future research based on our findings (Section 10).

2 GitHub Discussions

GitHub Discussions is a new feature for GitHub specifically designed to support communication for the members of an open source project community. The forum is designed to host conversations that, unlike issues, are not directly related to code development and, thus, are not supposed to be included in the project boards. Nevertheless, the official GitHub guidelines[5] recommend to clearly define a code of conduct for each project to guide contributors towards effective communication.

The GitHub Discussion interface is structured in a Question-Answering fashion (see Figure 1). However, there are several differences with respect to popular QA-technical forums, such as Stack Overflow, which we highlight in the following. Differently from Stack Overflow, a GitHub Discussion can serve

---

[5] `https://docs.github.com/en/discussions/quickstart`

different communicative intentions, beyond question-answering. For example a Discussion can be started to share information or make announcements, report errors or discuss about potential evolution of a software project. For example, Figure 1 depicts a thread starting from a conceptual question aimed at understanding the code behavior.

Discussion are organized according to categories (highlighted as item number 1 in the figure), such as 'General', 'Q&A', 'Ideas', and so on. Differently from what happens in Stack Overflow tags, where the information seeker can either reuse or create a new tag at the moment of question writing, in GitHub Discussions categories are created by either repository owners and contributors with write access. Participants (2) of discussions can use the most relevant categories among the existing ones in order to group and organize threads.

Discussion can either starts from scratch or from a previously existing issue. The web interface includes this information (3) together with the number of participants in the discussion. Similarly to other communication in GitHub, one can subscribe to receive notifications (4). Similarly to Stack Overflow, it is possible to flag an answer as answering the original question. However, differently from Stack Overflow where the question author is the only one allowed to flag an answer as accepted, in GitHub Discussion this can be done by other people as well, as shown in Figure 1 (5).

3 Research Questions

The main goal of the study is to gain insights into the use of GitHub Discussions. Based on this goal, we constructed three main research questions to guide our study. We present these main questions and sub-questions, along with their motivation.

RQ1 *How have GitHub Discussions been adopted?*
    RQ1.1 *What are GitHub Discussions about?*
    RQ1.2 *Who contributes to GitHub Discussions?*
    RQ1.3 *What is the relationship between project characteristics and GitHub Discussion adoption?*

The motivation of our first main research question (**RQ1**) is to understand the kind of discussions that occur in GitHub Discussions. We use a qualitative method to categorize the discussions by the content and the individuals that contribute to the discussions.

RQ2 *What reasons do developers have to adopt GitHub Discussions?*
    RQ2.1 *Why do projects adopt GitHub Discussions?*
    RQ2.2 *How do developers perceive GitHub Discussions?*

Our second main research question (**RQ2**) requires an analysis of the different perceptions of why GitHub Discussions is being used and how developers feel about the feature.

RQ3 *How do GitHub Discussions relate and compare to other communication channels?*

  RQ3.1 *What is the impact of GitHub Discussions on other channels?*

  RQ3.2 *How does the sentiment differ compared to Stack Overflow posts?*

GitHub Discussions does not exist in isolation, but are part of a network of a myriad of other artefacts. Our third main research question (**RQ3**) explores the impact of GitHub Discussions on other artefacts and compares emotional styles with an existing communication channel. The key motivation is that we would like to understand the potential roles of GitHub Discussions for a software project.

4 Data Collection

In this section we describe our data collection methodology, and present our replication package. When we initiated this research project, the GitHub REST API[6] did not support the new Discussions feature yet. In addition, when we asked the GitHub support for projects that have access to the beta feature, they replied that they could not disclose that information. Thus, to determine which projects already use the feature and to collect metadata and posts for analysis, we had to rely on a custom web scraper we built (available on GitHub[7]). In a first step, we utilized the API to retrieve the most popular projects according to their number of stargazers. Previous work has shown that this approach has a high precision in retrieving engineered software projects (Munaiah et al, 2017). We assumed that more popular projects have a larger user base and are more likely to adopt Discussions early on. Our goal was to retrieve at least 10,000 projects. Due to API limitations, we had to execute multiple queries and then merge the data, yielding the 10,899 most popular GitHub projects as of June 26, 2020. Next, we used our web scraper to determine which GitHub features, including Discussions, Issues, and Pull Requests, are activated in those projects. Many projects used Pull Requests (10,896), Actions (10,744), and Issues (10,432). The Projects feature was less common (8,997), followed by Wikis (4,071) and the new Discussions feature (99). After manually checking the projects using Discussions, we excluded one not being used for software development.

For the remaining 98 projects, we scraped all discussions, including metadata such as title, state (answered or unanswered), author, timestamp, and whether the discussion thread was converted from an existing Issue. For individual posts, we stored author, timestamp, whether a post was part of the selected answer (posts can be nested in discussions), and the HTML content.

We executed several data collection runs. The dataset we used to answer **RQ1** and **RQ2** was collected on July 22 and contains 7,116 discussions with 29,136 posts from 92 projects; the dataset used to answer **RQ3** was collected

---

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

[7] https://github.com/sbaltes/github-retriever/

on August 1 and contains 9,234 discussions with 32,714 posts from 92 projects. The earliest discussion thread in our dataset had started in January 2020. Note that some projects had the Discussions feature activated, but no discussion threads yet. Further, some projects converted existing Issues into Discussion threads: In the August 1 dataset, 1,104 discussions (12%) with 5,885 posts (18%) were converted from Issues, as opposed to 1,007 (14%) discussions with 5,401 posts (19%) in the older one.

4.1 Online Appendix

Our appendix containing results of our qualitative analyses is available on GitHub[8] and Zenodo (Hata et al, 2021). The appendix includes files with links to the full Discussions on GitHub. We further published the scripts we used to collect the data not available via the GitHub API.[9]

5 Methods

This section describes our mixed-methods procedure including a qualitative analysis, a quantitative analysis, a survey, and a semantic analysis.

5.1 Qualitative Analysis

To understand what kind of discussions occurred in GitHub Discussions (**RQ1.1**), investigate the relationship between project characteristics and GitHub Discussion adoption (**RQ1.3**), get an initial view of developer perceptions of the GitHub Discussions feature and to understand why projects adopt GitHub Discussions (**RQ2.1**), and investigate how GitHub Discussions affects the projects that they belong to and in particular understand their impact on other channels and artefacts (**RQ3.1**), we manually annotate target instances, with multiple coders, in multiple iterations.

Discussion Categories (RQ1.1) We conduct a qualitative analysis of statistically representative samples of two groups of discussions, namely, discussions converted from Issues and not from Issues (started originally at GitHub Discussions). We have 1,007 and 6,109 discussions for the two groups, respectively. We randomly sampled a statistically representative number of discussions from each discussion group, ensuring that our findings regarding percentages within each sample would generalize to the entire groups with a confidence level of 95% and a confidence interval of 5. We obtained 278 (from Issues) and 361 Discussions (not from Issues) for our samples, for a total of 639 discussions overall. Our codes for discussion categories were informed by a taxonomy of question

---

[8] https://github.com/sbaltes/GHDiscussions
[9] https://github.com/sbaltes/github-retriever

categories to understand kinds of questions on Stack Overflow (Beyer et al, 2020). Since we found that their codes did not cover all types of discussions on GitHub Discussions, we developed our coding schema by modifying certain codes. The following list shows the 13 categories along with their descriptions. The top seven categories are based on the existing taxonomy (Beyer et al, 2020): some are exactly the same and the others are slightly modified. Four categories emerged from our analysis, in addition to *Other* and *404*. These four categories are unique discussion categories in GitHub Discussions compared to questions on Stack Overflow.

- *Errors*: This is said to be equivalent to the categories Error (Treude et al, 2011) and Exception handling (Beyer and Pinzger, 2016). Discussions of this category contain exceptions and stack traces and ask for help in fixing errors or understanding what the exceptions mean.
- *Discrepancy*: This contains the categories Discrepancy (Treude et al, 2011), Do not work (Allamanis and Sutton, 2013), and What is the problem? (Beyer and Pinzger, 2016). Such discussions contain questions about problems or unexpected behavior where the questioners have no clue how to solve them.
- *Review*: This category merged Decision help and Review (Treude et al, 2011), How/why something works (Allamanis and Sutton, 2013), Better solution (Beyer and Pinzger, 2016), and What (Rosen and Shihab, 2016). Discussions of this category contain code snippets and ask for better solutions, help to make decisions, or to review their code snippets.
- *Conceptual*: This is equivalent to the category Conceptual (Treude et al, 2011) and related to the categories Why and Is it possible? (Beyer and Pinzger, 2016), What (Rosen and Shihab, 2016), and How/why something works (Allamanis and Sutton, 2013). These discussions mainly ask high-level questions, such as the limitations of functionalities, underlying concepts (design patterns, architectural styles, etc.), and background information.
- *Usage*: The taxonomy contains the category API usage (Beyer et al, 2020), which is related to the categories How to implement something and Way of using something (Allamanis and Sutton, 2013) and How-to (Beyer and Pinzger, 2016; Treude et al, 2011). Since we observed questions of usages not limited to APIs, we extended this category to general usages. Discussions of this category contain questions asking for concrete instructions of specific functionalities, which is different from questions in *Discrepancy*.
- *Learning*: This is related to the category Learning a language/technology (Allamanis and Sutton, 2013). In these discussions, the questioners ask for documentation or tutorials to learn the software of the projects. Instead of asking for solutions or instructions on how to do something, they aim at asking for resources of documentation, tutorials, or examples, to learn on their own.
- *Versions*: The taxonomy contains the category API change (Beyer et al, 2020), which is equivalent to the category Version (Beyer and Pinzger, 2016). Discussions in this category are not limited to API changes but

are related to any questions that arise because of different versions of the software or environmental settings.
- *Plans*: Discussions of this category ask for future plans of software releases or development processes.
- *Announcement*: These discussions are used for announcements about specific events of the software (e.g., future updates, releases).
- *Information*: Discussions in this category provide general information, which is not related to events of the software.
- *Recruitment*: These discussions indicate offers of job vacancies or recruiting contributors to the teams.
- *Other*: Discussions that do not fit the above categories.
- *404*: Discussions which had been removed when we accessed them.

