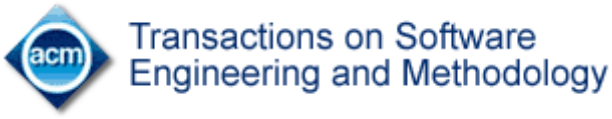

An Empirical Analysis of Issue Templates Usage in Large-Scale Projects on GitHub

Journal:	*Transactions on Software Engineering and Methodology*
Manuscript ID	TOSEM-2023-0169.R1
Manuscript Type:	Paper
Date Submitted by the Author:	15-Oct-2023
Complete List of Authors:	Sülün, Emre; Bilkent University, Computer Engineering Saçakçı, Metehan; Bilkent University, Computer Engineering Tüzün, Eray; Bilkent University, Computer Engineering
Computing Classification Systems:	Software and its engineering, Software development process management, Open source model

SCHOLARONE™
Manuscripts

An Empirical Analysis of Issue Templates Usage in Large-Scale Projects on GitHub

EMRE SÜLÜN, Bilkent University, Turkey
METEHAN SAÇAKÇI, Bilkent University, Turkey
ERAY TÜZÜN, Bilkent University, Turkey

GitHub Issues is a widely used issue tracking tool in open-source software projects. Originally designed with broad flexibility, its lack of standardization led to incomplete issue reports, impeding software development and maintenance efficiency. To counteract this, GitHub introduced issue templates in 2016, which rapidly became popular. Our study assesses the current use and evolution of these templates in large-scale projects, and their impact on issue tracking metrics, including resolution time, number of reopens, and comments. Employing a comprehensive analysis of 350 templates and their past versions from 100 large-scale open-source projects, we also evaluated over 1.9 million issues for template conformity and impact. Additionally, we solicited insights from open-source software maintainers through a survey. Our findings highlight issue templates' extensive usage in 99 of the 100 surveyed large-scale projects, with a growing preference for issue forms, a more structured template variant. Projects with a template exhibited markedly reduced resolution time (374.56 days to 103.46 days) and reduced comment count (4.90 to 4.34) compared to those without. The use of issue forms further significantly decreased resolution time, the number of reopenings, and the discussion extent. Thus, our research underscores issue templates' positive impact on large-scale projects, offering recommendations for improved effectiveness.

CCS Concepts: • **Software and its engineering → Software development process management**; **Open source model**.

Additional Key Words and Phrases: issue templates, issue forms, issue tracking, GitHub issues, bug tracking, empirical study

ACM Reference Format:
Emre Sülün, Metehan Saçakçı, and Eray Tüzün. 2023. An Empirical Analysis of Issue Templates Usage in Large-Scale Projects on GitHub. 1, 1 (October 2023), 27 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

GitHub hosts millions of projects, making it the most popular repository hosting service for open-source software projects. In addition to Git hosting, GitHub offers several other features, such as issue tracking, pull requests for code review, continuous integration, and package management. In GitHub, issue tracking is done with GitHub Issues, a built-in feature of GitHub.

Issue is a generic term that refers to any kind of task. For example, an issue can be a bug report, a feature request, a question, etc. Issue tracking, which is the practice of managing and maintaining lists of issues throughout the lifecycle of a project, has been a widely accepted and applied practice in software development. Earlier and more traditional issue tracking systems, such as Bugzilla and Jira, can be categorized as structured issue tracking systems. These systems have the ability to define extra fields for different purposes and provide customizable workflows by introducing validation

Authors' addresses: Emre Sülün, emre.sulun@bilkent.edu.tr, Bilkent University, Ankara, Turkey; Metehan Saçakçı, metehan.sacakci@ug.bilkent.edu.tr, Bilkent University, Ankara, Turkey; Eray Tüzün, eray.tuzun@cs.bilkent.edu.tr, Bilkent University, Ankara, Turkey.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Association for Computing Machinery.
Manuscript submitted to ACM

For Peer Review

rules and custom issue states. Unlike structured issue tracking systems, GitHub Issues offers very few fields for an issue. In the initial versions of GitHub Issues, the reporter of an issue was only required to provide a short description and a title. In GitHub Issues, project maintainers are unable to add new fields or create rules and workflows for issue tracking. To address a few of these shortcomings, GitHub Issues introduced the labeling feature. However, it was scarcely used [12].

Although the unstructured notion of GitHub Issue provides a flexible and faster way to create issues, in the long run, it hinders maintaining the issue tracking system [12]. This is especially true for large projects where dozens of issues are created daily. The lack of a strict structure in issue tracking may increase the number of incomplete issue reports. For example, a bug type of issue should include multiple types of crucial information, such as Steps to Reproduce or Environment information. For faster issue triaging and resolution, these crucial elements should be provided by the person who opens the issue [22, 25]. However, GitHub was suffering from the absence of these features because of its lightweight design, and the maintainers of GitHub repositories complained about that. Indeed, more than 2000 open-source maintainers wrote a letter titled "Dear GitHub" that explains the existing problems with the issue tracking feature of GitHub in 2016 [18]. One of the problems was expressed as follows; "Issues are often filed missing crucial information like reproduction steps or tested version." In the end, the maintainers proposed adding a template for bug reports. "We'd like issues to gain custom fields, along with a mechanism (such as a mandatory issue template, perhaps powered by a *newissue.md* in root as a likely-simple solution) for ensuring they are filled out in every issue."

To address the problems mentioned earlier, GitHub introduced the issue templates in 2016, aiming to improve the issue handling process [1]. The main aim of this study is to understand the impact of GitHub issue templates and gain insights into their usage by inspecting large-scale projects. By "large-scale," we refer to GitHub projects that have a substantial number of issues, stars, and pull requests, as well as projects that operate under valid licenses. We present the following research questions and subquestions based on this aim.

RQ1 What is the current status of issue templates on large-scale projects?
  RQ1.1 What is the ratio of projects using issue templates?
  RQ1.2 Which fields are most frequently used in a template?
RQ2 How does issue template usage evolve over time?
RQ3 Do contributors conform to the templates?
RQ4 How do issue templates affect issue tracking metrics (time to resolution, number of reopening events, number of comments)?
RQ5 How is the perception of issue templates among project maintainers?

