When Conversations Turn Into Work: A Taxonomy of Converted Discussions and Issues in GitHub

Dong Wang ✉ · **Masanari Kondo** ·
Yasutaka Kamei · **Raula Gaikovina**
Kula · **Naoyasu Ubayashi**

Abstract Popular and large contemporary open-source projects now embrace a diverse set of documentation for communication channels. Examples include contribution guidelines (i.e., commit message guidelines, coding rules, submission guidelines), code of conduct (i.e., rules and behavior expectations), governance policies, and Q&A forum. In 2020, GitHub released Discussion to distinguish between communication and collaboration. However, it remains unclear how developers maintain these channels, how trivial it is, and whether deciding on conversion takes time. We conducted an empirical study on 259 NPM and 148 PyPI repositories, devising two taxonomies of reasons for converting discussions into issues and vice-versa. The most frequent conversion from a discussion to an issue is when developers request a contributor to clarify their idea into an issue (*Reporting a Clarification Request –35.1% and 34.7%, respectively*), while agreeing that having non actionable topic (*QA, ideas, feature requests –55.0% and 42.0%, respectively*) is the most frequent reason of converting an issue into a discussion. Furthermore, we show that not all reasons for conversion are trivial (e.g., not a bug), and raising a conversion intent potentially takes time (i.e., a median of 15.2 and 35.1 hours, respectively, taken from issues to discussions). Our work contributes to complementing the GitHub guidelines and helping developers effectively utilize the Issue and Discussion communication channels to maintain their collaboration.

✉ Corresponding author - Dong Wang

Dong Wang, Masanari Kondo, Yasutaka Kamei, Naoyasu Ubayashi
Kyushu University, Japan
E-mail: {d.wang, kondo,kamei, ubayashi}@ait.kyushu-u.ac.jp

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

arXiv:2307.07117v1 [cs.SE] 14 Jul 2023

1 Introduction

Contemporary open-source projects nowadays employ a plethora of communication channels to facilitate knowledge sharing and sustain the community around them (Storey et al., 2016). Complementary studies have shown that GitHub projects tend to adopt multiple communication channels and they are used to both capture new knowledge and update existing knowledge (Tantisuwankul et al., 2019; Vale et al., 2020). As part of a community, the management of communication and collaboration channels has led to the increase in the sophistication of documentation standards such as contribution guidelines (e.g., commit message guidelines, coding rules, submission guidelines), code of conduct (e.g., rules and behavior expectations), governance policies, and Q&A forum.

Launched in 2020[1], GitHub Discussion is a community forum that serves as an asynchronous communication channel. The intended usage is for open-source communities, developer teams, and companies to ask questions, share ideas, and build connections with each other, all within the same GitHub platform. The first exploratory study by Hata et al. (2022) showed that Discussion is considered useful by developers and plays a crucial role in advancing the development of projects, uncovering several reasons for using GitHub Discussion mentioned in initial discussions (e.g., question-answering, community engagement, etc.). Formally, Discussion is designed as a tool to share questions, ideas, conversations, requests for comment (RFC), resource planning, and community engagement. This is different from the more traditional bug and code review systems such as GitHub Issue, which tracks executable pieces of work with a defined start and end point, including new features, fixing bugs, general updates, and tracking for epics and sprints, among other things. Used together, GitHub claims that one benefit is that a developer can reference a Discussion in an issue as background and context for a piece of work, while converting an issue into a discussion could be due to the lack of information and decisions needed to complete a task. Although GitHub provides guidelines[2], it remains unclear how developers maintain this intertwine between communication and actionable collaboration, how trivial is the conversion, and whether or not it takes time to decide on a conversion.

In this paper, we investigate the maintenance between communication and actionable collaboration by analyzing how developers decide between Discussion and Issue in real-world projects. Hence, we conduct an empirical study to mine the conversions between them from 259 NPM repositories and 148 PyPI repositories. Three research questions are formulated to guide this study:

– **RQ1: What is the reason of converting a discussion to an issue?**
  <u>Motivation:</u> Although the prior work (Hata et al., 2022) explored the adoption of GitHub Discussion in terms of the usage and perception by devel-

---

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

[2] https://resources.github.com/devops/process/planning/discussions/

opers, it is still unclear how contributors choose between the two communication channels. Specifically, Hata et al. studied the appearance of issue links and demonstrated that GitHub Discussions play a role in moving development forward by triggering new issues, but the intentions behind such triggers are not yet revealed. Answering this RQ would help the project to better maintain the communication channel and mentor newcomers to find a way to make contributions.

Results: Four reasons for suggesting a transition from discussions to issues are identified from a manual classification of a total of 331 samples, including Reporting a Bug, External Repository, Reporting a Clarification Request, and Reporting an Enhancement. Reporting a Clarification Request is the most common reason, with 35.1% and 34.7% of instances being classified for the NPM and the PyPI, respectively.

– **RQ2: What is the reason of converting an issue to a discussion?**
Motivation: As one strategy for moderating discussions, developers are allowed to convert an issue to a discussion. Hata et al. (2022) shows that around 18% of discussions are converted from issues. However, the reason behind this conversion is not widely explored. Answering this RQ would help contributors further understand the proper position of the issue tracker system and avoid unnecessary burdens for engineering developers.
Results: Through a manual classification of a total of 433 samples, seven reasons are identified for converting issues to discussions. Non actionable topic is the most frequent reason, with 55.0% and 42.0% of instances being classified for the NPM and the PyPI, separately. Furthermore, Question-answering is the main non actionable topic.

– **RQ3: How long does it take to raise a conversion intent?**
Motivation: We would like to explore whether deciding on a conversion from discussions to issues and vice versa takes time or not (i.e., discussion length and spent time). Answering this RQ would provide a potential future venue for the automatic classifier proposal to identify the appropriate topics between GitHub Issue and Discussion.
Results: The quantitative analysis shows that few posts are involved until the conversion intent is raised, i.e., mostly the median is one in both studied conversion kinds. However, raising the conversion intent could potentially take time. For instance, 15.2 hours and 35.1 hours (median) are taken until the conversion intent is raised from issues to discussions for the NPM and the PyPI, respectively.

Based on our empirical study results, we provide the following implications for the stakeholders. For *maintainers and contributors*, we find that our devised taxonomy complements the GitHub guidelines and further helps them decide when to contribute to each channel and reduce unnecessary conversions. Meanwhile, Discussion could be considered as a means to attract, and onboard potential new contributors. We recommend that project maintainers should clearly state the submission rules in their README or contribution guidelines to avoid the inconsistent use of communication channels. We also

Fig. 1 GitHub issue is converted into Discussion thread (`pixijs #7680`).

suggest contributors, especially newcomers, to start from the Discussion for the uncertainty problem (e.g., reporting potential bugs). For *researchers*, we provide the future direction of automatic classifier needs, including the detection of duplication between Issue and Discussion, the identification of non actionable topics from Issue, and bug-related topics from Discussion.

The remainder of this paper is organized as follows: Section 2 illustrates an example to motivate this study. Section 3 describes the repository selection and the discussion extraction. Section 4 presents the approach to the proposed questions. Section 5 shows the research results. Section 6 discusses the insights from our findings. Section 7 discloses the threats to validity and Section 8 presents the related work. Finally, we conclude the paper in Section 9.

2 Motivating Example

As pointed out by the survey feedback in the work of Hata et al. (2022), developers face the problem of topic duplication between Discussions and Issues.
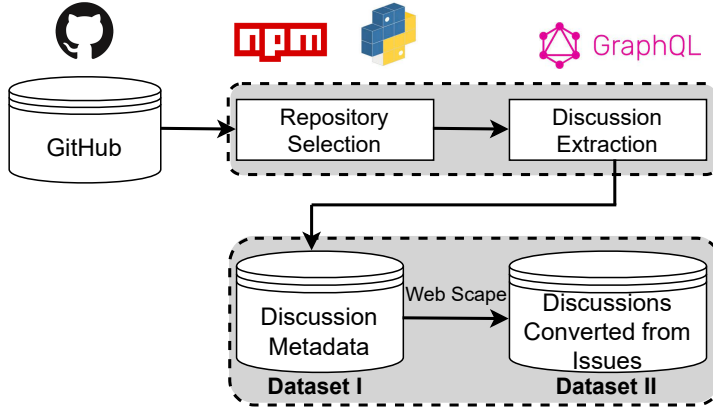
Fig. 2 The overview of dataset preparation.