Our annotation guidelines do not allow for multiple categories. We annotate the categories based on the initial posts of the threads. The annotation was conducted by three of the authors of this paper. We conducted an initial pilot annotation on two subsets of 30 discussions from the two groups (from Issues and not from Issues). The three coders read and discussed the annotation taxonomy in order to reach a shared understanding of the codes. The three coders labeled individually all items in the two pilot sets of 30 discussions. After this first iteration, we computed the inter-rater agreement to assess the reliability of the annotation approach. Specifically, we computed Fleiss' k index, which extends Cohen's k for evaluating the inter-rater level of agreement between two or more raters, in presence of a categorical variable (Fleiss and Cohen, 1973). It expresses the degree to which the observed proportion of agreement among raters exceeds what would be expected if all raters made their ratings completely randomly. We obtained an observed agreement of 50% and $k = .46$, denoting moderate agreement (Viera and Garrett, 2005). In order to consolidate the annotation guidelines, the three raters discussed and resolved the disagreement cases. We then incorporated the insights derived from the discussion phase into the annotation guidelines and repeated the pilot annotation. After this second round, we observed substantial agreement, with $k = .63$ and observed agreement $= 66\%$. Given the reliability of the annotation guidelines, two of the raters individually completed the annotation for the remaining cases in the two samples.

Project Characteristics (RQ1.3) For software domains, we adopt the following five domains presented in a previous study of GitHub repositories (Borges et al, 2016). While there were six domains in the previous study, since this study focuses on repositories for software development, we excluded "Documentation," which represents repositories with documentation, tutorials, and source code examples. In this analysis, all repositories could be classified into the five domains.

- *Web libraries and frameworks*: Software for web development.
- *Software tools*: Systems that support software development tasks, like IDEs, package managers, and compilers.

- *Application software*: Systems that provide functionalities to end-users, like browsers and text editors.
- *Non-web libraries and frameworks*: Frameworks not intended for web development.
- *System software*: Systems that provide services and infrastructure to other systems, like operating systems, middleware, servers, and databases.

Three authors independently coded 98 projects. We conducted a pilot analysis in order to assess the reliability of our annotation schema. The three raters independently labeled 28 projects from our sample, achieving a substantial inter-rater agreement ($k = .61$, observed agreement = 69%). Thus, the three raters completed the annotation of the software domains for the remaining projects, obtaining again a substantial agreement ($k = .64$, observed agreement = 71%). Overall, we observed a $k = .63$ and observed agreement = 71%. We assign the final domain label to all repositories in our sample based on the majority agreement among the three raters. On four projects, a majority agreement was not reached. Thus, the three raters assigned the domain labels to these cases after a discussion.

Initial Discussions (RQ2.1) We conduct a qualitative analysis of the first discussion thread of each project in our data set since we had anecdotally observed that the first thread was often used for meta-discussions, i.e., discussions about the Discussion feature and social negotiation of how it should be used (Giuffrida and Dittrich, 2013). One author developed a coding schema based on the inspection of 30 randomly selected first threads, and another author annotated the same threads to establish the reliability of the coding schema via inter-rater agreement. We allowed multiple codes per discussion thread. Both annotators achieved perfect agreement in 23/30 cases (77%) and partial agreement in another 4/30 cases (13%), and they disagreed in the remaining 3/30 cases (10%). Based on this satisfactory agreement, one author then annotated the remaining threads. The coding schema we established consisted of the following codes:

- *Question-answering*: Several projects indicated that the Discussion feature could be used for asking and answering questions, e.g., "I'm trying out GitHub Discussions in the hopes that folks will be encouraged to both ask questions and give answers. Remember, there are no stupid questions!"[10]
- *Idea sharing*: Sharing of ideas was also mentioned by several projects, although few projects provided details of how such sharing could be structured.
- *Community engagement*: Projects also mentioned using GitHub Discussions to engage with their broader community, e.g., "We want to try using GitHub's new discussion feature to structure conversations that may not quite be an Issue, but would benefit from community involvement and searchability."[11]

---

[10] `https://github.com/BurntSushi/ripgrep/discussions/1552`
[11] `https://github.com/linkerd/linkerd2/discussions/4347`

- *Decluttering issue tracker*: A few projects explicitly mentioned the relationship between GitHub Discussions and GitHub Issues, suggesting that Discussions could help declutter the issue tracker, e.g., "Unsure whether what you're experiencing is a bug or not? Post it here first! if it is indeed a bug, moving it to Issues once it's confirmed is an easy task."[12]
- *Information resource building*: Projects also mentioned using Discussions as a way of building an information resource, which is in line with GitHub's original intent behind the Discussions feature.
- *Feature requests*: Discussing new features was also mentioned as a use case of GitHub Discussions.
- *Thank-you*: In one case, the first discussion thread was used to say thank-you to everyone involved in the project.
- *Testing the feature*: Several development teams used their first GitHub Discussion to test the new functionality, in particular threaded replies.
- *Not meta*: We used the code "not meta" to indicate cases where the first Discussion thread had not been used to discuss the feature itself.

Impact on Channels (RQ3.1) To study the impact of GitHub Discussions on other channels, we collect all links from GitHub Discussions to Issues and Pull Requests in the same repository. To focus on those artefacts that could actually have been affected by a Discussion, we limit our analysis to Issues and Pull Requests which had been opened or closed after the Discussion started. We found 577 such links in our dataset. To investigate whether these links are indicators of GitHub Discussions affecting other artefacts, we manually analysed a statistically representative and random sample of 231 links to determine the relationship between the Discussion and the Issue or Pull Request. Two authors annotated 20 links separately (Cohen's $k = 0.93$), and based on the near perfect agreement, one author annotated the remaining links. We used the following codes:

- *Newly open*: An Issue or Pull Request was opened as a result of a GitHub Discussion. For example, Pull Request #498[13] was opened as a result of Discussion #489[14] in the alpinejs/alpine repository. The discussion started with a question around potential library use, and after ten discussion messages, a contributor announced the corresponding Pull Request which has since been merged.
- *Deepen discussion*: In some cases, GitHub Discussions was used to deepen the discussions related to an Issue or Pull Request, in many cases exploring the broader implications of these artefacts.
- *Reference*: In other cases, the Discussion thread did not directly affect an Issue or Pull Request, but referenced it for traceability purposes. For example, Discussion #4383[15] in the linkerd/linkerd2 repository provides

---

[12] https://github.com/containrrr/watchtower/discussions/567

[13] https://github.com/alpinejs/alpine/pull/498

[14] https://github.com/alpinejs/alpine/discussions/489

[15] https://github.com/linkerd/linkerd2/discussions/4383

an update on the project roadmap, referencing five Issues or Pull Requests as part of the update.

5.2 Quantitative Analysis (RQ1.2)

To identify the types of participants who contribute to GitHub Discussions (**RQ1.2**), we investigated the roles of all discussion participants in our dataset using the GitHub REST API. Although there are some specific roles in GitHub, such as maintainers (promoted organization members who have a subset of privileges available to organization owners) and collaborators (outside contributors who have Read, Write, or Admin permissions), they are not always available via the API. To conduct a consistent analysis of all the targeted projects, we distinguish all the discussion participants based on GitHub terminology, as follows.

– *Member*: A participant belonging to an organization. After retrieving the list of organization members with the GitHub REST API v3, we investigate whether the participant is a member or not. If a project is owned by a personal developer, we consider the personal developer to be a member.
– *Contributor*: A participant that is listed in the repository contributors of a project. We investigate whether the participant is not a member, but belongs to the repository contributors, which can also be obtained with the GitHub REST API v3.
– *User*: Any participant who does not fit the above types.

5.3 Survey (RQ2.2)

To answer **RQ2.2**, we carry out a survey to get developer feedback on GitHub Discussions. Following recent work (Robillard and Treude, 2020), we structured our analysis of the survey answers based on the thematic analysis framework proposed by Braun and Clarke (2006). We process the results in two ways. For the quantitative results, we show the distribution statistics for each answer. However, for the qualitative responses, we first identify whether or not the response is positive or negative against GitHub Discussions. Then in the results, we present representative quotes to support the themes that arise from the thematic analysis. We ask developers the following three questions:

1. *What is your motivation to use GitHub Discussions for your project?*
2. *Do you find GitHub Discussions useful or redundant?*
3. *How does GitHub Discussions differ from the existing communication channels such as GitHub Issues, Pull Requests, or Stack Overflow?*

As a follow-up to our early adopters survey, we sent a similar survey to the rest of GitHub users. At this point the GitHub Discussion function was open to all hosted projects. We ask developers the following four questions:

1. *What do you think of GitHub Discussions?*

Table 1 Demographics of survey participants

		first survey (n=19)	follow-up survey (n=52)
role	core contributor	10	-
	regular contributor	8	-
	newcomer	1	-
gender	male	16	48
	prefer not to say/non-binary	3	4
	female	-	-
programming years	less than 5	2	-
	5-9	6	10
	10-19	6	14
	20-29	3	13
	30 and above	2	8

2. *Do you currently use an existing communication channel similar to GitHub Discussions in your software projects?*
3. *Do you have plans to use GitHub Discussions? Please, kindly add some reasons to support your decision*
4. *In your opinion, how does GitHub Discussions differ from submitting a GitHub Issue or Pull Request? Also, how does it compare to using external forums such as Stack Overflow?*

Table 1 presents an overview of the demographics of our survey participants. The main survey was distributed to all 98 projects we identified as having GitHub Discussions activated. On August 11 and 12, 2020, we posted our survey directly to the discussion channel in each project, yielding 19 responses. Similar to the first survey, the follow-up survey was distributed to 2,000 randomly selected npm developers that hosted their projects on GitHub. By May 2021, the survey had yielded 52 responses. More than half of respondents are core contributors, such as maintainers or lead developers. Programming experience varies from less than 5 years to 30 and above.