In summary, this study examines the effectiveness of structured templates in issue tracking systems, with a specialized focus on software development projects on GitHub Issues, which is highly popular in software industry. Our investigation involves a pragmatic comparison of GitHub Issues managed with and without these structured templates, intending to unravel clear, empirical evidence of their overall impact. The exploration encompasses various dimensions such as the current utilization status of issue templates, their evolutionary trajectory, and a quantitative assessment of their influence on pivotal metrics like time to resolution, frequency of issue reopening, and the number of comments generated. Complementing our analytical findings, we also incorporate insights gleaned from surveys conducted with project maintainers, enriching the depth and breadth of our conclusions. The findings from this research will offer valuable insights, helping software developers make informed decisions to improve issue tracking processes. The rest of the paper is organized into the following sections. In Section 2, we provide a brief overview of issue templates in

For Peer Review

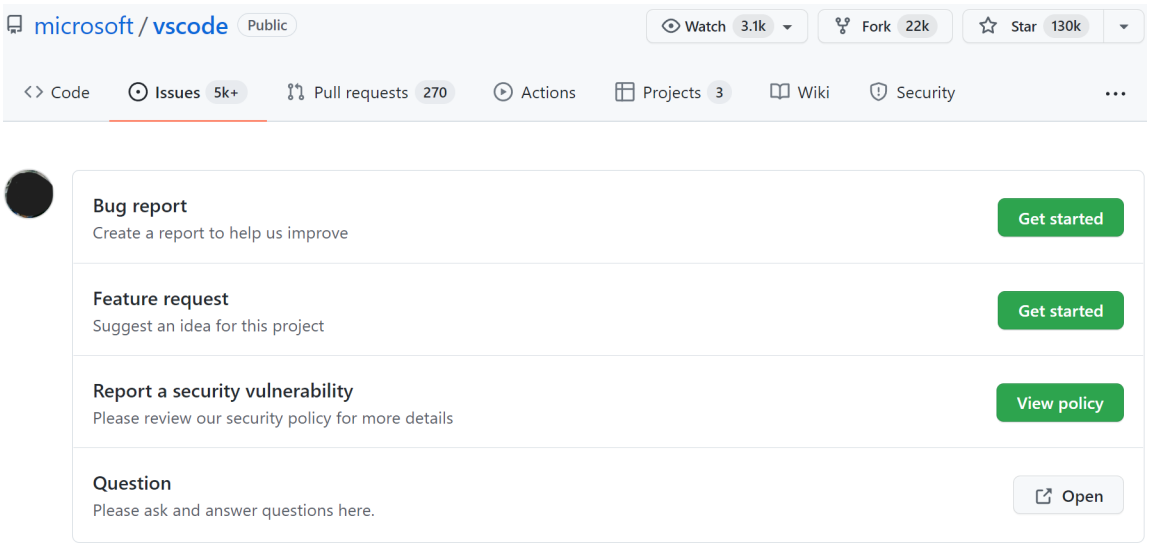

Fig. 1. Issue template chooser menu of Microsoft VS Code

GitHub. In Section 3, we present the related work. In Section 4, we describe how we conduct our study. In Section 5, we present our results and discuss them in Section 6. In Section 7, we discuss the threats to the validity of our study. Finally, in Section 8, we present our concluding remarks and recommendations.

2 BACKGROUND ON GITHUB ISSUE TEMPLATES

An issue template is a pre-defined set of fields with descriptions that can be used to create an issue. With issue templates, project maintainers can define different types of issues. For example, the Microsoft VS Code project uses two issue templates[1]: bug report and feature request. The template chooser menu is shown in Figure 1. The template guides users to provide various important details such as software version, operating system, and reproduction steps. The template content is shown in Figure 2. Besides the templates, project maintainers restrict users from creating issues for questions and security problems, and they prefer forwarding the users to external pages.

The details of the issue templates and how to use them are described in GitHub documentation [8]. Some of the features are listed below.

- A project can have multiple issue templates.
- Templates can be created with the template builder or issue forms (currently in beta).
- Templates are saved under *.github/ISSUE_TEMPLATE* directory and they are tracked in the version control system. There are two extra cases where a single template is saved with the name *ISSUE_TEMPLATE.md* under the project directory or multiple templates are saved under a repository which is named as *.github*.
- Templates are either written in Markdown or YAML formats.
- Templates can be used to add assignees and labels automatically. A default issue title can also be added with templates.

[1] https://github.com/microsoft/vscode/issues/new/choose

For Peer Review

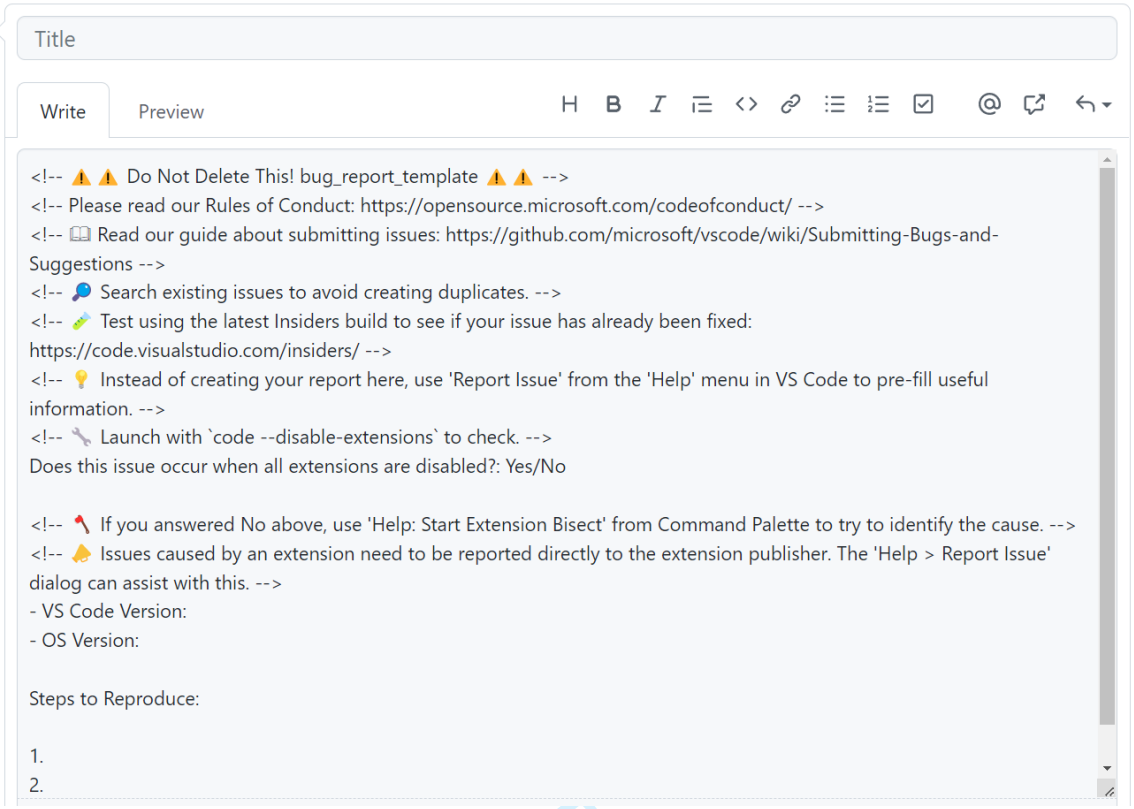

Fig. 2. Bug reporting issue template of Microsoft VS Code

- Templates created with issue forms (YAML format) can have various input types, such as open-ended text inputs, dropdowns, and checkboxes. For example, "a text area for providing the user's operating system, a dropdown menu for choosing the software version the user is running, a checkbox to acknowledge the Code of Conduct, and Markdown that thanks the user for completing the form."[2]
- The template chooser menu can be customized so that users can be forced to select a template or allowed to continue with a blank issue. Similarly, users can be redirected to external sites for non-issue type submissions.