Inspired by their work, we searched for some anecdotal evidence to understand the conversion process from Discussion to Issue and vice-versa.

Figure 1 shows an example from `pixijs #7680` where a GitHub issue was converted into the Discussion channel. As shown in the figure, the issue was initially entitled as process interactive bug on hover. To further confirm the bug or not, the author additionally provided the error message screenshot and the environment. Then, a conversation consisting of 22 posts was taken between four developers. During the half-conversation, three developers tried to investigate the causes and claimed that they found a way to resolve the raised issues. At the same time, they raised another potential problem around the pre-issue "the issue seems to be bound to our codebase, not reproducible on a clean project" and they further discussed this regarding the conflict of package versions. At the end of this conversation, the maintainer suggested that this issue had turned more into support and raised an intention of converting it to a discussion thread. In total, around two days were taken for this conversion process between the issue creation time (5 Aug 2021) and the conversion time (7 Aug 2021). Although the work of Hata et al. (2022) highlighted the challenge of choosing appropriate channels from the survey view, it is still unclear and lacks in-depth empirical analysis to reveal how the developers are using this new feature in real life. Inspired by this example, we hypothesize that (i) *the conversion process between Issue and Discussion may not be trivial*, and (ii) *the conversion process may take a long period in terms of discussion length and time.*

3 Studied Datasets

In this section, we describe the process of preparing studied datasets, including repository selection and discussion extraction.

Repository Selection. To answer our proposed RQs, we perform an empirical study on the NPM package and PyPI ecosystems. We selected the NPM and PyPI package ecosystems as (I) they are two of the largest package collections that are hosted on the GitHub platform and have been widely studied in the recent studies (Abdalkareem et al., 2017; Cogo et al., 2019; Chinthanet et al., 2021), (II) inspired by the work of Hata et al. (2022), 38% of web libraries and frameworks (i.e., the highest proportion) have adopted the discussion beta feature. Similar to the previous work (Chinthanet et al., 2021), we refer to the listing of NPM packages from the NPM registry and then matched them to the projects that are available in GitHub. For the PyPI ecosystem, we rely on the open-source discovery service Libraries.io[3] and the PyPI registry listing. We assume that the more active and well-maintained package repositories are more likely to adopt the discussion feature. Thus, we filtered and collected the repositories based on their contributor number (i.e., the number of contributors is larger than 100), resulting in 1,255 and 510 distinct repositories from the NPM and the PyPI, respectively. Then, we use GraphQL API[4] to determine whether or not these repositories have already adopted the Discussion feature. In the end, 263 NPM package repositories and 148 PyPI package repositories have introduced the GitHub Discussion, by the end of March 2022.

Discussion Extraction. For the remaining 411 repositories, we then used GraphQL API to retrieve all discussions, including metadata such as discussion title, body, author, created time, and whether the discussion has selected answer or not. For each post inside the discussion, we collected its body text, author, timestamp, and whether the post is an answer or not. Moreover, for each post, we as well collected its nested replies, including reply body, reply time, and reply author. After this, we were able to obtain 33,716 discussions from 259 NPM repositories (the rest of the four repositories have the Discussion feature but no discussions) with 68,695 individual posts and 69,593 replies in these posts, by April 2022. For the PyPI ecosystem, 12,662 discussions were retrieved from 148 repositories with 28,753 individual posts and their 29,001 replies. We regard this dataset as `Dataset I`, as shown in Figure 2. Since we would like to understand the characteristics and reasons of those discussion threads that are converted from the issues, we rely on the custom web scraper[5] provided by Hata et al. (2022) to determine whether the discussion thread was converted from an existing Issue. Specifically, we provide the standard input as required (repository names and their owners) and the scraper (i.e., HTML parser) would automatically output the related attributes of discussion threads including the information of whether the thread was converted in the form of binary values. Table 1 shows the statistic summary of our studied datasets. As shown in the table, 5,689 discussion threads (16.8%) and 3,337 discussion threads (26.4%) are converted from the existing issues for the NPM and the PyPI, separately, denoted as `Dataset II`.

---

[3] https://libraries.io/

[4] https://graphql.org/

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

Table 1 Summary of Studied Datasets.

	NPM	PyPI
Studied period	$\sim$ 2022.04	
# Package repositories	259	148
# Discussion threads	33,716	12,622
# Avg. Discussion threads per repositories	130	85
# Discussion posts	68,695	28,753
# Discussion threads converted from Issues	5,689 (16.8%)	3,337 (26.4%)
# Avg. Discussion threads converted from Issues	22	23

4 Approach

In this section, we will describe the approach of our proposed RQs.

4.1 RQ1 Analysis

To answer RQ1: What is the reason of converting a discussion to an issue?, we analyze a discussion that is suggested to be converted into an issue by looking at what is the conversion reason, using content analysis (one of the most broadly used qualitative data analysis methods) (Stemler, 2000).

Identifying discussions that are converted to issues. Since there are no available datasets and tools or classifiers that were provided in the literature to be used, we first have to identify discussions that are suggested to be converted to issues. To do so, we applied a semi-automatic method to identify such discussions from 46,388 discussion threads (33,716 and 12,622 discussion threads for the NPM and PyPI, respectively) and their posts (`Dataset I`), using a list of keywords. To ensure the keyword is sufficient, we first manually inspected a group of 100 discussions that contain issue related links and picked up the potential indicator words. Based on the observations and our knowledge, we refined the keyword and came up with the following keyword list: <`open, move, convert, create, please, transfer, submit, file, bug, issue`>, by taking case sensitive into account. Then we used this keyword list to match the discussion posts, resulting in 15,126 discussion threads (7,751 and 7,375 discussion threads for the NPM and PyPI, respectively) that at least contained one keyword in their posts. Next, the first author manually validated these posts to identify the true positives where the discussion thread is suggested to be converted to an issue. Finally, 595 discussion threads (374 and 221 discussion threads for the NPM and the PyPI, respectively) were collected as shown in Table 2.

Representative dataset construction. Similar to the prior work (Wang et al., 2021c; Chouchen et al., 2021), we then drew a statistically representative sample and the required sample size was calculated so that our conclusions

Table 2 Summary of Representative Dataset. # of validated Dis. refers to the number of the discussions that are retrieved from the keyword list. # of satisfied Dis. denotes the number of discussions that satisfy the criteria after the manual validation.

	Dis. from issues (RQ1)		Dis. into issues (RQ2)	
	NPM	PyPI	NPM	PyPI
# validated Discussions	7,751	7,375	1,156	523
# satisfied Discussions	374	221	562	438
# representative samples	190	141	228	205
Total		331		433

about the reasons of converted discussion threads would be generalized to all discussion threads in the same bucket with a confidence level of 95% and a confidence interval of 5. [6] Thus, we randomly selected 190 and 141 discussion threads from the NPM and the PyPI that are suggested to be converted into issues to conduct the subsequent analysis, as shown in Table 2.

Manually Coding Reasons. We performed a manual analysis to investigate the reasons behind those discussions that are suggested to be filed as issues. The manual analysis was conducted in multiple rounds, similar to the prior work (Hata et al., 2019). In the first round, the first two authors opened a round-table discussion on classifying twenty randomly selected discussion threads from the sample, and constructed an initial coding guide. Note that to classify the reasons, we not only relied on the specific discussion comments, but also referred to the context of the whole discussion threads and sometimes tracked back to the opened issues, in order to obtain a comprehensive understanding. To validate the coding guide and ensure that if there exist any missing codes, in the second round, the first two authors independently coded another twenty discussion threads. After this round, we found that the constructed coding guide can fit the sample and no new codes occurred. Similar to prior work (Wang et al., 2021c; Chouchen et al., 2021), we then calculated the Kappa agreement of this iteration between two authors across four codes for these twenty discussions. The score of the Free-marginal Kappa agreement is 0.80, implied as nearly perfect (McHugh, 2012). Based on the encouraging results, the first author then coded the rest of the samples.

4.2 RQ2 Analysis

To answer RQ2: What is the reason of converting an issue to a discussion?, we analyze a discussion that is converted from an issue in terms of the reasons behind such conversion, through content analysis same to RQ1.