5.4 Sentiment Analysis (RQ3.2)

To answer **RQ3.2**, we analyse and compare the sentiment polarity of posts in GitHub Discussions and in Stack Overflow. Our focus on sentiment is motivated by repeated mentions to Stack Overflow being 'anti-discussion' on Hacker News[16] and related work which has identified unfriendly comments on Stack Overflow (Cleary et al, 2013). Being explicitly 'pro-discussion', we speculate that GitHub Discussions could be a forum with comparatively positive sentiment. To this end, we use the collected 32,714 Discussion posts. In parallel, we collect questions and answers from Stack Overflow threads related to the targeted projects. We manually searched Stack Overflow tags for repository and owner names, resulting in a set of 79 tags (82% of project coverage)

---

[16] https://news.ycombinator.com/item?id=22388639

as some projects did not have specific tags on Stack Overflow. The list of tags is included in our replication package in Zenodo.

Using the Stack Exchange Data Explorer,[17] we collected 99,918 posts (questions with the tags and their answers) created between January and July, 2020. We preprocessed all the posts to remove HTML tags and code snippets using the Beautiful Soup library.[18] To extract the sentiment conveyed by the posts in the GitHub Discussions and in Stack Overflow questions and answers, we use two sentiment analysis tools specifically tuned for the software engineering domain, namely SentiStrength-SE and Senti4SD, as they have been demonstrated to be robust enough in either within- or cross-platform settings (Novielli et al, 2018).

Senti4SD (Calefato et al, 2018a) is trained by leveraging supervised machine learning on a manally annotated gold standard of ~4K questions, answers, and comments from Stack Overflow. It exploits a suite of features including bag of words, sentiment lexicons, and semantic features based on word embedding. For this analysis, we used the Python version of Senti4SD (Calefato et al, 2019). Since our goal is to label data from Stack Overflow and GitHub, we use Senti4SD in two different conditions: as off-the-shelf tool, leveraging the classification model originally trained on Stack Overflow, and by retraining it on the GitHub gold standard dataset provided by Novielli et al. (Novielli et al, 2018), in line with their recommendations to retrain supervised tools with data from the target platform. Senti4SD outputs a polarity label in $\{positive, negative, neutral\}$, consistently with the gold labels in the training sets used for training.

SentiStrength-SE (Islam and Zibran, 2017) implements a set of heuristics leveraging sentiment lexicons, that is, large collections of words annotated with their prior polarity, i.e., the positive or negative orientation of the word. The overall sentiment of a text is computed based on the prior polarity of the words composing it, under the assumption that words with negative prior polarity convey negative sentiment, and vice versa. SentiStrength-SE is built upon the API of the general-purpose tool SentiStrength (Thelwall et al, 2010). It leverages a manually adjusted version of the SentiStrength lexicon and implements *ad hoc* heuristics to correct the misclassifications due to domain-specific semantics of terms in the lexicon. SentiStrength-SE assigns an integer value between 1 and 5 for the positivity of a text, p and, analogously, between -1 and -5 for the negativity n. We map these scores to polarity values in $\{positive, negative, neutral, mixed\}$. Specifically, a text is considered positive when $p + n > 0$, negative when $p + n < 0$, and neutral if $p = 1$ and $n = -1$. Texts with a score of $p = -n$ are considered mixed.

We assign the final polarity label for each text in our datasets based on majority agreement between the output of SentiStrength-SE and the two classification models of Senti4SD. Specifically, positive, neutral, and negative labels are assigned based on majority voting, that is when at least two tools

---

[17] https://data.stackexchange.com/stackoverflow/query/new
[18] https://pypi.org/project/beautifulsoup4/

Table 2 Majority voting criterion for polarity label assignment using the polarity labels from SentiStrength-SE and the two versions of Senti4SD.

Input text	SentiStrength-SE			Senti4SD		Final label
	Positive score	Negative Score	Polarity label	Trained on SO	Trained on GH	
"Which version of AdonisJS?"	1	-1	neutral	neutral	neutral	neutral
"Closing discussion is low quality"	1	-1	neutral	negative	negative	negative
"Glad, you found it :)"	2	-1	positive	positive	positive	positive
"You can also force the <CODE> in your configuration <CODE>. Thanks! You shouldn't pass an anonymous class. Yes, this is a very bad example! Only for demo, no one use it :D"	2	-3	mixed	negative	negative	mixed

Table 3 Frequency of discussion categories in our samples

category	from issues		not from issues	
Errors	102	(37%)	77	(21%)
Discrepancy	25	(9%)	134	(37%)
Review	40	(14%)	80	(22%)
Conceptual	45	(16%)	25	(7%)
Usage	27	(10%)	12	(3%)
Learning	12	(4%)	11	(3%)
Versions	11	(4%)	4	(1%)
Plans	8	(3%)	5	(1%)
Announcement	2	(1%)	0	(0%)
Information	1	(0%)	2	(1%)
Recruitment	0	(0%)	2	(1%)
Other	5	(2%)	5	(1%)
404	0	(0%)	4	(1%)
sum	**278 (100%)**		**361**	**(100%)**

agree on a given polarity class (see the first three examples in Table 2). Mixed cases are those classified as such by Sentistrength-SE, which issues to separate predictions for positive and negative polarity (see last example in the table).

6 Results

In the following, we answer our research questions with the data collected according to the research methodology described above. Each sub-question is presented in a dedicated section, ending with a summary of the corresponding results.

6.1 Discussion Categories (RQ1.1)

To answer our first research question (What are GitHub Discussions about?), we manually annotated a representative sample of discussion threads.

Table 3 shows the result of our annotation. A prevalence of Errors (37%) is observed in the discussions originating from previous Issues ('from issues' in the table), which denotes the goal of asking for support in fixing errors or understanding exceptions. As these discussions had been converted from Issues, it is revealed that developers prefer to manage errors on GitHub Discussions

until the underlying bugs have been identified. Errors are reported only in 21% of posts originally started as discussion ('not from issues' in the table). Conceptual and Usage questions are also more frequent in the 'from issues' group. Conversely, we observe a prevalence of Discrepancy (37%) in the 'not from issues' sample, compared to 9% observed for the 'from issues' group. This might be an indication of the more open nature of the Discussions, with respect to Issues or other Q&A platforms, such as Stack Overflow. In fact, Discrepancy refers to the kind of questions that are not well-suited for Issues due to their high-level of abstraction and due to the fact that the authors of such posts have no clue how to solve the problems they describe. Indeed, this is the kind of questions that would not be appreciated on Stack Overflow, whose guidelines explicitly recommend information seekers to provide evidence of previous attempts to solve the problem.[19] For example, Conceptual is considered to be a category related to such evidence, and is less common in 'not from issue' (7%) than in 'from issue' (16%). Conversely, Review is more frequent in 'not from issue' (22%) than in previous Issues (14%).

We compare the distribution of categories for GitHub Discussions with the distribution reported by Beyer et al (2020) for Stack Overflow. In their study, they performed automated categorization of Stack Overflow questions with the above-mentioned taxonomy and obtained the following distribution of question categories: Errors (19%), Discrepancy (26%), Review (7%), Conceptual (22%), API Usage (36%), Learning (2%), and API change (2%). Compared to such distribution, there are less Conceptual and Usage, and more Discrepancy and Review discussions in GitHub Discussions 'not from issues'. We conjecture that this is because there is no strict requirement for providing evidence (leading to more Discrepancy) and because of the increased connectivity to source code (leading to more Review) for GitHub Discussions. Although the percentages are small, there are some discussion categories that do not appear in Stack Overflow, such as Plans, Announcement, Information, and Recruitment in GitHub Discussions, which indicate that GitHub Discussions is more open channels for communication and a community knowledge base.

Summary: Errors, Discrepancy, and Review are prevalent discussion categories in GitHub Discussions, whereas various other categories, such as Announcement, Information, and Recruitment also exist.

6.2 Discussion Participants (RQ1.2)

To investigate our second research question (Who contributes to GitHub Discussions?), we distinguish between the roles of Member, Contributor, and User.