The current state of issue templates is not the same as when they were first introduced. When introduced in 2016, a repository could have only one issue template [1]. Then in 2018, GitHub introduced multiple issue templates [4]. GitHub continued to improve issue templates by adding more features, such as the chooser menu[3], automated labeling, and assigning[4], organization-level templates[5]. Finally, in 2021, GitHub announced the issue forms, which is currently in beta for public repositories only. Issue forms are written in YAML, while the previous templates (vanilla templates) are in Markdown. For the rest of the paper, we use the term issue templates for both vanilla templates and issue forms.

[2] https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/syntax-for-githubs-form-schema#about-githubs-form-schema
[3] https://github.blog/changelog/2018-05-02-mutiple-template-choice/
[4] https://github.blog/changelog/2018-12-05-issue-template-automation-improvements/
[5] https://github.blog/changelog/2019-02-21-organization-wide-community-health-files/

Review

Since YAML is a data serialization language, unlike Markdown, a markup language, issue forms are more structured and organized than the previous templates. Furthermore, issue forms include novel features such as dropdown menus, checkboxes, and validations.

3 RELATED WORK

3.1 Issue Tracking Systems and GitHub Issues

Researchers frequently utilize GitHub as a data source for their studies due to its popularity and easy to use API for data retrieval [11]. GitHub Issues is also used to study issue tracking systems as a built-in feature of GitHub. For example, Kallis et al. [13] developed a GitHub app to predict the issue types automatically. Panichella et al. [19] conducted a similar study to identify *won't fix* issues on GitHub. Likewise, Izadi et al. [10] proposed a method to predict the objective and priority of the issues using feature engineering methods and text classifiers.

Chen et al. [3] discussed that issue titles and descriptions should be high quality. They developed a method to generate titles from the issue descriptions automatically.

Sasso et al. [21] identify a minimal bug report template that only includes the significant components. In addition, they discuss the effectiveness of different defect reporting models by comparing various tools such as Jira, Bugzilla, and GitHub Issues. They claim that tools with simpler interfaces and flexible structures will be more prevalent in the future. A similar study was conducted by Zimmermann et al. [25]. The study includes an analysis of 289 bug reports and a survey with 466 developers to identify the essential components of a bug report. The identified essential components are steps to reproduce and stack traces.

Another related research by Soltani et al. [22] analyzes the elements of bug reports to understand which parts are more significant. Their interviews with developers show that developers find descriptions, reproducing steps, test cases, and stack traces more critical than other elements, such as software version. Furthermore, their empirical analysis of 250 GitHub repositories shows reproducing steps, stack traces, fix suggestions, and user contents statistically impact bug resolution times.

Also, Luijten et al. [15] investigate bug reports by focusing on how their bug resolution times change over time. Based on their observation, Issabayeva et al. [9] applied a similar approach on three different software projects to claim software maintenance quality's positive effect on the issue handling process.

Chaparro et al. [2] built a tool to automatically identify the reproducing steps and then assess their quality. External evaluators assessed the tool and it can identify the steps to reproduce with 98% accuracy.

Similarly, Song and Chaparro [23] developed a tool to extract specific bug report elements. The tool, *BEE*, can detect the type of an issue: whether a bug, enhancement, or question. If the type is a bug, the tool can identify the observed behavior, expected behavior, and the steps to reproduce by analyzing the natural language text.

Furthermore, Rocha et al. [20] approach the 13,564 bugs reported on Mozilla Firefox in 2015 to extract information about bug workflow characteristics. For example, some of the found characteristics are the following: not assigned bugs require more than 70 days to be closed. 94% of duplicated bugs tend to be closed within two days. Skilled developers provide lower bug resolution time than less skilled ones.

In conclusion, many studies focus on the content of bug reports and which components a bug report should include. On the other hand, other types of issues, such as feature requests and questions, were not thoroughly studied.

For Peer Review

3.2 Template Usage in Software Development

Similar to issue templates, a pull request templates feature was introduced by GitHub in 2016 to increase the quality of pull requests. A recent study by Zhang et al. [24] empirically investigated the use of pull request templates. They found that using pull request templates positively impacts the maintainability of an open-source project, such as less time for code review, fewer duplicated pull requests, and invalid comments. They also found that the templates are used by only 1.2% of the repositories, and the majority of them are prevalent open-source projects.

Another recent study by Li et al. [14] used mixed methods to empirically analyze the issue and pull request templates and it aims to explore contents, impacts, and perceptions. The study reveals that templates contain elements such as greetings, project guidelines, and relevant information collection. After template adoption, there is a decrease in the monthly volume of incoming issues and pull requests, along with reduced discussion comments for issues and longer resolution durations. Similar to our findings, contributors, and maintainers rate the usefulness of templates positively but also report challenges in their use and suggest potential improvements for better user interaction and automation.

However, their study does not differentiate between different template types or analyze the historical change. In this study, we focus on the issue templates only, and we try to understand the historical evolution and the effect of different template types.

Coelho et al. [5] examined the maintenance levels of open-source projects to understand their inactivity. One of the research questions aims to compare the adoption of open-source best practices between active and inactive projects. Using issue and pull request templates is one of the best practices. Their analysis concluded that the difference is negligible between the active and inactive projects in terms of the use of issue templates.

Templates in requirements engineering were studied by Mohanani et al. [17]. Their results suggest that using templated requirements specification inhibits creativity by reducing critical evaluation and critical thinking.

4 METHODOLOGY

In this section, we describe how we conducted our study. The methodology can be summarized in Figure 3. In the figure, white color is used to indicate the data mining process, blue color is used to indicate the analyzing process, and finally, green color is used to indicate documenting the final result for research questions.

4.1 Data Collection

GitHub hosts more than 200 million repositories. Instead of analyzing a large number of projects in a shallow manner, we decided to select a subset of these projects and provide a more detailed analysis. Therefore, we sampled 100 repositories with the help of the GitHub Search Tool by Dabic et al. [6].