---

[6] https://www.surveysystem.com/sscalc.htm

Representative dataset construction. To understand what is the reason of this kind of discussion conversion, similar to RQ1, we perform a manual analysis on a statistically representative sample of our discussion dataset (`Dataset II`) where the discussion threads are automatically identified as converted from issues or not by the crawler tool. Through an exploration of ten randomly selected samples, we observed that around 70% of these discussions do not imply explicit reasons from their post context. However, our study interest is to find the reasons and we would like to avoid the potential subjective threat. Thus, we then applied a filter to retrieve those discussion threads that could probably provide the reasons. To do so, we used the keyword `discussion` to narrow the data scope since we assume that the keyword `discussion` could indicate the discussion feature, resulting in 1,156 and 523 discussion threads for the NPM and the PyPI, separately. Then, the first author manually inspected these discussion threads to validate whether or not their comments imply the conversion between discussions and issues. We noticed that existing old issues can also be converted into discussion threads. We argue that these existing old issues are out of our scope, as we are interested in the instances where issues are converted due to the adoption of the Discussion feature. Therefore, we further excluded the converted issues that were submitted before the GitHub Discussion feature was firstly introduced (i.e., May 2020). In the end, 562 and 438 discussion threads from the NPM and the PyPI satisfied the criteria.

To be consistent with the manual analysis in RQ1, we drew a statistically representative sample. To reach the confidence level of 95% and a confidence interval of 5, we then randomly sampled 433 discussion threads (228 and 205 threads for the NPM and the PyPI, respectively) from the bucket. Table 2 shows the summary of the studied dataset in RQ2.

Manually Coding Reasons. We classify the reasons of converting issues to discussions, using statistically representative samples (433 discussion threads). Our initial codes were informed by taxonomy to understand the reasons for using GitHub Discussions mentioned in initial discussions, such as question-answering, idea sharing, information resource building, and so on (Hata et al., 2022). We refer to their taxonomy since it is closely relevant to our study scope, and the taxonomy is validated by the systematic process.

To test whether or not the existing taxonomy can fit our cases, in the first iteration, we randomly selected twenty samples and classified them into the available codes between the first two authors. After the classification, an open discussion was conducted between the two authors and we observed that the existing taxonomy cannot fit well. We then refined the coding schema by modifying certain codes and adding new codes. To validate our refined coding schema, another twenty samples were selected and the first two authors independently classified the samples. After the second iteration, we found that there existed new codes that were not covered in our coding schema. Then, an open discussion was held to discuss these new codes and further polish up our coding schema by emerging new codes. To assure that there was no new code and further evaluated the coding schema, in the third iteration, we selected

another twenty samples, and the first two authors independently coded them again. After the third iteration, no new codes occurred and we found that the coding schema can fit well the samples. In total, 60 samples were used to establish our coding schema of reasons of converting issues to discussions. Note that our annotation is based on the whole issue conversation and the guidelines do not allow for multiple categories. We then evaluated the inter-rater level of agreement between two raters across nine reasons, relying on the Kappa agreement. The Kappa agreement score is 0.72 (i.e., substantial agreement). Encouraged by this result, the first author then manually coded the rest of the 374 samples. We classify those instances that do not fit the above codes into Others. Then the above codes were merged into cohesive groups that can be represented by a similar subcategory (i.e., Question-answering, Idea sharing, and Feature request are merged into Non Actionable Topic), by conducting an open discussion among the four authors of this paper.

4.3 RQ3 Analysis

To answer RQ3: How long does it take to raise a conversion intent?, we perform a quantitative analysis on the manually labeled representative samples that are constructed from RQ1 and RQ2.

We define two metrics to conduct our statistical analysis, to understand the process of raising a conversion intent from discussion to issues and vice versa, as shown below:

- *Raising time (# hours):* The duration that is from the time when the discussion or issue is submitted to the time when the post firstly suggests that the discussion or issue should be converted.
- *Number of posts until the raising time:* The number of posts that are submitted until the first conversion intent is raised. Note that it includes the post in which the conversion suggestion is provided. For instance, in the motivating example presented in Section 2, the twenty-second post suggests that the issue topic should be converted to a discussion. Thus, in this case, we count *Number of posts until the raising time* as 22.

To assure that the post firstly raises the conversion intent, the first author manually validated 1,243 posts (740 and 503 posts for the NPM and the PyPI, respectively) and 1,938 posts (892 and 1046 posts for the NPM and the PyPI, respectively) from the studied discussions that are suggested to convert into issues (i.e., 331 samples in RQ1) or the discussions that are converted from issues (i.e., 433 samples in RQ2). We then measure these two metrics for those labeled discussion threads that are either converted from issues or converted to issues.

Meanwhile, to further understand the effect of different reasons, we propose a null hypothesis that *'Raising time and the number of posts until the conversion intent are significantly different among reasons of the conversion'*. To statistically confirm the significant differences, we use the Kruskal-Wallis H

test (Kruskal and Wallis, 1952). This is a non-parametric statistical test to be used when comparing two or more than two categories. In our study, there are four and seven main categories for RQ1 and RQ2, separately. The advantage of applying non-parametric statistical methods is that they make no assumptions about the distribution of the data (Hecke, 2012). In addition, we invoke a Mann-Whitney test (Mann and Whitney, 1947) to examine any significant difference in each pair category between NPM and PyPI ecosystems.

5 Results

In this section, we present the results of the empirical study.

5.1 Conversion from Discussions to Issues (RQ1)

To answer RQ1, we analyze (I) the reasons of converting discussions to issues, and (II) the frequency of these reasons across our studied samples. Below, we first provide representative examples for each reason type, and then we discuss the result of the frequency of reasons (Table 3 and Table 4).

Taxonomy of Reasons. Four reasons of converting discussions to issues are classified through our qualitative analysis, which is described in Section 4.1:

(I) Reporting a Bug. This category refers to the reason where the discussion post indicates that the discussion topic describes a bug. In this category, the keyword "bug" is usually left on the post, suggesting that the submitted post is explicitly related to the bug. For instance, in the Ex 1[7], one collaborator pointed out that the discussion topic (entitled as "prisma generates creates a package-lock.json") was indeed a bug and encouraged the author to open an issue in the repository, along with a directory structure sharing.

> Ex 1
> **Collaborator:** This doesn't sound right and it's definitely a **bug**. Could you please **open an issue** and share your directory structure in the monorepo and which directory you are running prisma generate from? Ideally, even a Git repository with minimal example that mimics your monorepo layout and triggers this bug. Thanks!

(II) External Repository. This category emerges by grouping discussion threads in which the post indicates that the discussion topic should not be in the current repository, instead should be opened as an issue in another repository. We observe that in this category, the appropriate repository that should be referred is always specifically provided by the collaborator. As shown in the Ex 2[8], a collaborator from the `docusaurus` repository posted a comment to make the author aware that the `openapi` plugin faces the incompatible issues

---

[7] https://github.com/prisma/prisma/discussions/10488
[8] https://github.com/facebook/docusaurus/discussions/6099

with the latest chance, and suggested the author to open an issue in plugin related repository.

> Ex 2
> **Collaborator:** Hi, it seems the openapi plugin is using the docusaurus internals which is not compatible with the latest change. Please **open an issue** in **their repo** telling them that the constants have been moved to @docusaurusutils.

(III) Reporting an Enhancement. This category denotes the reason where the post indicates that the idea or feature request or any warning that should be highlighted in the issue to be aware. For example, in the Ex 3[9], an author submitted a discussion to ask questions regarding `prefer-destructuring undestructurable`. One collaborator suggested the author to use slint-disabl, and also encouraged the author to file an issue to enhance this (i.e., ignore these cases by default).

> Ex 3
> **Collaborator:** you can just use eslint-disable. :) it sounds a reasonable **enhancement** to ignore these cases by default (or behind an option). can you **file an issue**, thanks!

(IV) Reporting a Clarification Request. We define this category as the post indicates that the topic stated in the discussion thread is not clear or lacks of details, and an issue is suggested in order to better understand the problem (e.g., add reproduce examples) or better track the progress how the problem evolves. We show the following two representative examples to describe this reason. In the Ex 4[10], the collaborator was unsure about the proposed error message. To further clarify this, the collaborator suggested the author to create an issue with a small reproduction. In another example Ex 5[11], to investigate the problem reason, resulting from the mixed versions or not, the collaborator encouraged the author to open an issue.