---

[19] 'Writing the perfect question' by Jon Skeet at `https://codeblog.jonskeet.uk/2010/08/29/writing-the-perfect-question/` linked in the official Stack Overflow guidelines at `https://stackoverflow.com/help/how-to-ask`

Table 4 Discussion patterns by questioners

	user		contributor		member	
has selected answer	2,550	(39%)	183	(44%)	33	(23%)
no selected answer	2,464	(39%)	165	(40%)	79	(55%)
no response	1,542	(24%)	68	(16%)	32	(22%)
sum	**6,556(100%)**		**416 (100%)**		**144(100%)**	

Table 5 Respondents of selected answers by questioners

	user		contributor		member	
answered by member	1,162	(46%)	106	(58%)	19	(58%)
answered by contributor	578	(23%)	72	(39%)	9	(27%)
answered by user	810	(32%)	5	(1%)	5	(15%)
sum	**2,550(100%)**		**183 (100%)**		**33(100%)**	

Table 6 Respondents for discussions without answers by questioners

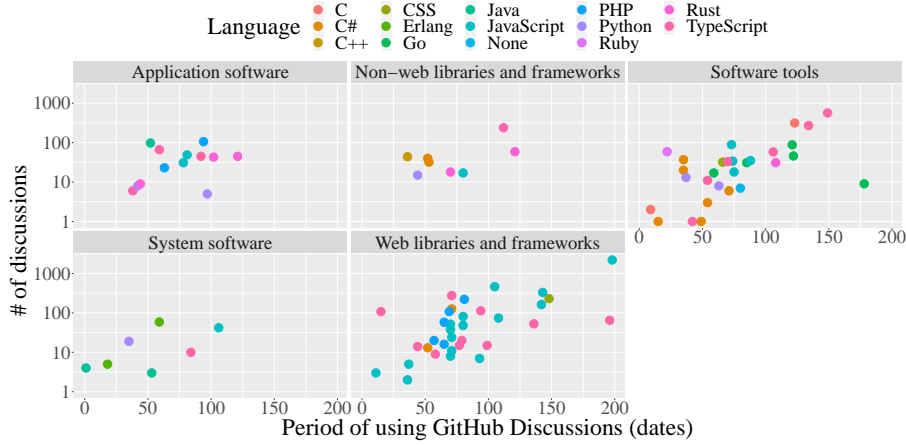|	user		contributor		member	
member responded	1,029	(42%)	82	(50%)	69	(87%)
contributor responded	554	(22%)	72	(44%)	4	(5%)
only user responded	881	(36%)	11	(7%)	6	(8%)
sum	**2,464(100%)**		**165 (100%)**		**79(100%)**	

Table 4 shows the discussion patterns divided by questioners who started discussions. From all 7,116 discussions, the majority have been started by Users, followed by Contributors and Members. For discussions started by all three types, more than three quarters have responses. Similar to Stack Overflow, GitHub Discussions allows users to select answers in discussions. However, as seen in Section 6.1, not all discussion categories require to select single answers, which could be seen in the large percentages of discussions without selected answers (39-55%).

For discussions that have selected answers, Table 5 presents the distributions of answerers by questioners. For all questioners, answers by Members are most frequently selected. Table 6 shows the participation patterns in discussions by questioners, for discussions without selected answers. We divide discussions into three types, that is, discussions in which at least one Member responded, discussions for which no Member was involved but Contributors responded, or discussions in which only Users responded. Similar to Table 5, Member-responded discussions are most frequent. These results indicate that Members actively participated in GitHub Discussions.

In addition, we investigated the response time of each Member, Contributor, and User. The response time was measured from the time a question was posted until a different person than the one who initially asked the question commented on it. The median response time was 4.4 hours for Members, 4.1 hours for Contributors, and 8.5 hours for Users. When the Mann-Whitney U

Table 7 Domains of Target Projects

domain	# projects	%
Web libraries and frameworks	37	38%
Software tools	32	33%
Application software	13	13%
Non-web libraries and frameworks	8	8%
System software	8	8%
sum	**98**	**100%**

Fig. 2 The number of discussions (log scale) by the period of using GitHub Discussions.

test was used to compare the differences in response times, there was no statistically significant difference between Members and Contributors, but there were statistically significant differences between Members and Users, and between Contributors and Users. We can see that comments from Members and Contributors were obtained earlier than comments from other Users.

Summary: Most of the discussions are initiated by Users, and Members were actively responding to them on GitHub Discussions. Also, Members and Contributors responded earlier than other Users.

6.3 Project Characteristics (RQ1.3)

To answer our next research question (What is the relationship between project characteristics and GitHub Discussion adoption?), we employed a qualitative analysis to reveal software domains of the targeted projects and analyzed the relationship of core developer involvement and discussion activities.

The result of the annotation of the software domain is reported in Table 7 and indicates a prevalence of *Web libraries and framework*, immediately
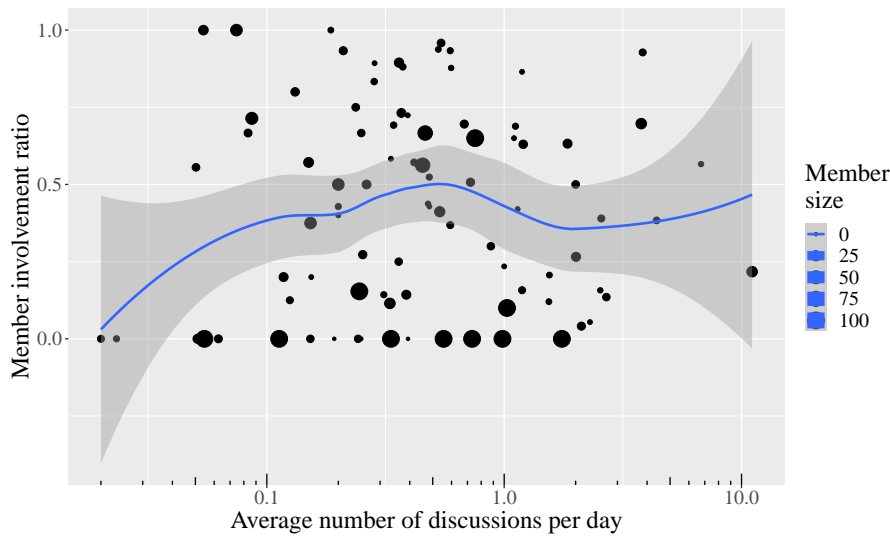
Fig. 3 Membership involvement ratio by average number of discussions per day (log scale). Point size represents the number of Members in a project.

followed by *Software tools* and *Application software*. A previous study that investigated the top 2,500 repositories by number of stargazers on GitHub reported that there were many *Non-web libraries and frameworks*, followed by *Web libraries and frameworks*. In comparison, there were not many Non-web libraries and frameworks in early adopters of GitHub Discussions. Figure 2 presents the distributions of the targeted projects based on the periods using GitHub Discussions and the total number of discussions. The periods were derived by identifying the initial discussions that were not converted from existing Issues. Seven projects are not shown since they had no discussions or had only discussions converted from issues. Programming languages of the projects retrieved using the GitHub REST API are shown with different colors, revealing that the studied projects use various different programming languages. As expected, there is a correlation between the number of discussions and the period of using GitHub Discussions. However, some projects have relatively few discussions considering their discussion periods.

To understand the characteristic of discussion participation and discussion activities, Figure 3 illustrates the relations of the average number of discussions per day and the ratio of Member involvement, which is the ratio of discussions that contain at least one post added by a Member to the total number of discussions. This is measured based on the posts that were included in the discussion at the time of collection. A regression line is plotted using LOESS smoothing, which fits multiple regressions in a local neighborhood (Cleveland and Loader, 1996). The shaded area indicates the confidence intervals. We observe that the Member involvement ratio is increasing until 0.5 in the average number of discussions per day, which could be considered as less active, and

Table 8 Reasons for using GitHub Discussions mentioned in initial discussions. Multiple reasons per project possible.

reason	# projects
question-answering	16
idea sharing	12
community engagement	12
decluttering issue tracker	3
information resource building	2
feature requests	2
thank-you	1
testing the feature	16
not meta	51

decreases and disperses for more than 0.5 average discussions per day. This suggests that high Member involvement encouraged active use of GitHub Discussions, and, for projects with many discussions per day, Contributors and Users also contributed to answering questions.

Summary: The early adopters of GitHub Discussions include projects developed in a variety of programming languages, especially in the domains of "Web libraries and frameworks" and "Software tools". Among the projects using GitHub Discussions, there is a positive relationship between Member involvement and discussion frequency for projects with less active discussions.

6.4 Initial Discussions (RQ2.1)

To investigate why projects adopt GitHub Discussions, we qualitatively analyzed the first discussion thread of each project in our data. This methodology was motivated by our anecdotal observation that the first discussion thread was often used to discuss the Discussions feature and how to best use it, rather than project-specific issues.

Table 8 shows the distribution of the codes across the first discussion threads we annotated. Note that one thread might mention multiple reasons for using GitHub Discussions. About half of the first discussion threads were not used for meta discussions ('not meta' in Table 8), but were instead used to discuss project-specific issues. For example, the first discussion thread of the PrefectHQ/prefect project[20] asks for help with a particular error message. The remaining threads mentioned various reasons for using the GitHub Discussions feature, in particular question-answering, idea sharing, and community engagement. To further understand developers' rationale, we conducted a survey, see next section.

---

[20] https://github.com/PrefectHQ/prefect/discussions/2860

Summary: For projects which used their first discussion thread to discuss how to use the new feature, question-answering, idea sharing, and community engagement were among the top reasons mentioned.

6.5 Developer Survey (RQ2.2)

To understand how software developers perceive the GitHub Discussions feature, we conducted a survey that 1) targeted the early adopters of GitHub discussions and 2) targeted the rest of GitHub projects.

What is your motivation to use the discussion channel for your project? (19 responses): Participants provided a range of answers that revolved around being a community medium to act as a general forum, and not necessarily about Issues. One participant responded:

It's the perfect place to discuss feature request and other communication forms (like questions) that don't really fit in Issues.

Four participants mentioned that they liked that discussions were closer to the code (within the GitHub Platform), as generic questions:

We wanted to offload the constant stream of questions about the project to someplace other than the issue tracker, but still linked to GitHub.

Do you find GitHub Discussions useful or redundant? (19 responses): 15 out of the 19 participants were very positive about the usefulness of GitHub Discussions. One participant responded that it was useful to store drawn-out discussions for the community:

Useful: allows us to have long, drawn-out discussions that don't necessarily have to have a conclusion.

On the other hand, there were three participants that were not as positive towards GitHub Discussions. One participant found the discussions difficult to differentiate from Issues and Pull Requests.