We used the following filters for sampling:

- Minimum 10,000 issues: To analyze more issues from a project
- Minimum 1,500 stars: To analyze popular and well-known projects
- Has open issues and pull requests: To analyze active projects
- Has a license: To analyze the projects with minimum standards
- Exclude forks: To avoid duplicates

The filters led to 101 repositories. However, we removed one repository[6] because our manual analysis showed that it is not a software project but a repository for hosting coding questions. We cloned the remaining 100 repositories to get

[6] https://github.com/type-challenges/type-challenges

For Peer Review

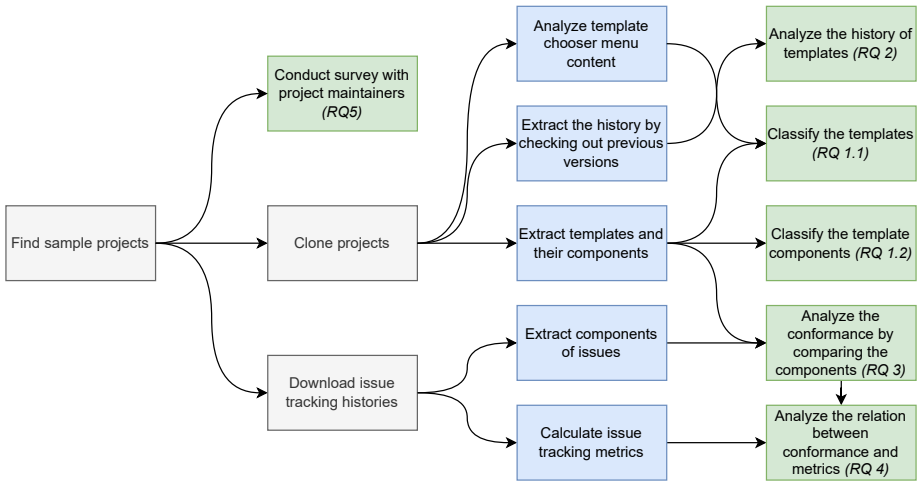

Fig. 3. The overview of the study

the content and the change history of issue templates. Cloned projects include some well-known open source projects such as Swift, React.js, Visual Studio Code, Pandas, and TensorFlow. The full list is available in our replication package [7].

We collected all the issues of these projects with the help of the Perceval tool [7]. The data collection step led to 100 repositories, 1,916,057 issues, and 350 issue templates.

4.2 RQ1: Analysis of Issue Templates

After cloning the repositories, we checked the Markdown and YAML files in the *.github/ISSUE_TEMPLATE* directory of the projects. Each file in this directory corresponds to a template. Historically, a template file can also be located in the project's root directory or *.github* directory. We inspected these directories as well.

We developed two parsers to extract the template information from the files. One of them parses the content of the templates written in Markdown language, while the other does the same for the YAML language.

A template file consists of two sections: the header and the body. The header includes metadata such as the template name and the template description. The body includes the template content, such as the software version and reproduction steps. In the rest of the paper, we will refer to each element (e.g., reproduction steps) in the content as *template component*.

We analyze the issue templates from four perspectives. First, we categorize templates by their use cases to understand the main motivation behind the templates. Then, we parse the template content and identify the components to understand how templates are utilized. In addition to the templates, we also analyze the content of the template chooser menu and classify its use cases to understand what kind of requests to open-source projects are not suitable for an issue tracking system. Finally, we analyze the conformance of issues made by contributors to understand the adoption of the templates.

[7] https://doi.org/10.6084/m9.figshare.21587088

For Peer Review

4.2.1 Template Category. A template can be classified into multiple categories, such as bugs or feature requests. In establishing these categories, we primarily drew inspiration from the predefined issue template categories of Jira [8] and GitHub [9]. By default, GitHub allows the creation of issue templates for "Bug" and "Feature Request," while Jira includes the "Task" category. Drawing on these defaults, we structured our categories accordingly.

Additionally, through the analysis of issues, it became evident that posing project-related questions is a prevalent practice, prompting us to incorporate the "Question" category.

To determine those categories, the 350 templates are independently and manually labeled by two authors of the study. After the independent categorization by each author, a collaborative review was undertaken by three authors to reconcile disparities and finalize the categories through consensus, solidifying the framework for our analysis.

Finally, we established four categories: *Bug*, *Feature Request*, *Task*, and *Question*. In conclusion,

(1) Bug: An issue type that reports unexpected or unintended behavior in a software project.
(2) Feature Request: An issue type used to suggest new functionality or improvements to existing features in an open-source project.
(3) Task: An issue type representing a specific action or piece of work that needs to be completed within the project, often related to maintenance or development.
(4) Question: An issue type for seeking clarification, assistance, or information related to the project or its usage.

A template is assigned to the *Other* category if no category is identified. Some other category templates would include library change, icon request, brand request, etc.

4.2.2 Template Content and Components. Issue templates typically consist of several questions that need to be answered by the user who opens the issue. Besides the questions, templates may also include informational texts to guide users. Project maintainers can configure the template content and components based on their needs.

In this part of our analysis, we investigate the typical components (fields, elements) of issue templates. To do this, we parsed the template content and extracted the headings. Then, we classified them based on keywords.

The parsers we developed work based on the following rules: Headings start with # character for the templates written in Markdown language. In issue forms, which are written in YAML language, the structure is more strict, and the heading of each component is located in the *label* section of template *attributes*.

We extracted 1,458 headings from 350 templates in total. To classify components, two authors of the study independently went over the 1,458 headings and classified them. During their manual analysis, two of the authors listed the keywords associated with each component category. In a separate meeting, three authors of the study went over the component categories and resolved conflicts after the discussion. The final list consisted of 11 component categories: *Actual Result*, *Additional Context*, *Alternative Solutions*, *Description*, *Environment*, *Expected Result*, *Logs*, *Motivation*, *Reproduction Steps*, and *Software Version*. Their descriptions can be found below:

(1) Actual Result: Detail the actual, flawed outcome caused by the bug to highlight discrepancies from the expected result.
(2) Additional Context: Include any supplementary information like screenshots or videos that could aid in understanding and resolving the bug more efficiently.

[8] https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-types/
[9] https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/configuring-issue-templates-for-your-repository

For Peer Review

(3) Alternative Solutions: Suggest temporary workarounds or solutions that could be applied while waiting for a complete fix.
(4) Description: Concisely outline the bug, providing a clear overview of the problem to set the stage for further details.
(5) Environment: Specify where the bug was identified, including details like operating system, browser, and device used.
(6) Expected Result: Clearly describe the intended outcome or behavior in the absence of the bug.
(7) Logs: Incorporate relevant error logs or messages that could shed light on the underlying issues causing the bug.
(8) Motivation: Express why it's essential to resolve this bug, outlining its impact and the necessity for its resolution.
(9) Reproduction Steps: Offer a clear, sequential guide, enabling others to replicate the issue accurately for debugging.
(10) Software Version: Indicate which version of the software the bug is occurring in, helping to identify if the issue is version-specific.

An example issue template divided to its components is shown in Figure 4. The screenshot is taken from the bug template of Salt project[10].

4.2.3 Template Chooser Menu. When creating a new issue, a template chooser menu is shown to the user (e.g., Figure 1). The menu has two use cases: showing the list of templates and the list of some external links to the user who wants to create a new issue. The project maintainers can configure the menu to redirect users to external links for non-issue type submissions. Also, the menu configuration includes an option to prevent users from creating a blank issue (an issue opened without a template).

The configuration is done by editing the *.github/ISSUE_TEMPLATE/config.yml* file. We developed a parser to extract the chooser content from the file. The parser identifies whether blank issues are allowed and finds the external links, if any.