> Ex 4
> **Collaborator:** i'm **unsure** if we can get a better error message but could you maybe **create an issue** with a small reproduction?

> Ex 5
> **Collaborator:** please **open an issue** and follow the issue template. it looks like you **might** have mixed versions of mongoose and mongodb.

Frequency of Reasons. We now examine what is the common reason of converting discussions to issues. Table 4 presents the distribution of the reason category in our studied NPM and PyPI repositories. We observe that

---

[9] https://github.com/eslint/eslint/discussions/14669
[10] https://github.com/gatsbyjs/gatsby/discussions/32147
[11] https://github.com/Automattic/mongoose/discussions/10516

Table 3 Descriptions of reasons for converting discussions to issues.

Category	Description
(I) Reporting a Bug	The comment indicates that the discussion topic describes a bug.
(II) External Repository	The comment indicates that the discussion topic should not be reported in the current repository, instead should be reported in another repository (e.g., dependent repository).
(III) Reporting an Enhancement	The comment indicates that an idea or a feature request or any warning is needed to be highlighted in the issue.
(IV) Reporting a Clarification Request	The comment indicates that to trigger more details and further confirm the problems (e.g., potential bugs), an issue is needed to follow up the discussion.

Table 4 Frequency of reasons for converting discussions to issues.

	NPM	PyPI
(I) Reporting a Bug	44 (23.4%)	32 (22.7%)
(II) External Repository	44 (23.4%)	22 (15.6%)
(III) Reporting an Enhancement	34 (18.1%)	38 (26.9%)
(IV) Reporting a Clarification Request	66 (35.1%)	49 (34.7%)

Reporting a Clarification Request is the most common reason for both package ecosystems, with 66 (35.1%) and 49 (34.7%) discussion threads being classified, separately. Such a result indicates that the discussions that should be converted to issues are more likely to result from uncertain discussion topics (e.g., potential bugs), which request further clarification from the issue tracker support. The following two reasons are the second most popular for the NPM repositories, i.e., *Reporting a Bug* and *External Repository*. 44 discussion threads were manually classified for both cases, accounting for 23.4% of instances. While, for the PyPI repositories, the following frequent reason is *Reporting an Enhancement*, with 26.9% of instances being classified.

> **RQ1 Summary**
>
> Four reasons are classified behind converting discussions to issues, including Reporting a Bug, External Repository, Reporting a Clarification Request, and Reporting an Enhancement. Reporting a Clarification Request is the most common reason for the two studied ecosystems (NPM and PyPI), accounting for 35.1% and 34.7% of instances, respectively.

5.2 Conversion from Issues to Discussions (RQ2)

To answer RQ2, we analyze (I) the reasons of converting issues to discussions, and (II) the popularity of the classified reasons. We now illustrate the reasons using representative instances and discuss the frequency results (Table 5 and Table 6).

Taxonomy of Reasons. Nine reasons of converting issues to discussions are classified through our qualitative analysis, which is described in Section 4.2:

(I) Non Actionable Topic. This category relates to the reason where the topic proposed in the issue is not actionable and does not fit the issue scope. The category includes three sub-codes: *Question-answering*, *Feature requests*, and *Idea sharing*. For example, in the Ex 1[12], the author submitted an issue to ask for *"DataStore: How to handle a partial sync up to AppSync?"*. While the maintainer left the comment and considered this topic was more of a general question that should be in the discussion area. Thus, we classify its reason into *Question-answering*. In another example Ex 2[13], the author proposed an issue to report a problem regarding query result sharing (i.e., *Unable to select Query X in pane*), attached with the screenshots showing unexpected results. However, a collaborator pointed out that this issue should be converted into the discussion as a feature request.

Ex 1

Maintainer: Hey Thanks for raising this! After reading over this issue, it appears to be **more of a general question** or topic for discussion and will be labeled as such. This is to differentiate it from the other types of issues and make sure it receives the attention it deserves.

...

Ex 2

Collaborator: Hello @<username>, thanks for reporting this. The feature is working as intended as it allows to share all the results in a panel, also as described in the docs. I do agree however that it may be an interesting feature. I'm converting this to a discussion where we **track feature requests**.

(II) Invalid Issues. This category is merged by two cohesive codes (i.e., *Lack of description information*, *Not follow the template*), referring to the reason where the reported issue lacks of sufficient information for other developers to investigate. The example Ex 3[14] illustrates a scenario regarding *Lack of description information*. As we can see, the maintainer can not understand the issue fully with only provided code and suggested the author to create a minimal live example. Furthermore, this issue was then moved to discussion due to its invalidity.

---

[12] https://github.com/aws-amplify/amplify-js/discussions/8106

[13] https://github.com/grafana/grafana/discussions/46356

[14] https://github.com/logaretm/vee-validate/discussions/3723

> Ex 3
> **Maintainer:** Please **create a minimal live example** on codesand-box, I can't guess the issue from this code alone. Also moved this to discussions since this doesn't satisfy an "issue" report.

(III) Not a Bug. This denotes the reason where the author proposes a bug report, while the developer does not agree that it should be a bug. For instance, in the example Ex 4[15], the author followed the issue template including unexpected behavior and reproduction steps to report a potential bug related to password reset. However, the maintainer did not agree with this and argued that this was not a bug, instead due to design and changed it to a discussion.

> Ex 4
> **Maintainer:** Changing this to a discussion as this is **not a bug** and is by design.
> Recover password is used in scenarios when a user has forgotten the password, as such should not invalidate other sessions. In fact that is pretty common practice. If a user suspect the account has been compromised they will update the credentials through the account console, which gives the option to logout existing sessions.

(IV) Further Discussion. This category refers to the reason where the issue requires further confirmation or additional feedback from GitHub Discussion. As shown in the Ex 5[16], to further validate the problem cause of *"Serial-Port.write(): callback never called"*, the maintainer moved the issue to the discussion system.

> Ex 5
> **Maintainer:** I'm going to convert this to our new discussion system until we **confirm** some sort of bug

(V) Already Fixed. This category denotes the reason where the comment indicates that the issue has already been raised in either the issue lists or the discussion thread. As shown in the Ex 6[17], the maintainer suggested that the pitfalls related to build has been addressed by adding a doc page and then converted the existing issue into a discussion thread.

> Ex 6
> **Maintainer:** Since we **added** a doc page about these pitfalls and there's **nothing to "fix"** here I'll move this to discussion :)

(VI) External Repository. This reason refers to the case where the comments indicate that the issue is not raised in the appropriate place. For instance, in the Ex 7[18], the author proposed an issue concerning task tracker app in the create-react-app repository. However, the collaborator considered that the issue was from CRA tool and further this issue was converted.

---

[15] https://github.com/keycloak/keycloak/discussions/8988

[16] https://github.com/serialport/node-serialport/discussions/2287

[17] https://github.com/gatsbyjs/gatsby/discussions/31283

[18] https://github.com/facebook/create-react-app/discussions/11405

Table 5 Descriptions of reasons for converting issues to discussions.

Category	Description
(I) Non Actionable Topic Question-answering Idea sharing Feature request	The comment indicates that the topic proposed in the issue is not actionable and does not fit the issue scope.
(II) Invalid Issues Lack of description Not follow the template	The comment indicates that the reported issue lacks of sufficient information for the developers to investigate.
(III) Not a Bug	The comment indicates that the bug proposed by the author does not get recognized by developers.
(IV) Further Discussion	The comment indicates that the issue requires further confirmation or additional feedback from GitHub discussion.
(V) Already Fixed	The comment indicates that the issue has already been raised in either the issue lists or the discussion thread.
(VI) External Repository	The comment indicates that the discussion topic should not be reported in the current repository, but instead should be reported in another repository (e.g., dependent repository)
(VII) Information Storage	The comment indicates that the issue contents should be kept as a reference for the community.

Table 6 Frequency of reasons for converting issues to discussions.