Redundant: people often don't know whether to use a discussion or an issue or a PR. Many discussions end up being opened that are duplicates of issues (and vice versa). In addition, speaking entirely from impressions, it seems that discussion posters do not seem to hold themselves to the same standards as issue posters.

How does GitHub Discussions differ from the existing communication channels such as submitting a GitHub Issue or Pull Request? Also, how does it compare to other external forums such as Stack Overflow? (16 responses): One participant noted that:

Way more appropriate for questions and other topics that are not actionable for the engineering team in any way. GitHub issues and PRs are used by our engineering team as the foundation for their process.

Table 9 Response if respondents use a channel similar to GitHub Discussions

Using Similar Channel	# Responses	
Yes	19	▮
No	26	▮
Other	7	▏
Total	52	

Table 10 Response if respondents consider to use GitHub Discussions

Plans to use GitHub Disussions	# Responses	
Yes	23	▮
No	21	▮
Other	8	▏
Total	52	

Another participant pointed out the accessibility compared to Stack Overflow:

discussions offer a place where we can discuss ideas that aren't necessarily actionable points. Discussions is very similar to StackOverflow in it's approach, but it's way more accessible as it's scoped to the project in question and doesn't have the barrier of entry (due to the somewhat hostile community) that StackOverflow does.

Overall, when comparing to Stack Overflow, there was no uniform opinion as we received both positive and negative responses. One participant noted that the usefulness of a discussion medium also depends on the nature of the project:

For our project, the discussion is a much better forum than StackOverflow (SO) because we are not a very well-known project and have some strange quirks in our configuration rules. For more widespread projects, I would likely prefer SO simply because it has more reach.

For the rest of GitHub projects that have not adopted GitHub Discussions, they provided the following responses to our survey:

What do you think of GitHub Discussions? (52 responses): All responses were positive, stating that GitHub Discussions was different from a pull request or issue. For instance:

I like the nested threads. Regular github issues don't have nested threads so the conversations get a bit shuffled.

Do you currently use an existing communication channel similar to GitHub Discussions in your software projects? (52 responses): Table 9 show the results of the follow-up survey sent to the rest of GitHub projects. The results were mixed, with 26 out of the 52 respondents claiming that they did not use such a similar channel for their communication. On

the other hand, 19 respondents claimed they had an existing communication channel like GitHub Discussions.

Do you have plans to use GitHub Discussions? (52 responses): Similar the previous question, Table 10 shows a mixed response to whether or not the participants had plans to adopt GitHub Discussions into their projects. Still a bit over 50% (23 participants) were willing to use GitHub Discussions.

In your opinion, how does GitHub Discussions differ from submitting a GitHub Issue or Pull Request? Also, how does it compare to using external forums such as Stack Overflow? (46 responses): Different to the early adopters responses, we find that there were more respondents that were positive in regards to the difference between GitHub Discussions and a GitHub Issue or a Pull Request. Applying a thematic analysis approach, we were able to extract the themes of how the respondents expressed themselves. Overall, we found 31 responses to be more positive when explaining the differences of GitHub Discussions. Most discussed the structure and how there was a need for a discussions thread that fell outside of the development pipelines. For instance,

> *Most questions and discussion should not be posted as Issues. However, an open source project can generate a lot of valid discussion. This is where GitHub Discussions will shine and for the reasons mentioned above: open-source communities are oftentimes overwhelmed by requests that are not issues.*

On the other hand, there were also responses that were not as positive. We found six respondents who raised concerns. For instance,

> *GH Issues (and pulls) are still the preferred way to tell the owner/maintainer that there is a bug/feature request/question. They are enabled on all repos by default whereas Discussions need to be enabled. StackOverflow has a wider community would would better be able to answer any question you throw at them, whereas GH Discussions are better suited to questions which are very specific to the project you are asking them in.*

Finally, we found 9 responses that were either unable to tell the difference, or they were neutral with their stance for GitHub Discussions. For example,

> *Compared to Stack Overflow: sounds very similar - we will see which system will win :)*

Summary: Developers are able to distinguish between question-answering and engineering tasks and react positively to the accessibility and integration of GitHub Discussion with other GitHub features, but they also point to the problem of topic duplication between Discussions and Issues. For developers that had not adopted GitHub Discussions, although there were still mixed reactions to its adoption, overall the feedback was positive, often with a wait-and-see approach.

Table 11 Why do links to Issues or Pull Requests exist? Result on a sample of 231 links.

link	reason of link appearance	
issue	newly open	42
	deepen discussion	11
	reference	90
pull requests	newly open	37
	deepen discussion	5
	reference	46
sum		**231**

Table 12 Sentiment analysis of all posts and accepted answers in the GitHub Discussions and Stack Overflow Q&A threads.

polarity	GH Discussions posts		Stack Overflow posts	
	all	*accepted*	*all*	*accepted*
positive	12%	13%	3%	4%
negative	6%	4%	6%	3%
neutral	73%	78%	82%	91%
mixed	9%	6%	9%	3%
Overall items	32,714	4,317	99,918	19,227

6.6 Impact on Other Channels (RQ3.1)

To investigate the impact of GitHub Discussions on other channels, we conducted a qualitative analysis of a representative sample of links to Issues and Pull Requests from Discussions. We only considered links to artefacts that had been opened or closed after the corresponding Discussion started.

Table 11 shows the distribution of reasons for links to Issues and Pull Requests from GitHub Discussions. Out of 231 linked Issues and Pull Requests in our sample, 79 (34%) were created as a result of a GitHub Discussion. In many other cases, links were provided for reference.

> **Summary**: GitHub Discussions does not exist in isolation, but can play a crucial role in moving development forward by triggering new Pull Requests or Issues.

6.7 Sentiment Characteristics (RQ3.2)

To answer our final research question (What are sentiment differences compared to Stack Overflow posts?), we analysed and compared the sentiment polarity of posts in GitHub Discussions and Stack Overflow. The voting label was assigned either with full agreement (the three tools agree) in 64% (GitHub) and 61% (Stack Overflow) of cases or by majority voting in 27% and 30% of cases (2 over 3 agree), indicating a substantial observed agreement

between the tools. Similarly to Stack Overflow, in GitHub Discussions, an answer can also be flagged as accepted. As such, we complement our analysis by also investigating whether there is any difference in the accepted answers between GitHub Discussions and Stack Overflow. The result of the sentiment analysis is reported in Table 12.

In line with previous work (Calefato et al, 2018b; Murgia et al, 2014) we observe that the majority of posts is neutral, i.e. they do not convey any emotions or opinions, which is reasonable, given the technical nature of Discussions. The amount of both mixed and negative posts in the two samples is comparable. As for positive sentiment, we observe that it is present in 12% of GitHub Discussions while it appears in 3% only of Stack Overflow posts, for which we observe also a higher percentage of neutral cases. This might be due to the Stack Overflow community adhering to a more neutral communication style, in line with official recommendations. A recent empirical study demonstrated how expressing either positive or negative emotions is associated with a lower probability of obtaining a useful answer (Calefato et al, 2018b), which further supports our conjecture. In both datasets, accepted answers appear more neutral, as a consequence of the less negative and mixed sentiment. We also observe a small increment in positive sentiment. It is the case, for example, of sentences expressing either wish (e.g., 'See the code below! Happy 2020!'), politeness expressions (e.g., 'I hope this solves your issue') or positive sentiment in formulating requests (e.g., 'It would be nice if we can convert that issue into a discussion ☺'). To conclude, the differences in the sentiment observed for the two samples are mainly due to GitHub Discussions being more positive than SO posts. The results of chi-squared test applied to the contingency table show that the differences are statistically significant ($p < .05$) but the effect size is negligible (Cramer's V = .17) (Cohen, 1988).

Previous results of qualitative analysis of emotions in Stack Overflow suggest that a polarity-based operationalization of affect might not be adequate to capture the nuances in the attitude conveyed by negative sentiment (Calefato et al, 2018b). In fact, the presence of negative sentiment does not necessarily indicate the use of abusive language or toxic interactions (Gachechiladze et al, 2017). Being able to identify rude, aggressive comments among negative ones is crucial, as toxicity might impair effective collaboration and prevent people from contributing to software projects (Steinmacher et al, 2014).

In line with this view, we performed a follow-up analysis to investigate the toxicity of 1,962 posts from GitHub Discussions and 5,635 questions and answers from Stack Overflow that were classified as negative. Consistently with previous research (Raman et al, 2020), we computed the toxicity scores for both samples of negative posts, using the Google Perspective API,[21] specifically designed to identify abusive language and harassment in online conversations. The Perspective API provides, for a given text in input, a toxicity score in $[0, 1]$ representing the probability that a reader would perceive the text as rude or disrespectful, leading him/her to abandon the conversation. Results
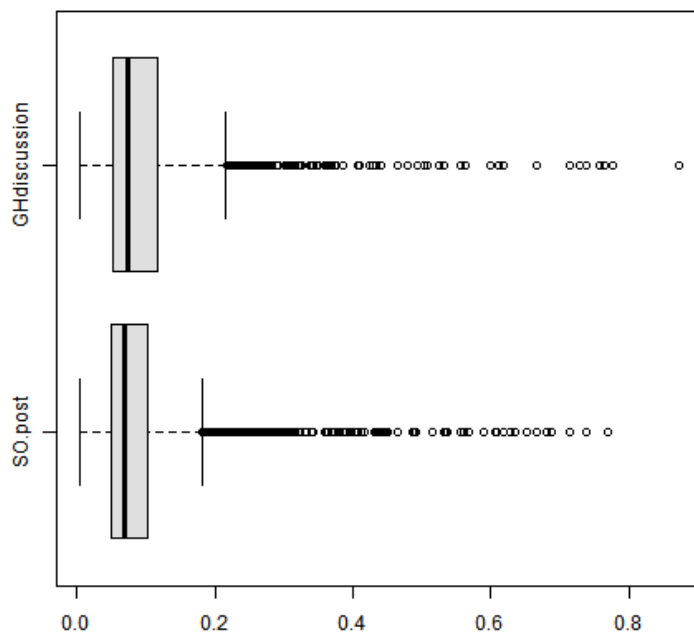
---

[21] www.perspectiveapi.com

Fig. 4 Toxicity in negative posts on GitHub Discussions and Stack Overflow.

are reported in Figure 4 and show that toxic posts are rare in line with previous evidence provided by Raman et al (2020), who analyzed the toxicity in open source discussions.