For each external link, there are three headings in the configuration file: *name*, *url*, and *about*. The *about* heading describes what the external link is used for. As a part of the issue templates analysis, we classified the external links' use cases. We classified 164 external links by reading their *about* section. Two of the authors classified independently and then merged the results by discussing the different answers. Finally, the classification led to the following categories:

- *Questions/Discussion:* Project maintainers might not prefer getting the questions in the issue tracker. Instead, they redirect users who want to ask a question or discuss a problem to designated discussion pages such as GitHub Discussions or Stack Overflow.
- *Redirecting to Another Repo:* If a repository is a part of a larger project, project maintainers redirect users to the correct repository. For example, the Angular CLI repository has a link to the Angular Framework repository with the description "Issues and feature requests for Angular Framework."
- *Documentation:* A link to the documentation page is provided since it can be helpful for users who want to read the documentation before raising an issue.
- *Security Issues:* Project maintainers prefer getting the security issues over a private medium instead of the public issue tracker.
- *Paid Support:* Some open-source projects may have paid tiers and redirect users to the appropriate support page.

[10] https://github.com/saltstack/salt

For Peer Review

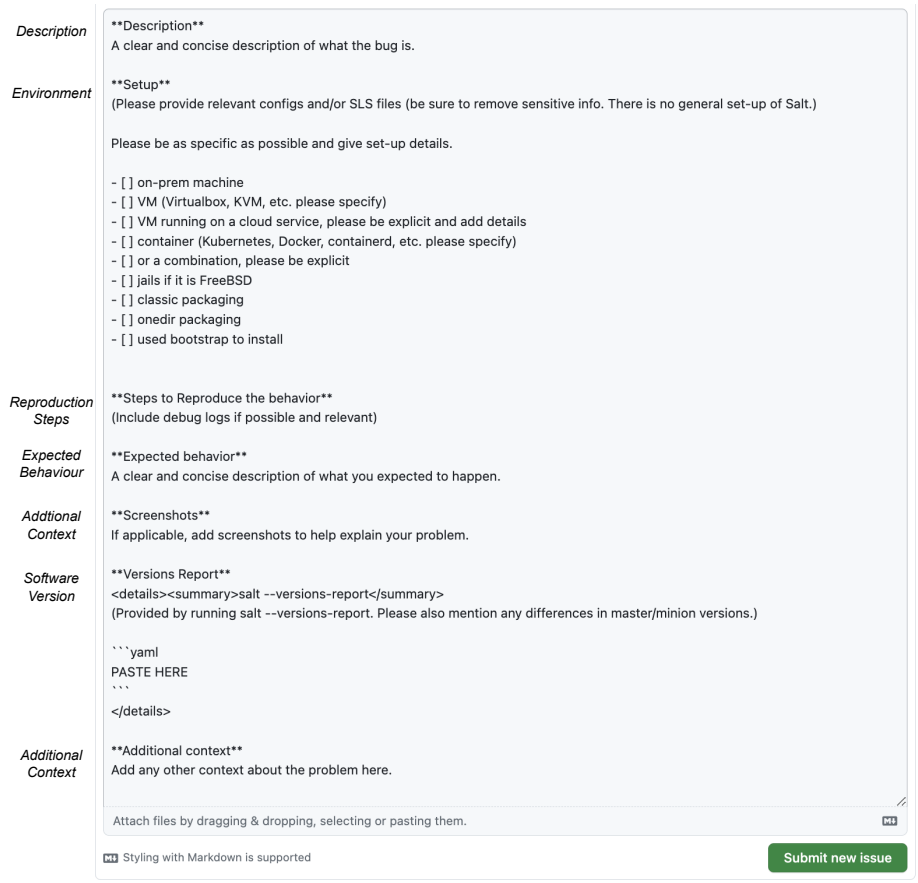

Fig. 4. An example bug template with its components

- *External Issue Tracker:* Some project maintainers prefer getting the issues related to a specific area by an external issue tracker instead of GitHub Issues.

4.3 RQ2: Analysis of Issue Templates' Histories

Issue templates are tracked on the Git version control system. Therefore, their histories can be analyzed by checking out the previous versions. We first run a Git command for the analysis to get the list of commits that modify the templates. Then, we checked out those commits and inspected the templates.

Historically, there are three major versions of issue templates, and their details were explained in Section 2. We refer to the initial version as *Old Markdown* where only one Markdown-based template was allowed. The second version allows users to create multiple Markdown-based templates, which we refer to as *Markdown*. Finally, we refer to the issue forms as *YAML* where the templates are more structured and written in the YAML language.

We analyze the transitions from the older versions to the newer versions. For this purpose, we take snapshots of the templates every three months to get quarterly updates between 01/01/2016 and 01/07/2022. Then, we inspect which version was used by a project at a given time.

4.4 RQ3: Conformance of Issues Made by Contributors

Although issue templates try to force users to answer required questions, issue reporters may not always provide legitimate answers when opening an issue. Since Markdown-based templates are actually editable free texts, questions in the templates can be deleted or modified. To parse this free text, we used a markdown parser called Marko [11]. By the help of this parser, we were able to parse the headings for each issue and for each markdown-based template.

On the other hand, the structure is more strict for YAML-based templates, and project maintainers can set validation rules to force users. Even in that case, people may skip some questions by filling in invalid answers such as *N/A*.

We analyze all 1,916,057 issues of 100 projects and try to understand whether the issues follow the templates. We measure a *conformance* score for each issue, and a score can be calculated only if the project had issue templates when the issue was opened. Template components can be extracted as described in Section 4.2.2.

The conformance measurement algorithm is given in Algorithm 1. It compares the expected title headings in a template with the actual titles in an issue. The conformance score is basically the ratio of correctly filled template components over the total number of components available for a template category. If an issue has a conformance score greater than zero, that means the corresponding issue template's at least one heading is used while generating the issue. We consider this case as the issue template used by the issue. Since we care about whether issues are created while considering an issue template or not, we are looking for the slightest clue that a portion of the issue template is used while creating an issue. We categorized the issues into ones with a conformance score of zero (issues that do not use issue templates) and ones with a conformance score greater than zero (issues that use issue templates). A component is correctly filled if its body is not empty and is not equal to the following strings: *No response*, *N/A*.

One should note that the issue templates can change over time. Therefore, when measuring the conformance of an issue, we consider the templates that existed when the issue was opened. Since there is no automated way of determining which issue template an issue is created from, we decided to find the originating template by calculating the conformance score per all available templates that were available at a given time and selecting the template with the highest conformance score.

4.5 RQ4: Analysis of Issues

After calculating the conformance of issues as described in Section 4.4, we investigate how the presence of templates, type of templates, and conformance scores affect the following issue tracking metrics:

- *Time to resolution*: Calculated as the time between the creation and the resolution. Measured in minutes.
- *Number of reopens*: Calculated as the number of times the issue was reopened.
- *Number of comments*: Calculated as the number of comments on the issue.