	NPM	PyPI
(I) Non Actionable Topic	**121 (55.0%)**	**86 (42.0%)**
Question-answering	76 (34.5%)	51 (24.8%)
Idea sharing	24 (10.9%)	20 (9.8%)
Feature request	21 (9.5%)	15 (7.3%)
(II) Invalid Issues	**19 (8.6%)**	**34 (16.6%)**
Lack of description	17 (7.7%)	32 (15.5%)
Not follow the template	2 (0.9%)	2 (1.0%)
(III) Not a Bug	**45 (20.5%)**	**38 (18.4%)**
(IV) Further Discussion	**10 (4.5%)**	**17 (8.3%)**
(V) Already Fixed	**7 (3.1%)**	**6 (2.9%)**
(VI) External Repository	**13 (5.9%)**	**7 (3.4%)**
(VII) Information Storage	**5 (2.2)**	**17 (8.3%)**

Ex 7
Collaborator: Hi @<username>, I converted this issue into an discussion as the issues are more for **CRA tooling** internal issues.

(VII) Information Storage. It means that the issue is converted since the comment indicates that the existing issue should be kept as a reference for the

community. For example, in the Ex 8[19], the author raised a problem regarding react-native formatting and seeked for the solution. One collaborator provided a couple of solutions and later moved this issue into a discussion for other developers to find it easily.

> Ex 8
> **Collaborator:** Going to move this to a discussion so it's **easier for people to find**.

Frequency of Reasons. Table 6 shows the reason frequency of converting issues to discussions. As shown in the table, we observe that *Non actionable topic* is the most common reason category within two studied package ecosystems, with 121 instances (55.0%) and 86 instances (42.0%) being classified for the NPM and the PyPI, separately. Upon closer look, *Question-answering* is the main non actionable topic, accounting for 34.5% and 24.8% of the occurrences of the *Not actionable topic* category for the NPM and the PyPI, respectively. The second most frequently occurring reason category is *Not a bug* (20.5% and 18.4% for the NPM and PyPI, respectively), where the identified bug by the author is not recognized by other developers. Least frequently occurring pattern includes the *Already fixed* reason, with 3.1% and 2.9% of instances being classified for two ecosystems.
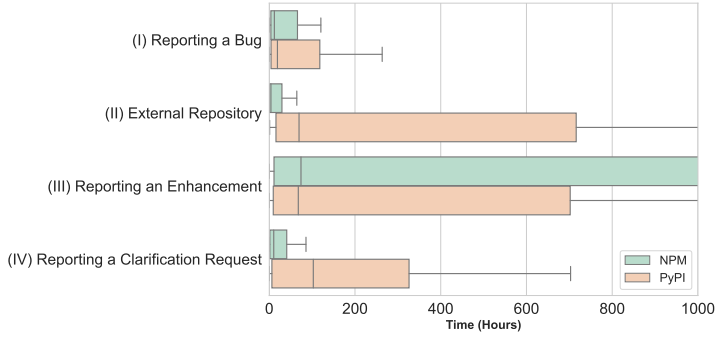
> RQ2 Summary
>
> Seven reasons are identified for converting issues to discussions. Non actionable topic is the most frequent reason, with 55.0% and 42.0% of instances being classified for the NPM and the PyPI, separately. Furthermore, Question-answering is the main non actionable topic.

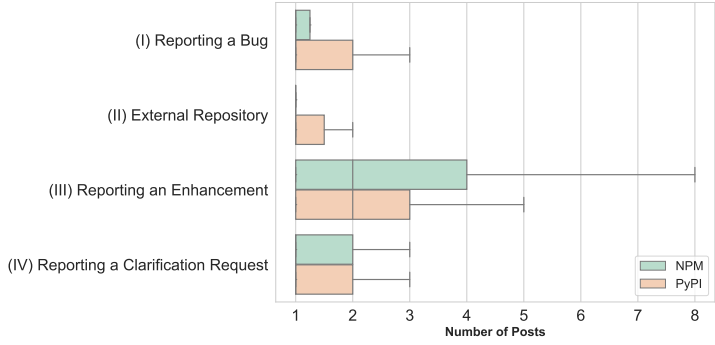5.3 Analysis of the Conversion Process (RQ3)

To answer RQ3, we analyze the process of the conversion from discussion to issues and vice versa, in terms of the following two aspects: (I) the raising time of conversion intent and (II) the number of post until the raising time. We below present the two metric related results (Figure 3 and Figure 4) with regard to two conversion kinds.

(I) Conversion from discussion to issue. Figure 3 presents the results of the computed metrics for the discussions that are suggested to be converted into issues. As shown in Figure 3 (a), on the one hand, we find that two ecosystems share a similar raising time in terms of categories *Reporting an Enhancement* and *Reporting a Bug.* For example, we observe that *Reporting an Enhancement* category takes a relatively long time (i.e., the medians of raising time are 74 hours and 67.8 hours for the NPM and the PyPI, respectively) when the conversion intent is raised when compared to the other three reason
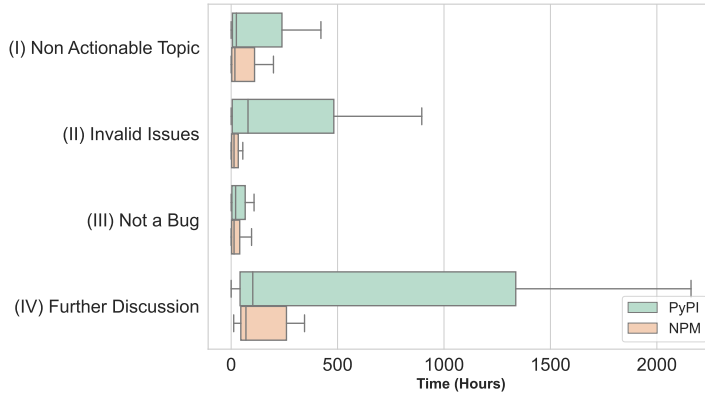
---

[19] https://github.com/date-fns/date-fns/discussions/2841
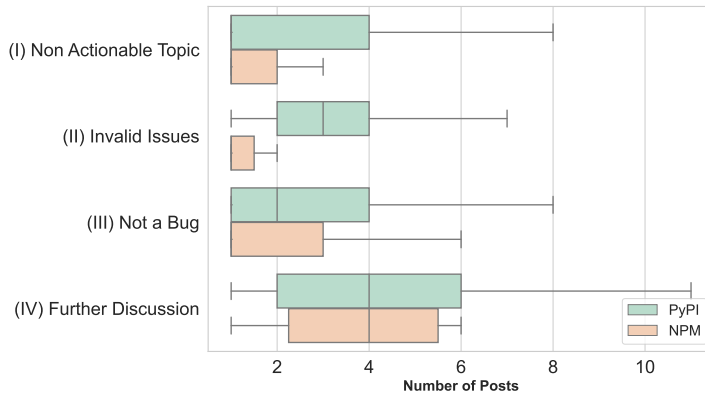
(a) Raising time of the conversion intent.

(b) Number of posts until the raising time.

Fig. 3 Discussions that are converted into issues: (I) raising time and (II) number of posts until the raising time.

categories. Such a result indicates that it may take time for developers to discuss and reach a consensus to propose an enhancement in the issue tracker. On the other hand, significantly large differences across two ecosystems are observed in terms of categories *External Repository* and *Reporting a Clarification Request*, validated by the Mann-Whitney test with *p-value* < 0.05. For instance, *External Repository* category takes the least time to be identified for the NPM, i.e., the median of raising time is 3.9 hours. While, for the PyPI, it takes almost a median of 69.5 hours to be notified of the external repository. One possible reason is that the NPM packages are less likely to be isolated when compared to the PyPI ones (Decan et al., 2016). For the metric related to the number of posts, as shown in Figure 4 (b), the median posts of *Reporting an Enhancement* category for both ecosystems are two which are larger than another three reason categories, suggesting that more discussions are likely to be involved in this instance. However, the Mann-Whitney test suggests that there is no significant difference for all the paired categories between the NPM and the PyPI.

(a) Raising time of the conversion intent.

(b) Number of posts until the raising time.

Fig. 4 Discussions that are converted from issues: (I) raising time and (II) number of posts until the raising time.

For the statistical test, Kruskal-Wallis H tests confirm that the hypothesis *'Raising time and the number of posts until the conversion intent are significantly different among reasons of the conversion'* is established in terms of the conversion from discussions to issues, with p-value < 0.001 for the *Raising time* and p-value < 0.05 for the *Number of posts* for the NPM repositories. However, the hypothesis is not supported for both raising time and the number of posts within the PyPI repositories.