Summary: The majority of Discussion posts is neutral. Toxicity and abusive language are rarely detected. The sentiment is comparable to the sentiment of Stack Overflow posts except for positive sentiment, which is more frequent in GitHub Discussions.

7 Discussion

Our findings can be summarized into the following recommendations for practitioners.

- *Set guidelines for participating in discussions.* As found in the response from developers in Section 6.5, users sometimes find it difficult to choose an appropriate channel from Discussions, Issues, and Pull Requests. It should be beneficial for community members to set a policy of using these channels depending on the project-specific development processes.
- *Encourage core developers to participate in discussions.* It is evident that the involvement of core developers is crucial in the successful use of GitHub Discussions as seen in Section 6.3.

– *Prepare for newcomers.* GitHub Discussions can be a gateway for newcomers to participate by creating discussions specifically for them (Steinmacher et al, 2019). Not only does a project present tasks that are easy for newcomers to start, but it can also provide a place for mentoring and effective communication between mentors and newcomers, potentially solving the mentoring challenges pointed out in a recent study (Balali et al, 2020).

Besides, our analysis has shown that Stack Overflow discussions are more neutral compared to GitHub Discussions and also less positive (see Section 6.7). Software projects might use this as an opportunity to establish GitHub Discussions as a feedback channel that explicitly encourages users to utter their appreciation for a certain feature or a response they received in a discussion. This would also help to distinguish GitHub Discussions from discussions on Stack Overflow, where, e.g., thank you messages are discouraged.[22]

From our exploratory study of early adoption of GitHub Discussions, we see the following avenues for future research:

– *Tool support for effective discussions.* As developers mentioned (Section 6.5), duplicate posts in Discussions and Issues could be troublesome. Automated support using bots like (Abdellatif et al, 2020) is promising.
– *Further studies of error/issue handling.* As we saw in Section 6.1, more than a third of the issues converted to discussions were "errors". This can be attributed to the fact that some projects prefer to have discussions on how to deal with errors, unexpected behavior, etc., and to determine if errors are bugs, in discussions rather than in issues. As seen in Section 6.6, there are issues and pull requests created after discussing potential issues and features, which indicates that the workflow of discussion before working on issues or pull requests is used in some projects. Such a new workflow may result in new bug lifecycles that are different from those previously considered (Aranda and Venolia, 2009). However, we have also observed that some projects had stopped using GitHub Discussions. For example, the dotnet/aspnetcore project decided to use issues for discussions and disabled GitHub Discussions because they faced difficulties in labeling and assignment using it.[23] What features are suitable for discussions may vary depending on the characteristics of projects, such as development histories, types of end users, maturity of the communities, etc. Since GitHub Discussions is still in beta, there is still room for improvement in terms of functionality. A better understanding of the required features and error/issue handling process depending on the characteristics of projects will be a challenge in the future.
– *Further studies of human aspects.* As recent studies have shown, communication is an essential attribute of a great maintainer (Dias et al, 2021). A quantitative analysis of the relationship between the type and experience

---

[22] https://codeblog.jonskeet.uk/2010/08/29/writing-the-perfect-question/
[23] https://github.com/dotnet/aspnetcore/issues/29935

of developers and the discussion categories that require different skills such
as technical, social, and managerial will be a future research challenge.

– *Relationship between GitHub Discussions and other channels.* In addition
to comparing to Stack Overflow, future work can also consider compar-
ing GitHub Discussions to other communication channels used by open
source projects, such as Gitter (Chatterjee et al, 2021; Sahar et al, 2021),
for example focusing on links between Gitter conversations and GitHub
Discussions or comparing the channels in terms of their sentiment.

8 Threats to Validity

Threats to internal validity concern the possibility of introducing a threat
due to a possible causal relationship between the treatment and the result of
an analysis. As for sentiment and toxicity in GitHub Discussions and Stack
Overflow posts, we performed the analysis at the platform level. However, we
might have observed different distribution of sentiment and toxicity at the
level of individual repositories, depending on the degree of awareness of the
code of conduct of community members of a software project. Furthermore,
we analysed and compared the sentiment polarity distribution in two dataset
including 32,714 posts from GitHub Discussions and 99,918 posts from Stack
Overflow, respectively. Specifically, we collected posts from Stack Overflow
based on the presence of tags and owner names from GitHub projects in our
dataset. The list of keywords (tags and name) was obtained via manual search
on Stack Overflow. We acknowledge that manual search might introduce a
threat to validity as well as replicability issues. To mitigate this threat, we
include the list of keywords in our replication package. Furthermore, some
projects did not have specific tags on Stack Overflow. Further replications
could be conducted in the future on a larger set of data in order to include more
projects, thus strengthening the representativeness of the sentiment analysis
study.

Threats to construct validity concern the degree of accuracy to which the
variables defined in a study measure the constructs of interest. As for analysis
of sentiment, we operationalized developers' emotions by measuring the senti-
ment polarity of a text, that is its positive or negative orientation. However,
emotion lexicon can convey a wide range of affective states or attitudes to-
wards the interlocutor beyond polarity. As for participant types in Section 6.4,
although we identified three types, more fine-grained type identification in-
cluding maintainers and collaborators could reveal project-specific patterns of
discussion activities and discussion participation.

Threats to external validity concern the ability to generalize the findings
in this study. As we study the usage of a beta feature, our data is derived from
a short term (seven months). Long-term studies may yield different conclu-
sions. In addition, since we studied the use of GitHub Discussions in popular
(based on the number of stargazers) and selected (by GitHub for beta testing)
projects, our results could be biased to those specific communities. To address

these issues, further research is needed based on various projects participating in GitHub Discussions after the feature is officially released.

9 Related Work

9.1 Communication during Software Development

Discussions play a vital role in reaching a decision when doing code review. Tsay et al (2014) studied how developers used discussions to evaluate Pull Requests. Hirao et al (2020) studied the divergent nature during the code review, while Ebert et al (2019) present a taxonomy of confusion that a reviewer may encounter when conducting a review. Pascarella et al (2018) present a taxonomy of information needs that help with conducting a review. Sulistyo Nugroho et al (2020) studied the characteristics of the usage of a project-specific forum in the Eclipse project. Other work has explored different aspects related to the analytics of review comments (Hirao et al, 2019; Rahman et al, 2017). One aspect that could benefit is how review comments could be used for reviewer recommendation (Jiang et al, 2017), where discussion recommendations can recommend appropriate team members to help answer generic questions from the community. This can be beneficial for the community, especially for newcomers or even to help deepen the understanding between both users and developers of the software.

Besides code review, discussions also take place on online platforms such as Stack Overflow. Previous work studied aspects such as who participates in such discussions (Vasilescu et al, 2012; Morrison and Murphy-Hill, 2015), how frequently users contribute (Wang et al, 2013), how frequently discussed code snippets are copied between platforms (Yang et al, 2017; Baltes and Diehl, 2019), and how content evolves on Q&A websites (Baltes et al, 2018; Wang et al, 2018). From the study of discussions in issue tracking systems, Arya et al. reported various discussion categories in issue discussions, such as observed bug behaviour, bug reproduction, solution discussion, etc. (Arya et al, 2019). We observed some categories overlap with categories found in GitHub Discussions, such as task progress, future plan, and social conversation. Since these discussions do not need to be closed with particular resolutions, GitHub Discussions could be an appropriate channel as mentioned by developers in Section 6.5.

From the point of a community knowledge base, source code comments have been found to document information specific to the associated code, such as technical debt (Potdar and Shihab, 2014), todo tasks (Storey et al, 2008), and algorithm (Inokuchi et al, 2019). Similar to the analysis in Section 6.6, referencing and connecting to external sources in source code comments had been analyzed (Hata et al, 2019). A recent study identified specific cases of issues and technical debt, called on-hold self-admitted technical debt, which needs appropriate management of code considering the updates of issues (Maipradit et al, 2020b,a). Although different knowledge is communicated and discussed in

different communication channels, a new feature of GitHub Discussions could be a center of a community knowledge base connecting with other artefacts.