We hypothesize that issues created from a template are resolved faster, are reopened fewer times, and have more occasional comments than issues created without a template. The rationale behind the hypothesis is that when a template is not used, then the required information might not be provided by the reporter (less conformance), project maintainers may ask more questions to elaborate on the issue (more comments) and wait for the response (more time to resolution). Also, the issue may be reopened more times since it was not fully understood when reported.

A similar hypothesis holds for non-zero conformance issues vs. zero-conformance issues as well.

[11] https://pypi.org/project/marko/

For Peer Review

```
Input: templateComponents, issueComponents
if length(templateComponents) == 0 then
    return null;
end
matchCount = 0;
foreach component of templateComponents do
    foreach issueComponent of issueComponents do
        if (component.title in issueComponent.title or issueComponent.title in component.title) and issueComponent
          has a valid body then
            matchCount += 1;
            break;
        end
    end
end
conformance = matchCount / length(templateComponents);
return conformance
```

Algorithm 1: Conformance measurement algorithm

Furthermore, since YAML-based templates are more structured than Markdown-based templates, we hypothesize that issues created from a YAML-based template are resolved faster, are reopened fewer times, and have more occasional comments than issues created from a Markdown-based template.

We utilize the Mann-Whitney U Test [16] to test the difference in nine hypotheses statistically. The data is suitable for the Mann-Whitney U Test since it is continuous and skewed, and the samples are independent.

Before the testing, we first filter out the open issues as they do not have resolution time, and also, they can get reopened more and receive comments later. This filtration step is independent of the filtration step in the previous research question.

After the statistical test, we calculate the effect sizes for each hypothesis with Cohen's d.

4.6 RQ5: Perception of Project Maintainers

We conducted a survey to understand the perception of project maintainers about issue templates. To reach out to project maintainers, we mined the Git commit histories and identified the users who have contributed to the issue templates. Then, we sent an email to 763 identified users asking them to participate in the survey. Also, we offered a raffle for a $100 Amazon gift card to encourage participation.

Before sending the survey to the actual respondents, we first sent it to two software engineer colleagues for review. This step was taken to ensure that the survey was clear, easy to understand, and free of any biases or errors. They provided valuable feedback and suggestions, which helped us make necessary revisions and improvements to the survey before sending it out to the respondents.

Out of the 763 sent, 685 emails were successfully delivered, while the remaining were not delivered due to various reasons such as disabled email addresses of former employees. Finally, we collected responses from 39 project maintainers and analyzed the responses both quantitatively for Likert-scale questions and qualitatively for open-ended questions.

The survey questions are listed in Appendix A. Question 1 is a multiple-choice question, Questions 2-5 are Likert-scale, and Questions 6-8 are open-ended.

For Peer Review

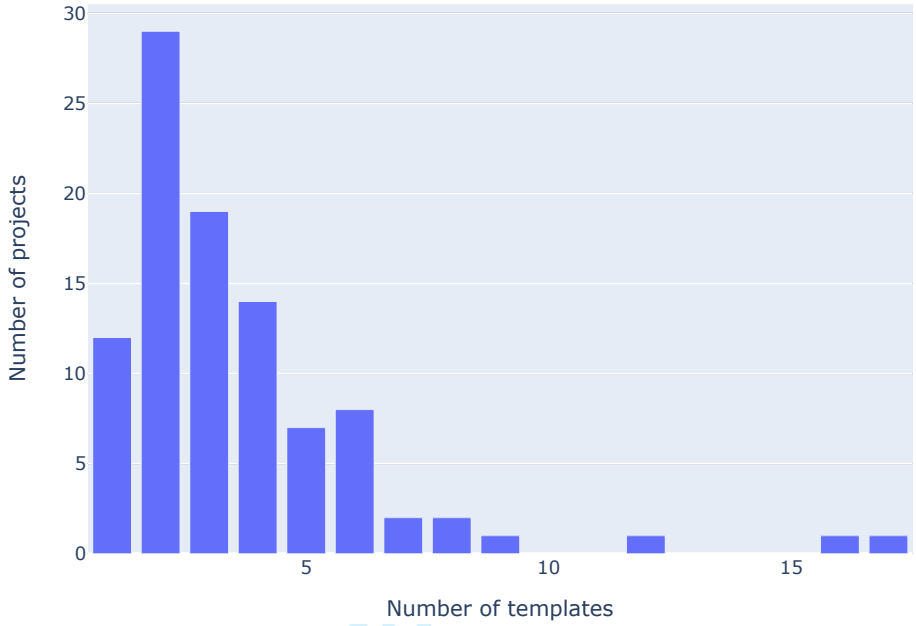

Fig. 5. The distribution of the number of templates per project

5 RESULTS

5.1 RQ1: The Current Status of Issue Templates

5.1.1 RQ1.1 Prevalence of Issue Templates. We analyzed 100 projects to find out the current status of issue templates. We found that 97 projects directly use the issue templates, 1 project[12] uses organization-level templates, 1 project[13] redirects users to an external issue creation form, and when this form is filled in, the user is redirected back to GitHub with a draft of the issue. In conclusion, only 1 project[14] does not have any issue templates. The rest of the projects have a varying number of templates ranging from 1 to 17 as presented in Figure 5.

> We found that only one out of 100 projects does not use issue templates. The most frequent template combination uses two templates (One for reporting bugs and one for suggesting new features).

A template can be created as a Markdown file or a YAML file. We found that 57% of templates are created as Markdown. The distribution of the template categories is shown in Figure 6.

Figure 7 shows whether the project maintainers allow users to create blank issues by setting the *blank issues enabled* option. Since this configuration is not mandatory, the chart has a *Not Specified* category. When not specified, the default behavior allows users to create blank issues.

The distribution of 164 external links in the template chooser menu is shown in Table 1. In this table, *Other* category represents the links for which we do not have a specific category. For example, we observed that some external links

[12] https://github.com/atom/atom
[13] https://github.com/vuetifyjs/vuetify
[14] https://github.com/sympy/sympy