(II) Conversion from issue to discussion. Figure 4 shows the results of the computed metrics for the discussions that are suggested to be converted from issues. Note that we only analyze the relatively frequent reasons for the two studied ecosystems (i.e., those instances whose frequencies are greater than 10 in Table 5). As shown in Figure 4 (a), we observe that it takes relatively a long time to receive the convention intent, i.e., the median is around 15.2 hours and 35.1 hours for all reason codes within the NPM and the PyPI, sepa-

rately. More specifically, 17.5 hours and 24.5 hours (median) are taken for *Non Actionable Topic* for the NPM and the PyPI, respectively. At the same time, we observe that there exists a significant difference via the Mann-Whitney test in terms of the category *Invalid Issues* between the two ecosystems. It takes a much longer time for developers to raise this intent in the PyPI, i.e., a median of 79 hours. On the other hand, as shown in Figure 4 (b), few posts are involved until the conversion intent of *Non Actionable Topic* is raised, with the median being one for both ecosystems. This result suggests that this conversion intent is likely to be raised in the first post of an issue. Unsurprisingly, *Further Discussion* category involves the most posts (i.e., the median is four for two ecosystems). Furthermore, the Mann-Whitney test confirms that there is a significant difference in categories *Non Actionable Topic* and *Invalid Issues* between the NPM and the PyPI. This suggests that relatively more posts are submitted to decide these two conversions with the PyPI.

For the statistical test, Kruskal-Wallis H tests confirm that the hypothesis *'Raising time and the number of posts until the conversion intent are significantly different among reasons of the conversion'* is established for the NPM repositories in terms of the conversion from issues to discussions, with p-value < 0.05 for both the *Raising time* and the *Number of posts*. For the PyPI repositories, a significant difference is observed for the *Number of posts* but is not observed for the *Raising time*.

> ### RQ3 Summary
>
> Our results show that there are few posts involved until the conversion intent is raised, i.e., mostly the median is one in both studied conversion kinds. However, we observe that deciding on a conversion may potentially take time. For instance, the median time of the conversion from issues to discussions is 15.2 hours and 35.1 hours for the NPM and the PyPI ecosystem, respectively.

6 Implications

Based on the results of our RQs, we now discuss the implications of the study and provide suggestions accordingly.

Reduce unnecessary conversion in the first place. Findings from the study illustrate a variety of reasons behind conversions, i.e., four reasons of converting discussion to issues (RQ1) and seven reasons of converting issues to discussions (RQ2). GitHub guidelines pointed out the hint that the topics like questions and ideas should be raised in the GitHub Discussion. While, we also find that other reasons exist, for instance, *invalid issues* and *not a bug*. These findings could complement the GitHub Discussion guidelines to guide those repositories that intend to adopt the GitHub Discussion feature. To maintain these channels, our constructed taxonomy can help maintainers and contributors decide when to contribute to each channel.

We further performed an additional analysis to investigate whether or not the inconsistent use of these two channels exists. In the context of feature requests (between *Reporting an Enhancement* and *Non Actionable Topic*), first, we observe that inconsistency does exist in the specific repositories. For example, within a repository namely *next.js*, on the one hand, a discussion[20] that requested adding custom attributes was suggested to be opened as an issue. On the other hand, an issue that requested a configuration setting was converted into a discussion and the author expressed confusion "Hey team, why did a feature request become a discussion?...". Similarly, another author from the discussion[21] doubted "Is it normal that feature requests are moved to a discussion?". Second, we notice that different repositories tend to have their own rules. For example, a collaborator from the *superset* repository left a comment[22] "Moving this feature request to Github Discussions so we can keep Issues focused on bugs!", while another repository *ant-design* allows feature requests to be proposed as one maintainer suggested in a discussion[23] "Since this not a bug or feature request. Move to discussion instead.". Such inconsistent use is also observed in the cases where issues should be created until enough information is provided or not (between *Reporting a Clarification Request* and *Invalid Issues*). Given the ecosystem PyPI, we observe that 28 out of 32 instances classified into *Invalid Issues* are from the *airflow* project. This may indicate that this project could use discussions as a gatekeeping mechanism. At the same time, we also notice that inconsistent use of two different channels did exist in the *airflow* project. For example, in this instance[24], the contributor suggested the author to open an issue from the discussion thread with some replication information (i.g., DAG). Based on these insights, to relieve the confusion for contributors and improve the user experience, we suggest that project maintainers should clearly state the submission rules in their project README or contribution guidelines.

A side-effect of keeping separate channels is duplication. Our empirical observations are align with the survey insights where the developers face the challenge of the duplication (Hata et al., 2022). We find that duplication indeed exists either between GitHub Discussions, or between GitHub Discussions and Issues. In terms of duplication between GitHub Discussions and Issues, for example, in this discussion [25], the maintainer left the post and pointed out that *"Already tracking here #4283 - please check issues first"*. Duplicate software artifacts have been proven to cause additionally unnecessary efforts and reduce efficiency. Many techniques are proposed to detect duplicate artifacts using information retrieval, such as issue report (Nguyen et al., 2012; Hindle et al., 2016), pull request (Li et al., 2017; Wang et al., 2019) on GitHub. We notice that the latest work has started to explore the related posts (duplicate or

---

[20] https://github.com/vercel/next.js/discussions/12325

[21] https://github.com/vercel/next.js/discussions/27756

[22] https://github.com/apache/superset/discussions/19185

[23] https://github.com/ant-design/ant-design/discussions/29818

[24] https://github.com/apache/airflow/discussions/14315

[25] https://github.com/invertase/react-native-firebase/discussions/4290

near duplicate) in GitHub Discussions and proposes an approach based on a SentenceBERT pre-trained model (Lima et al., 2022). Hence, another challenge would be the detection and removal of such duplication between these different channels.

Furthermore, the study shows evidence that GitHub Discussion not only facilitates communication but also can lead to actionable contributions to the project. For instance, our manual classification in RQ1 shows that these contributions are also significant, i.e., 76.6% of samples being classified for *Reporting a Clarification Request*, *Reporting a Bug*, and *Reporting an Enhancement* reason categories for the NPM ecosystem (Table 3). We point out that maintainers may consider Discussion as a means to attract and onboard potential new contributors, as Discussion can be acted as an incubation for those uncertain problems (e.g., potential bugs). After the evaluation by the project developers, these initial discussions have opportunities to be converted into actionable contributions.

Conversion is not trivial. Through the manual analysis of RQ2, our results show that some conversions may not be trivial to be decided. For instance, we observe that *Not a Bug* reason category accounts for up to 20.5% and 18.4% for the two ecosystems (Table 5). This indicates that the bug proposed by the contributor is not true from the engineering developer's perspective and thus it is converted into a discussion. We argue that such a type of conversion is not trivial and would take up time and pipeline for the developers to investigate the real causes. For instance, in the example[26], the contributor originally submitted an issue (including bug description and bug related logs). While, after the confirmation, the maintainer commented that *"This is not a bug. It's working as intended. Please open a discussion first. Maybe we can allow this as a feature request."*. Based on this insight, we suggest the contributor, especially those newcomers, to start from the GitHub Discussion for the uncertainty problem. At the same time, one potential future research direction would include the tendency analysis of non-trivial conversions. Some conversions would be converted back and forth, thus it is valuable to know what kinds of discussions or issues are difficult to get a consensus, especially bug related topics. In addition, the fact exists that invalid issues could be closed while they are not converted to Discussion. Hence, another direction for researchers is to investigate how common it is and the reasons behind this fact, which would gain a deeper insight into the maintenance of channels.