9.2 Sentiment Polarity in Communication Channels

Our sentiment analysis in Section 6.7 is motivated by the interest of the research community in sentiment analysis of developers' communication traces (Novielli and Serebrenik, 2019; Novielli et al, 2019). Specifically, researchers investigated how developers share their emotions in the communication channels within collaborative development environments, including issue tracking systems (e.g., Jira) (Mäntylä et al, 2016; Ortu et al, 2015), software repository forges (e.g., GitHub) (Guzman et al, 2014; Pletea et al, 2014; Sinha et al, 2016), and technical Q&A sites (e.g., Stack Overflow) (Uddin and Khomh, 2017; Calefato et al, 2018b; Lin et al, 2019). What the above mentioned studies have in common is that they rely on sentiment analysis, – i.e. the study of the subjectivity (neutral vs. emotionally loaded) and polarity (positive vs. negative) of a text (Pang and Lee, 2008) – to mine emotions and opinions from textual developer-generated content. Beyond sentiment polarity, Gachechiladze et al. investigated the problem of identifying the target of anger expressions in collaborative software development (Gachechiladze et al, 2017). They distinguish between *self*-directed emotions, such as sadness, which do not involve a negative attitude towards the interlocutor, hostile language expressing anger towards *others*, and negative evaluation of *objects*, such as tools or programming languages. More recently, Raman and colleagues proposed an approach to mine the toxicity of developers' online conversations towards identification and mitigation of unhealthy interactions in open source software development (Raman et al, 2020). Toxicity has been also reported as a problem of Stack Overflow, which has been recognized as a hostile place especially for newcomers, women, non-native speakers, and others in marginalized groups by both users[24] and community managers.[25]

10 Conclusion

We conducted the first comprehensive mixed-methods study on how and why software developers adopt GitHub Discussions and how they perceive the new feature's usefulness. We further studied how Discussions compare to existing channels such as GitHub Issues, Pull Requests, and Stack Overflow threads. Based on an analysis of Discussions in 92 GitHub projects, a survey with developers already using the feature, and a comparison with related Stack Overflow posts, we derive the following recommendations for project managers. (i) *Set guidelines for participating in discussions,* as users sometimes find it

---

[24] https://meta.stackoverflow.com/q/262791
[25] https://stackoverflow.blog/2018/04/26/stack-overflow-isnt-very-welcoming-its-time-for-that-to-change/

difficult to choose an appropriate channel from Discussions, Issues, and Pull Requests. (ii) *Encourage core developers to participate in discussions,* as it is evident that the involvement of core developers is crucial in the successful use of GitHub Discussions.

References

Abdellatif A, Badran K, Shihab E (2020) MSRBot: Using bots to answer questions from software repositories. Empir Softw Eng 25(3):1834–1863, DOI 10.1007/s10664-019-09788-5, URL https://doi.org/10.1007/s10664-019-09788-5

Allamanis M, Sutton C (2013) Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In: Proc. of the 10th Working Conference on Mining Software Repositories, IEEE Press, MSR '13, pp 53–56

Aranda J, Venolia G (2009) The secret life of bugs: Going past the errors and omissions in software repositories. In: Proc. of the 31st International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, ICSE '09, p 298–308, DOI 10.1109/ICSE.2009.5070530, URL https://doi.org/10.1109/ICSE.2009.5070530

Arya D, Wang W, Guo JLC, Cheng J (2019) Analysis and detection of information types of open source software issue discussions. In: Proc. of the 41st International Conference on Software Engineering, IEEE Press, ICSE '19, pp 454–464, DOI 10.1109/ICSE.2019.00058, URL https://doi.org/10.1109/ICSE.2019.00058

Balali S, Annamalai U, Padala HS, Trinkenreich B, Gerosa MA, Steinmacher I, Sarma A (2020) Recommending tasks to newcomers in oss projects: How do mentors handle it? In: Proc. of the 16th International Symposium on Open Collaboration, Association for Computing Machinery, New York, NY, USA, OpenSym '20, DOI 10.1145/3412569.3412571, URL https://doi.org/10.1145/3412569.3412571

Baltes S, Diehl S (2019) Usage and attribution of stack overflow code snippets in github projects. Empir Softw Eng 24(3):1259–1295, DOI 10.1007/s10664-018-9650-5, URL https://doi.org/10.1007/s10664-018-9650-5

Baltes S, Dumani L, Treude C, Diehl S (2018) Sotorrent: reconstructing and analyzing the evolution of stack overflow posts. In: Proc. of the 15th International Conference on Mining Software Repositories, ACM, MSR '18, pp 319–330, DOI 10.1145/3196398.3196430, URL https://doi.org/10.1145/3196398.3196430

Beyer S, Pinzger M (2016) Grouping android tag synonyms on stack overflow. In: Proc. of the 13th International Conference on Mining Software Repositories, Association for Computing Machinery, New York, NY, USA, MSR '16, pp 430–440, DOI 10.1145/2901739.2901750, URL https://doi.org/10.1145/2901739.2901750

Beyer S, Macho C, Di Penta M, Pinzger M (2020) What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches

to classify posts into question categories. Empir Softw Eng 25(3):2258–2301, DOI 10.1007/s10664-019-09758-x

Borges H, Hora A, Valente MT (2016) Understanding the factors that impact the popularity of github repositories. In: Proc. of the 32nd IEEE International Conference on Software Maintenance and Evolution, ICSME '16, pp 334–344

Braun V, Clarke V (2006) Using thematic analysis in psychology. Qualitative research in psychology 3(2):77–101

Calefato F, Lanubile F, Maiorano F, Novielli N (2018a) Sentiment polarity detection for software development. Empir Softw Eng 23(3):1352–1382, DOI 10.1007/s10664-017-9546-9, URL https://doi.org/10.1007/s10664-017-9546-9

Calefato F, Lanubile F, Novielli N (2018b) How to ask for technical help? evidence-based guidelines for writing questions on stack overflow. Information & Software Technology 94:186–207, DOI 10.1016/j.infsof.2017.10.009, URL https://doi.org/10.1016/j.infsof.2017.10.009

Calefato F, Lanubile F, Novielli N, Quaranta L (2019) Emtk: The emotion mining toolkit. In: Proc. of the 4th International Workshop on Emotion Awareness in Software Engineering, IEEE Press, SEmotion '19, pp 34–37, DOI 10.1109/SEmotion.2019.00014, URL https://doi.org/10.1109/SEmotion.2019.00014

Chatterjee P, Damevski K, Pollock L (2021) Automatic extraction of opinion-based q&a from online developer chats. In: Proc. of the 43rd International Conference on Software Engineering (ICSE), IEEE, pp 1260–1272

Cleary B, Gómez C, Storey MA, Singer L, Treude C (2013) Analyzing the friendliness of exchanges in an online software developer community. In: Proc. of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), IEEE, pp 159–160

Cleveland WS, Loader C (1996) Smoothing by local regression: Principles and methods. In: Härdle W, Schimek MG (eds) Statistical Theory and Computational Aspects of Smoothing, Physica-Verlag HD, Heidelberg, pp 10–49

Cohen J (1988) Statistical Power Analysis for the Behavioral Sciences. Lawrence Erlbaum Associates

Dias E, Meirelles P, Castor F, Steinmacher I, Wiese I, Pinto G (2021) What makes a great maintainer of open source projects? In: Proc. of the 43rd International Conference on Software Engineering, ICSE '21, pp 982–994, DOI 10.1109/ICSE43902.2021.00093

Ebert F, Castor F, Novielli N, Serebrenik A (2019) Confusion in code reviews: Reasons, impacts, and coping strategies. In: Proc of the IEEE 26th International Conference on Software Analysis, Evolution and Reengineering, SANER '19, pp 49–60

Fleiss JL, Cohen J (1973) The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. Educational and Psychological Measurement 33(3):613–619, DOI 10.1177/001316447303300309

Gachechiladze D, Lanubile F, Novielli N, Serebrenik A (2017) Anger and its direction in collaborative software development. In: Proc. of the 39th In-

ternational Conference on Software Engineering: New Ideas and Emerging Results Track, IEEE Press, ICSE-NIER '17, pp 11–14, DOI 10.1109/ICSE-NIER.2017.18, URL https://doi.org/10.1109/ICSE-NIER.2017.18

Giuffrida R, 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

Guzman E, Azócar D, Li Y (2014) Sentiment analysis of commit comments in github: An empirical study. In: Proc. of the 11th Working Conf. on Mining Software Repositories, ACM, New York, NY, USA, MSR '14, pp 352–355, DOI 10.1145/2597073.2597118, URL https://doi.org/10.1145/2597073.2597118

Guzzi A, Bacchelli A, Lanza M, Pinzger M, Van Deursen A (2013) Communication in open source software development mailing lists. In: Proc. of the 10th Working Conference on Mining Software Repositories, IEEE, MSR '13, pp 277–286

Hata H, Todo T, Onoue S, Matsumoto K (2015) Characteristics of sustainable oss projects: A theoretical and empirical study. In: Proc. of the IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, IEEE Computer Society, USA, CHASE '15, pp 15–21, DOI 10.1109/CHASE.2015.9, URL https://doi.org/10.1109/CHASE.2015.9

Hata H, Treude C, Kula RG, Ishio T (2019) 9.6 million links in source code comments: Purpose, evolution, and decay. In: Proc. of the 41st International Conference on Software Engineering, IEEE Press, ICSE '19, pp 1211–1221, DOI 10.1109/ICSE.2019.00123, URL https://doi.org/10.1109/ICSE.2019.00123

Hata H, Novielli N, Baltes S, Kula RG, Treude C (2021) Research Artifact: An Exploratory Study of GitHub Discussions Early Adoption. DOI 10.5281/zenodo.5026134, URL https://doi.org/10.5281/zenodo.5026134

Hirao T, Kula RG, Ihara A, Matsumoto K (2019) Understanding developer commenting in code reviews. IEICE Transactions on Information and Systems E102.D(12):2423–2432

Hirao T, McIntosh S, Ihara A, Matsumoto K (2020) Code reviews with divergent review scores: An empirical study of the openstack and qt communities. IEEE Transactions on Software Engineering

Inokuchi A, Sulistyo Nugroho Y, Wattanakriengkrai S, Konishi F, Hata H, Treude C, Monden A, Matsumoto K (2019) From Academia to Software Development: Publication Citations in Source Code Comments. arXiv e-prints arXiv:1910.06932, 1910.06932

Islam MR, Zibran MF (2017) Leveraging automated sentiment analysis in software engineering. In: Proc. of the 14th International Conf. on Mining Software Repositories, IEEE Press, MSR '17, pp 203–214, DOI 10.1109/MSR.2017.9, URL https://doi.org/10.1109/MSR.2017.9