Peer Review

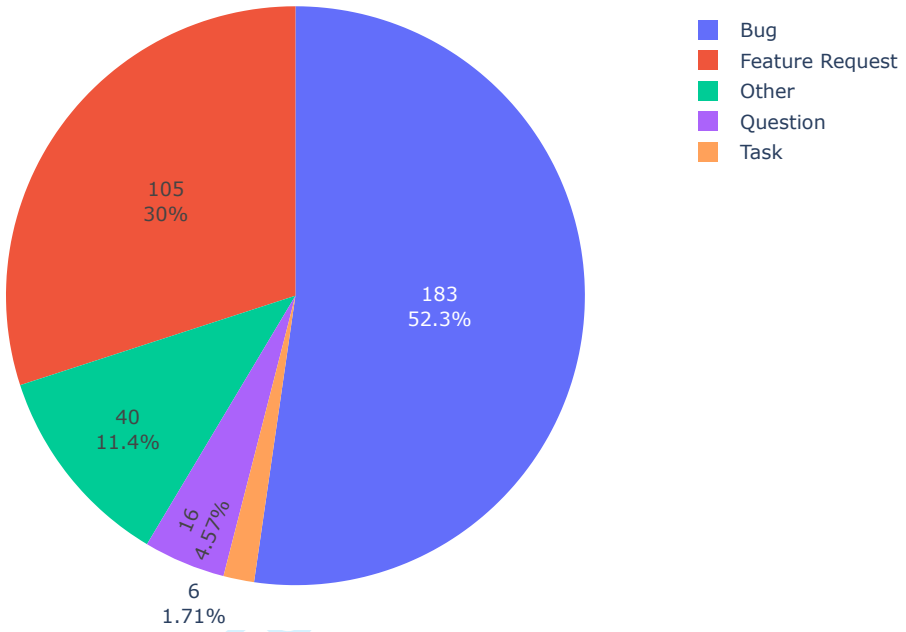

Fig. 6. The distribution of the template categories

Table 1. The distribution of external links in the template chooser menu

Category	Number of links
Questions/Discussion	71
Redirecting to Another Repo	49
Documentation	18
Security Issues	9
Paid Support	8
External Issue Tracker	4
Other	5
Total	164

are used to request donations, promote a product, and recruit developers. Because these cases are rare, we put them in the *Other* category.

5.1.2 RQ1.2 Template Components. The result of the analysis of the template content and the used components is shown in Table 2. In this table, template components are grouped by the template category. The unclassified components and templates are represented under the *Other* categories in both rows and columns. We have placed them in the *Other* category because the components are either

(1) Project specific: Some examples include “Are you using the Nextcloud Server Encryption module?” , “For PowerShell Team **ONLY**”
(2) Temporary fields: Some examples include “Final Release - On the 27th”, “After Release - Conclusion - 1 business day after the 27th”

Peer Review

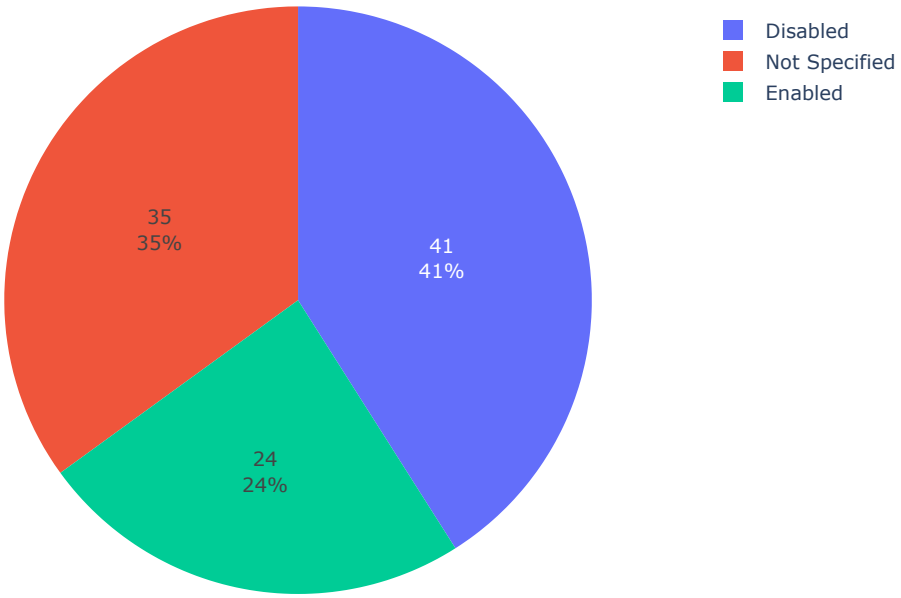

Fig. 7. Project's blank issue allowance distribution *(If not specified, it is enabled by default)*

Table 2. Components used in the templates

	Bug	Feature Request	Question	Task	Other
Actual Result	56	0	0	0	2
Additional Context	57	46	2	1	7
Alternative Solutions	11	28	0	2	0
Description	87	103	5	3	9
Environment	73	4	0	0	3
Expected Result	69	16	0	0	5
Information Text	97	30	4	1	8
Logs	33	0	2	0	2
Motivation	3	20	1	0	4
Reproduction Steps	92	0	0	0	5
Software Version	105	5	3	0	5
Other	255	96	20	7	71

(3) Cryptic and does not provide any help text to understand: Some examples include "Flag", "Command (mark with an x)".

The most common template components used in the issue templates are: Actual Result, Additional Context, Alternative Solutions, Description, Environment, Expected Result, Information Text, Logs, Motivation, Reproduction Steps, and Software Version.

Peer Review

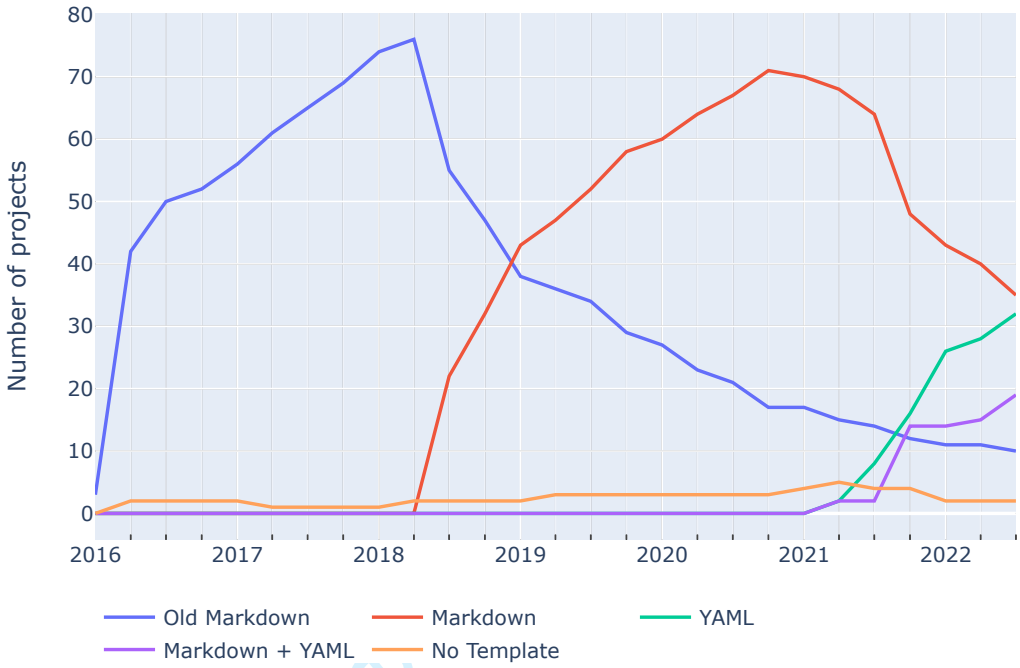

Fig. 8. The historical evolution of issue template usage

5.2 RQ2: The Evolution of Issue Templates

Since the introduction of issue templates in 2016, open-source maintainers have started to use this feature. Figure 8 shows how many projects use issue templates over time by the template type. Note that multiple versions (a project may have YAML-based templates and Markdown-based templates at the same time) can be used simultaneously.