Raising conversion intent potentially takes time. In RQ3, a quantitative analysis was conducted to look at the conversion process. Although few discussions are involved, the time to receive the conversion intent is relatively long (Figure 3 and Figure 4). On the one hand, for those non-trivial reasons (e.g., *Reporting an Enhancement* and *Not a Bug*), we argue that such a long process would not be avoidable since these discussion/issue topics may require time and conversations to confirm their significance and correctness, regardless of the channel it happens on. Thus, we recommend developers that they

---

[26] https://github.com/renovatebot/renovate/discussions/14457

should accommodate the development pipeline of the contributed projects. On the other hand, for those trivial reasons (e.g., *Non Actionable Topic* and *External Repository*), we find that it still takes almost 17.5 hours and 24.5 hours for the issues related to non actionable topics to be converted into discussions for the NPM and the PyPI. One potential reason could be that these topics are not being paid attention to by the community members. Hence, a future direction for the researchers is to investigate whether the authorships of discussions or issues play a role in the conversion process. Interestingly, our statistical tests indicate that raising conversion intent within the PyPI likely takes a longer time and involves more posts than the NPM in some specific categories, e.g., *External Repository, Invalid Issues.* Therefore, we encourage researchers to further explore how the nature of the ecosystem will affect the conversion process. Meanwhile, we argue that such a long process of trivial ones could be wasteful, hence another direction is to call for a promising classifier that is able to identify these non actionable topics effectively to make the issue tracker lighter and save the developers' efforts.

7 Threads to Validity

Below, we describe the threats to the validity of this study:

External validity. External validity is with regard to the ability to generalize based on our results. In this study, we conduct a case study of NPM package repositories (web libraries) that adopt the Discussion feature. Thus, the observations based on this case study may not be generalized to other kinds of repositories (e.g., system software). However, our goal is not to build a theory to fit all repositories, but rather to shed light on the challenge that developers face during the choice between GitHub Issue and Discussion. Nonetheless, additional replication studies would help to generalize our observations in other repository domains, such as software tools, application software, and Non-web libraries and frameworks. Thus, to encourage future replication studies, we disclose our research materials and provide a replication package online, including raw NPM repository discussion data, manually labeled data, and the script to retrieve the discussions. The package is available: `https://github.com/posl/GitHub_Discussion_Conversion`.

Construct validity. Construct validity refers to the degree to which our measurements capture what we aim to study. During the data collection of those discussions that should be converted into issues, we rely on the heuristic approach using a list of keywords (e.g., open, transfer, convert, and so on) to filter the discussion posts. The threat may occur due to the incompleteness of the initialized keyword list. To mitigate this threat, we manually explored the random 100 discussion posts, and inspected the potential keywords that were highly frequent. At the same time, the goal is not to retrieve all these kinds of discussions, instead, we aim to get insights from a sufficient sample size.

Thus, we believe that our sample size is sufficient enough to provide insightful observations.

Internal validity. Internal validity denotes the approximate truth about inferences regarding cause-effect or causal relationships. Three threats are summarized. First of all, to understand the reasons behind the conversion, we performed a manual analysis, which may be mislabeled due to its subjective nature. To relieve such a threat, we conducted the manual coding in multiple iterations with two authors and calculated the Kappa agreement scores to ensure the quality. Once the scores suggested nearly perfect or substantial agreement, the first author then coded the rest of the samples. Additionally, to separate category *Reporting a Bug* and category *Reporting a Clarification Request*, we refer to whether the possibility-related word is given in a comment. This is likely to introduce a bias due to the natural usage of English words. The second threat occurs in the metric computation (RQ3). Since there is no automatic method to identify the exact time of the conversion, we rely on the post time when the conversion intent is raised. The third threat may exist in the selection of statistical tests. To test the significance of the studied metrics among different conversion reasons (RQ3), we use the Kruskal-Wallis H test. The cause-effect may differ from the other statistical tests. We are however confident, as the selected test is widely used in the prior study (Chinthanet et al., 2021; Wang et al., 2021a).

8 Related Work

In this section, we position our work with respect to the literature on question and answer forum, developer communication in software development, and barrier for newcomers in OSS projects.

8.1 Question & Answer Community

Developers often turn to programming question and answer (Q&A) communities to seek for help with their codes. Stack Overflow, as one of the most popular Q&A communities (Wang et al., 2023), has become a gold mine for software engineering research and is found to be useful for software development. Treude et al. (2011), as a pioneer work, categorized the kinds of questions that are asked, and explored which questions are answered well and which ones remain unanswered. Vasilescu et al. (2012) provided a quantitative study of the phenomenon, in order to assess the representation and social impact of gender in Stack Overflow. Treude and Robillard (2017) conducted a survey-based study to understand developers' information needs as they relate to code fragments. A large body of studies targets the knowledge extraction and challenges of specific topics from Stack Overflow. To facilitate program repair, Liu and Zhong (2018) proposed an approach to extract code samples

from Stack Overflow, and mined repair patterns from extracted code samples. Wan et al. (2021) used Stack Overflow to understand the challenges and needs amongst blockchain developers by applying Balanced LDA. Bangash et al. (2019) studied the machine learning related posts and found that some machine learning topics are significantly more discussed than others, and others need more attention.

To brainstorm feature ideas, help new users get their bearings, and further improve collaboration, GitHub released Discussion in 2020. Hata et al. (2022) took a first look at the early adoption of GitHub Discussion from several aspects. For instance, they found that errors, unexpected behavior, and code reviews are prevalent discussion categories. Specifically, the perception from the developer survey pointed out that developers consider GitHub Discussions useful but face the problem of topic duplication between Discussions and Issues. Motivated by their survey insight, we conduct an empirical study on NPM repositories to understand how developers maintain these channels, so as to complement the knowledge gap and provide empirical suggestions for the practitioners on how to select the appropriate channel.

8.2 Developer Communication in Software Development

Developer communication plays a significant role in software development, such as code review process (Wang et al., 2021b). Bacchelli and Bird (2013) stated that developers have need of richer communication than comments annotating the changed code when reviewing. Pascarella et al. (2018) found that during reviews, reviewers often request additional information about correct understanding, alternative solution, to improve patch quality. Recent work cited that reviewers suffer from confusion due to a lack of information about the intention of a patch (Ebert et al., 2019). Wang et al. (2021c) observed that developers are likely to share links during review discussions with seven intentions to fulfill information needs. Meanwhile, Hirao et al. (2019) reported that the patch linkage (i.e., posting a patch link to another patch) is used to indicate patch dependency, competing solutions, or provide broader context. In addition to code reviews, nowadays interactive communication channels are available to support development. Stray and Moe (2020) conducted a longitudinal study on coordination to understand the use of meetings and Slack and found that collaboration tools increase awareness and informal communication. Parra et al. (2022) presented a comparative study of developer communications on Slack and Gitter. Raglianti et al. (2022) proposed a tool using Discord conversations to aid program comprehension. With the official release of GitHub Discussion, it would gradually become a popular centre for developers to communicate and share ideas during their software development. Hence, one potential benefit of our study is to improve the communication efficiency between GitHub Discussion and Issue.

8.3 Barriers for Newcomers in OSS Projects

Newcomers are significant to the survival, long-term success, and continuity of OSS projects (Kula and Robles, 2019). However, literature pointed out that newcomers face many challenges in their initiative activities in OSS projects (Steinmacher et al., 2014). For instance, Lee et al. (2017) reported that newcomers lack the necessary domain knowledge and programming skills. In addition, some non-technical also affect newcomers' onboarding process, such as communication and social interaction (Tan and Zhou, 2019; Rehman et al., 2022). In the recent work, Mendez et al. (2018) studied newcomer barriers and gender through a new perspective, i.e., the usage of OSS tools and infrastructure. To facilitate the newcomers' onboarding process, a series of theories and strategies have been proposed. Steinmacher et al. (2018) provided guidelines for both OSS communities interested in receiving more external contributions, and newcomers who want to contribute to OSS projects. Tan et al. (2020) discovered the criteria to identify good first issues (GFIs) that may make GFIs more likely to be solved by newcomers. In the following study, Xiao et al. (2022) proposed RECGFI, an effective practical approach for the recommendation of good first issues to newcomers. With the introduction of a new feature (i.e., GitHub Discussion), the newcomers may also face the barrier of choosing proper communication channels when asking questions or raising issues. Especially for those uncertain problems (i.e., whether they are real issues or not), our results suggest that it would be appropriate for newcomers to start from the Discussion channel.

9 Conclusion

With the adoption of the GitHub Discussion feature, it becomes challenging for developers to appropriately choose or maintain between GitHub Discussion and Issue. In this work, we conducted an empirical study on 259 NPM and 148 PyPI repositories to understand the reasons behind converting Discussion to Issue and vice versa, and to investigate whether or not the conversion requires additional effort. Our empirical results show that reporting a clarification request is the most common reason of converting discussions to issues (35.1% and 34.7%, respectively), while having non actionable topic is the most frequent reason of converting issues to discussions (55.0% and 42.0%, respectively). Moreover, we observe that it potentially takes time to raise a conversion intent. This study contributes to helping developers effectively utilize these different communication channels, and also provides the future direction on the automatic classifier needs such as duplication detection between GitHub Discussion and Issue, and identification of non actionable topics from Issue.