Jiang J, Yang Y, He J, Blanc X, Zhang L (2017) Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development. Inf Softw Technol pp 48–62

Lin B, Zampetti F, Bavota G, Di Penta M, Lanza M (2019) Pattern-based mining of opinions in q&a websites. In: Proc. of the 41st International Conference on Software Engineering, IEEE Press, ICSE '19, pp 548–559, DOI 10.1109/ICSE.2019.00066, URL https://doi.org/10.1109/ICSE.2019.00066

Maipradit R, Lin B, Nagy C, Bavota G, Lanza M, Hata H, Matsumoto K (2020a) Automated identification of on-hold self-admitted technical debt. In: Proc. of the IEEE 20th International Working Conference on Source Code Analysis and Manipulation, IEEE Computer Society, Los Alamitos, CA, USA, SCAM '20, pp 54–64, DOI 10.1109/SCAM51674.2020.00011, URL https://doi.ieeecomputersociety.org/10.1109/SCAM51674.2020.00011

Maipradit R, Treude C, Hata H, Matsumoto K (2020b) Wait for it: identifying "On-Hold" self-admitted technical debt. Empir Softw Eng 25(5):3770–3798, DOI 10.1007/s10664-020-09854-3, URL https://doi.org/10.1007/s10664-020-09854-3

Mäntylä M, Adams B, Destefanis G, Graziotin D, Ortu M (2016) Mining valence, arousal, and dominance: Possibilities for detecting burnout and productivity? In: Proc. of the 13th International Conf. on Mining Software Repositories, ACM, New York, NY, USA, MSR '16, pp 247–258, DOI 10.1145/2901739.2901752, URL https://doi.org/10.1145/2901739.2901752

Morrison P, Murphy-Hill E (2015) Is programming knowledge related to age? An exploration of Stack Overflow. In: Di Penta M, Pinzger M, Robbes R (eds) 12th Working Conference on Mining Software Repositories (MSR 2015), IEEE Computer Society, Florence, Italy, pp 69–72

Munaiah N, Kroh S, Cabrey C, Nagappan M (2017) Curating github for engineered software projects. Empir Softw Eng 22(6):3219–3253, DOI 10.1007/s10664-017-9512-6, URL https://doi.org/10.1007/s10664-017-9512-6

Murgia A, Tourani P, Adams B, Ortu M (2014) Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In: Proc. of the 11th Working Conf. on Mining Software Repositories, ACM, New York, NY, USA, MSR '14, pp 262–271, DOI 10.1145/2597073.2597086, URL https://doi.org/10.1145/2597073.2597086

Novielli N, Serebrenik A (2019) Sentiment and emotion in software engineering. IEEE Software 36(5):6–23, DOI 10.1109/MS.2019.2924013

Novielli N, Girardi D, Lanubile F (2018) A benchmark study on sentiment analysis for software engineering research. In: Proc. of the 15th International Conference on Mining Software Repositories, Association for Computing Machinery, New York, NY, USA, MSR '18, pp 364–375, DOI 10.1145/3196398.3196403, URL https://doi.org/10.1145/3196398.3196403

Novielli N, Begel A, Maalej W (2019) Introduction to the special issue on affect awareness in software engineering. Journal of Systems and Software 148:180–182, DOI https://doi.org/10.1016/j.jss.2018.11.016, URL http://www.sciencedirect.com/science/article/pii/S0164121218302504

Ortu M, Adams B, Destefanis G, Tourani P, Marchesi M, Tonelli R (2015) Are bullies more productive? empirical study of affectiveness vs. issue fixing time. In: Proc. of the 12th Working Conf. on Mining Software Repositories, IEEE Press, MSR '15, pp 303–313

Pang B, Lee L (2008) Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2):1–135, DOI 10.1561/1500000011, URL https://doi.org/10.1561/1500000011

Pascarella L, Spadini D, Palomba F, Bruntink M, Bacchelli A (2018) Information Needs in Contemporary Code Review. In: Proc. of the 21st ACM Conference on Computer Supported Cooperative Work, CSCW '18, vol 2, pp 135:1–135:27

Pletea D, Vasilescu B, Serebrenik A (2014) Security and emotion: Sentiment analysis of security discussions on github. In: Proc. of the 11th Working Conf. on Mining Software Repositories, ACM, New York, NY, USA, MSR '14, pp 348–351, DOI 10.1145/2597073.2597117, URL https://doi.org/10.1145/2597073.2597117

Potdar A, Shihab E (2014) An exploratory study on self-admitted technical debt. In: Proc. of the 2014 IEEE International Conference on Software Maintenance and Evolution, IEEE Computer Society, USA, ICSME '14, pp 91–100, DOI 10.1109/ICSME.2014.31, URL https://doi.org/10.1109/ICSME.2014.31

Rahman MM, Roy CK, Kula RG (2017) Predicting usefulness of code review comments using textual features and developer experience. In: Proc. of the 14th International Conference on Mining Software Repositories, MSR '17, pp 215–226

Raman N, Cao M, Tsvetkov Y, Kästner C, Vasilescu B (2020) Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In: Proc. of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results, Association for Computing Machinery, New York, NY, USA, ICSE-NIER '20, pp 57–60, DOI 10.1145/3377816.3381732, URL https://doi.org/10.1145/3377816.3381732

Robillard MP, Treude C (2020) Understanding wikipedia as a resource for opportunistic learning of computing concepts. In: Proc. of the 51st ACM Technical Symposium on Computer Science Education, Association for Computing Machinery, New York, NY, USA, SIGCSE '20, pp 72—-78, DOI 10.1145/3328778.3366832, URL https://doi.org/10.1145/3328778.3366832

Rosen C, Shihab E (2016) What are mobile developers asking about? a large scale study using stack overflow. Empir Softw Eng 21(3):1192–1223, DOI 10.1007/s10664-015-9379-3, URL https://doi.org/10.1007/s10664-015-9379-3

Sahar H, Hindle A, Bezemer CP (2021) How are issue reports discussed in gitter chat rooms? Journal of Systems and Software 172:110852

Sinha V, Lazar A, Sharif B (2016) Analyzing developer sentiment in commit logs. In: Proc. of the 13th International Conf. on Mining Software Repositories, ACM, New York, NY, USA, MSR '16, pp 520–523, DOI 10.1145/

2901739.2903501, URL https://doi.org/10.1145/2901739.2903501

Steinmacher I, Graciotto Silva MA, Gerosa MA, Redmiles D (2014) A systematic literature review on the barriers faced by newcomers to open source software projects. Information and Software Technology 59:67–85, DOI 10.1016/j.infsof.2014.11.001

Steinmacher I, Treude C, Gerosa MA (2019) Let me in: Guidelines for the successful onboarding of newcomers to open source projects. IEEE Software 36(4):41–49, DOI 10.1109/MS.2018.110162131

Storey MA, Ryall J, Bull RI, Myers D, Singer J (2008) Todo or to bug: Exploring how task annotations play a role in the work practices of software developers. In: Proc. of the 30th International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, ICSE '08, pp 251–260, DOI 10.1145/1368088.1368123, URL https://doi.org/10.1145/1368088.1368123

Storey MA, Zagalsky A, Figueira Filho F, Singer L, German DM (2016) How social and communication channels shape and challenge a participatory culture in software development. IEEE Transactions on Software Engineering 43(2):185–204

Sulistyo Nugroho Y, Islam S, Nakasai K, Rehman I, Hata H, Gaikovina Kula R, Nagappan M, Matsumoto K (2020) Sustaining a Healthy Ecosystem: Participation, Discussion, and Interaction in Eclipse Forums. arXiv e-prints arXiv:2009.09130, 2009.09130

Thelwall M, Buckley K, Paltoglou G, Cai D, Kappas A (2010) Sentiment strength detection in short informal text. J Am Soc Inf Sci Technol 61(12):2544–2558

Treude C, Barzilay O, Storey MA (2011) How do programmers ask and answer questions on the web? (nier track). In: Proc. of the 33rd International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, ICSE '11, pp 804–807, DOI 10.1145/1985793.1985907, URL https://doi.org/10.1145/1985793.1985907

Tsay J, Dabbish L, Herbsleb J (2014) Let's talk about it: Evaluating contributions through discussion in github. In: Proc. of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE '14, pp 144–154

Uddin G, Khomh F (2017) Opiner: An opinion search and summarization engine for apis. In: Proc. of the 32nd IEEE/ACM International Conf. on Automated Software Engineering, IEEE Press, ASE '17, pp 978–983

Vasilescu B, Capiluppi A, Serebrenik A (2012) Gender, Representation and Online Participation: A Quantitative Study of StackOverflow. In: Aberer K, Flache A, Jager W, Liu L, Tang J, Gueret C (eds) Proc. of the 4th International Conference on Social Informatics, Springer, Lausanne, Switzerland, SocInfo '12, pp 332–338

Viera AJ, Garrett JM (2005) Understanding interobserver agreement: the kappa statistic. Family medicine 37 5:360–3

Wang S, Lo David, Jiang L (2013) An empirical study on developer interactions in StackOverflow. In: Shin SY, Maldonado JC (eds) Proc. of the 28th Annual

ACM Symposium on Applied Computing, ACM, Coimbra, Portugal, SAC
'13, pp 1019–1024

Wang S, Chen TP, Hassan AE (2018) How do users revise answers on technical q&a websites? a case study on stack overflow. IEEE Transactions on Software Engineering 46(9):1024–1038

Yang D, Martins P, Saini V, Lopes CV (2017) Stack Overflow in Github: Any Snippets There? In: Gonzalez-Barahona JM, Hindle A, Tan L (eds) Proc. of the 14th International Conference on Mining Software Repositories, IEEE Computer Society, Buenos Aires, Argentina, MSR '17, pp 280–290