Acknowledgement

Declarations

Conflict of Interest

The authors declare that Raula Gaikovina Kula and Yasutaka Kamei are members of the EMSE Editorial Board. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report.

Data Availability Statements

The datasets generated during and/or analysed during the current study are available in the `https://github.com/posl/GitHub_Discussion_Conversion`.

References

Abdalkareem R, Nourry O, Wehaibi S, Mujahid S, Shihab E (2017) Why do developers use trivial packages? an empirical case study on npm. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ACM, ESEC/FSE 2017, p 385–395

Bacchelli A, Bird C (2013) Expectations, Outcomes, and Challenges of Modern Code Review. In: Proceedings of the 35th International Conference on Software Engineering, pp 712–721

Bangash AA, Sahar H, Chowdhury S, Wong AW, Hindle A, Ali K (2019) What do developers know about machine learning: a study of ml discussions on stackoverflow. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), IEEE, pp 260–264

Chinthanet B, Kula RG, McIntosh S, Ishio T, Ihara A, Matsumoto K (2021) Lags in the release, adoption, and propagation of npm vulnerability fixes. Empirical Software Engineering pp 1–28

Chouchen M, Ouni A, Kula RG, Wang D, Thongtanunam P, Mkaouer MW, Matsumoto K (2021) Anti-patterns in modern code review: Symptoms and prevalence. In: 2021 IEEE international conference on software analysis, evolution and reengineering (SANER), IEEE, pp 531–535

Cogo FR, Oliva GA, Hassan AE (2019) An empirical study of dependency downgrades in the npm ecosystem. IEEE Transactions on Software Engineering pp 1–1

Decan A, Mens T, Claes M (2016) On the topology of package dependency networks: A comparison of three programming language ecosystems. In: Proccedings of the 10th European Conference on Software Architecture Workshops, pp 1–4

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

Hata H, Treude C, Kula RG, Ishio T (2019) 9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay. In: Proceedings of the 41st International Conference on Software Engineering, p 1211–1221

Hata H, Novielli N, Baltes S, Kula RG, Treude C (2022) Github discussions: An exploratory study of early adoption. Empir Softw Eng 27:3

Hecke TV (2012) Power study of anova versus kruskal-wallis test. Journal of Statistics and Management Systems 15(2-3):241–247

Hindle A, Alipour A, Stroulia E (2016) A contextual approach towards more accurate duplicate bug report detection and ranking. Empirical Software Engineering 21(2):368–410

Hirao T, McIntosh S, Ihara A, Matsumoto K (2019) The Review Linkage Graph for Code Review Analytics: A Recovery Approach and Empirical Study. In: Proc. of the International Symposium on the Foundations of Software Engineering (FSE), p 578–589

Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association pp 583–621

Kula RG, Robles G (2019) The Life and Death of Software Ecosystems, Springer, pp 97–105

Lee A, Carver JC, Bosu A (2017) Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: a survey. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), IEEE, pp 187–197

Li Z, Yin G, Yu Y, Wang T, Wang H (2017) Detecting duplicate pull-requests in github. In: Proceedings of the 9th Asia-Pacific Symposium on Internetware, pp 1–6

Lima M, Steinmacher I, Ford D, Liu E, Vorreuter G, Conte T, Gadelha B (2022) Looking for related discussions on github discussions. arXiv preprint arXiv:220611971

Liu X, Zhong H (2018) Mining stackoverflow for program repair. In: 2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER), IEEE, pp 118–129

Mann HB, Whitney DR (1947) The Annals of Mathematical Statistics 18:50–60

McHugh ML (2012) Interrater reliability: the kappa statistic. Biochemia medica 22(3):276–282

Mendez C, Padala HS, Steine-Hanson Z, Hilderbrand C, Horvath A, Hill C, Simpson L, Patil N, Sarma A, Burnett M (2018) Open source barriers to entry, revisited: A sociotechnical perspective. In: Proceedings of the 40th

International conference on software engineering, pp 1004–1015

Nguyen AT, Nguyen TT, Nguyen TN, Lo D, Sun C (2012) Duplicate bug report detection with a combination of information retrieval and topic modeling. In: 2012 Proceedings of the 27th IEEE/ACM international conference on automated software engineering, IEEE, pp 70–79

Parra E, Alahmadi M, Ellis A, Haiduc S (2022) A comparative study and analysis of developer communications on slack and gitter. Empirical Software Engineering 27(2):1–33

Pascarella L, Spadini D, Palomba F, Bruntink M, Bacchelli A (2018) Information Needs in Contemporary Code Review. Proceedings of the ACM Conference on Computer Supported Cooperative Work 2:135:1–135:27

Rehman I, Wang D, Kula RG, Ishio T, Matsumoto K (2022) Newcomer oss-candidates: Characterizing contributions of novice developers to github. Empirical Software Engineering 27(5):1–20

Steinmacher I, Gerosa MA, Redmiles D (2014) Attracting, onboarding, and retaining newcomer developers in open source software projects. In: CSCW

Steinmacher I, Treude C, Gerosa MA (2018) Let me in: Guidelines for the successful onboarding of newcomers to open source projects. IEEE Software 36(4):41–49

Stemler S (2000) An overview of content analysis. Practical assessment, research, and evaluation 7(1):17

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

Stray V, Moe NB (2020) Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and slack. Journal of Systems and Software 170:110717

Tan X, Zhou M (2019) How to communicate when submitting patches: An empirical study of the linux kernel. Proceedings of the ACM on Human-Computer Interaction 3(CSCW):1–26

Tan X, Zhou M, Sun Z (2020) A First Look at Good First Issues on GitHub, Association for Computing Machinery, New York, NY, USA, p 398–409. URL https://doi.org/10.1145/3368089.3409746

Tantisuwankul J, Nugroho YS, Kula RG, Hata H, Rungsawang A, Leelaprute P, Matsumoto K (2019) A topological analysis of communication channels for knowledge sharing in contemporary github projects. Journal of Systems and Software 158:110416

Treude C, Robillard MP (2017) Understanding stack overflow code fragments. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, pp 509–513

Treude C, Barzilay O, Storey MA (2011) How do programmers ask and answer questions on the web? (nier track). In: Proceedings of the 33rd International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, ICSE '11, p 804–807

Vale G, Schmid A, Santos AR, De Almeida ES, Apel S (2020) On the relation between github communication activity and merge conflicts. Empirical Software Engineering 25(1):402–433

Vasilescu B, Capiluppi A, Serebrenik A (2012) Gender, representation and online participation: A quantitative study of stackoverflow. In: 2012 International Conference on Social Informatics, IEEE, pp 332–338

Wan Z, Xia X, Hassan AE (2021) What do programmers discuss about blockchain? a case study on the use of balanced lda and the reference architecture of a domain to capture online discussions about blockchain platforms across stack exchange communities. IEEE Transactions on Software Engineering pp 1331–1349

Wang D, Kula RG, Ishio T, Matsumoto K (2021a) Automatic patch linkage detection in code review using textual content and file location features. Information and Software Technology 139:106637

Wang D, Ueda Y, Kula RG, Ishio T, Matsumoto K (2021b) Can we benchmark code review studies? a systematic mapping study of methodology, dataset, and metric. Journal of Systems and Software 180:111009

Wang D, Xiao T, Thongtanunam P, Kula RG, Matsumoto K (2021c) Understanding shared links and their intentions to meet information needs in modern code review. Empir Softw Eng 26(5):96

Wang D, Xiao T, Treude C, Kula RG, Hata H, Kamei Y (2023) Understanding the role of images on stack overflow. arXiv preprint arXiv:230315684

Wang Q, Xu B, Xia X, Wang T, Li S (2019) Duplicate pull request detection: When time matters. In: Proceedings of the 11th Asia-Pacific Symposium on Internetware, pp 1–10

Xiao W, He H, Xu W, Tan X, Dong J, Zhou M (2022) Recommending good first issues in github oss projects. In: 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), IEEE, pp 1830–1842