The transition between different template types is also shown in Figure 9. The gray lines on the figure indicate those transitions. For example, it can be seen that a large number of projects stopped using "Old Markdown" templates and started to use "Markdown-based" templates. This also can be verified in Figure 8. Realize that when the number of "Old Markdown" templates started to drop, the number of "Markdown-based" templates started to increase. In general, Figure 9 gives insight into the number of transitions from one template type to another template type. Also, a trend of upgrading the template type is observed in this figure.

> Large-scale projects tend to use new generation issue templates over time. As Figure 8 and 9 suggest, more projects increasingly prefer issue forms that are more strict but still configurable.

5.3 RQ3: Conformance of Issues Made by Contributors

In conformance analysis, we found that 1,022,522 issues are not eligible for conformance analysis because there was no template at that time (554,331 issues) or we could not parse the issue to calculate the conformance (648,272 issues) due to usage of non-formal denotation for headings in Markdown-based templates (For more details see the Construct Validity). Among the eligible ones, we found that 33% of the issues have zero conformance to templates. The rest of the issues have varying conformance to templates ranging from 0.04 to 1. The mean conformance is 0.71, while the median

Peer Review

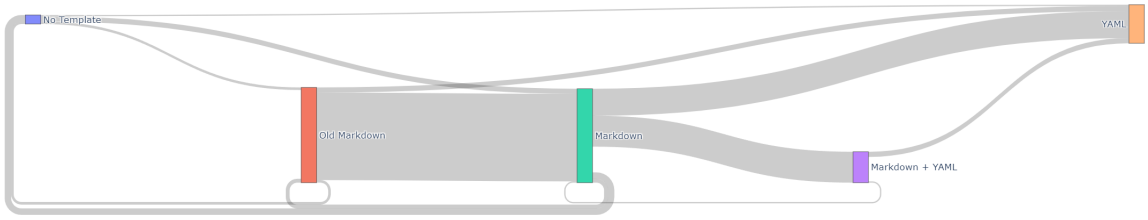

Fig. 9. Transition between different template types

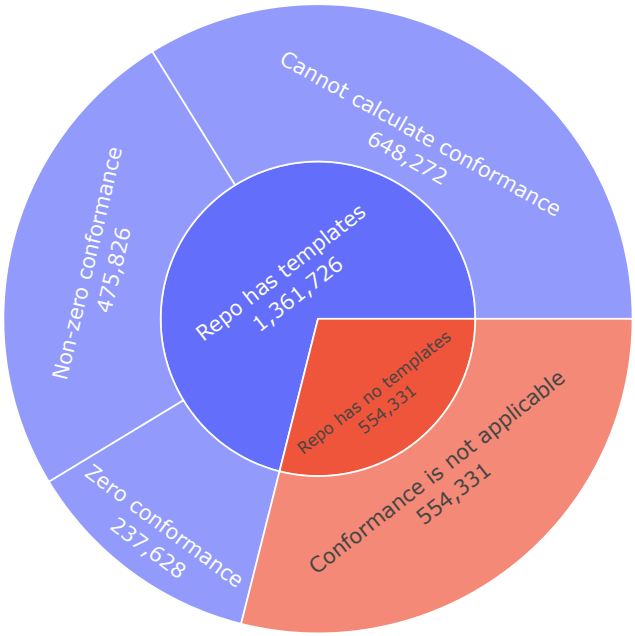

Fig. 10. Distribution of the issues. The inner circle shows whether projects use issue templates when an issue is created, while the outer circle shows the conformance category.

is 0.75. Figure 11 presents the conformance analysis results. A visual summary of the distribution of 1,916,057 issues is given in Figure 10.

5.4 RQ4: The Effect of Issue Templates

To understand the effect of using issue templates, we analyzed 1,916,057 issues as described in Section 4.5. The filter (closed state) led to 1,661,788 issues. 209,654 issues have zero conformance to templates, while 394,285 issues have non-zero conformance.

We have nine hypotheses for three metrics *(TTR, number of comments, number of reopens)* and three issues groups *(zero conformance and non-zero conformance, no template available and template available, Markdown templates and YAML templates)* to test the effect of issue templates overall. The p-values and effect sizes of the hypotheses are shown in Table 3, and the statistical summary of the comparison is given in Table 4.

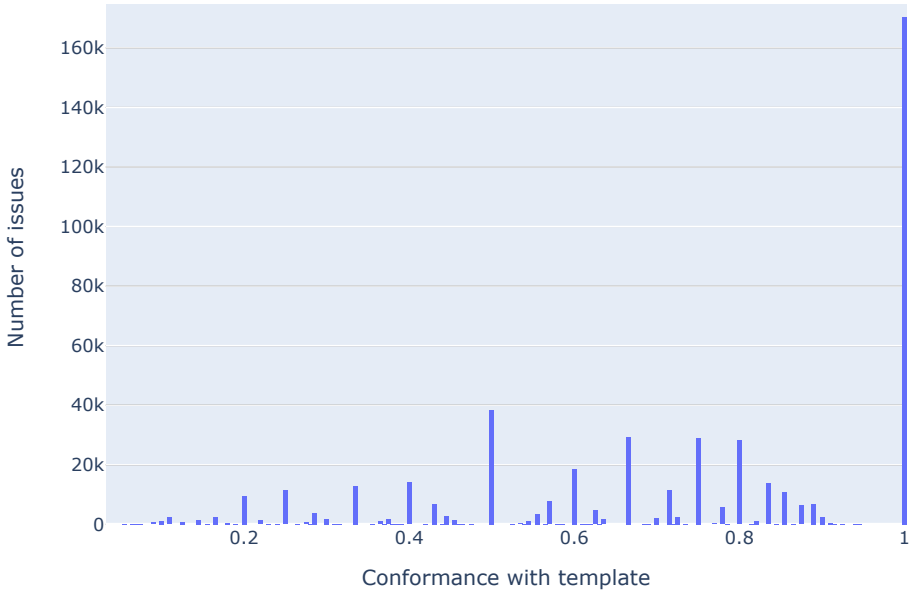

Fig. 11. The conformance of issues (excluding zero conformance)

In the comparison of template availability by the time-to-resolution metric, we found that *No Template Available* (374.56) has three times larger mean value than *Template available* (103.46), and the change in time-to-resolution metric is statistically significant as can be seen in Table 3 (p-value = 0.00). Also, the number of comments metric is statistically significant (p-value = 0.00) for the same comparison where *No Template Available* has a mean of 4.90 and *Template Available* has a mean of 4.34.

Similarly, in the comparison of the used template type by the time-to-resolution metric, we found that *Markdown-based Template* (81.76) has four times larger mean value than *YAML-based Template* (23.52), and the change in time-to-resolution metric is statistically significant as can be seen in Table 3 (p-value = 0.00). Also, the number of comments metric is statistically significant (p-value = 0.00) for the same comparison where *Markdown-based Template* has a mean of 4.37 and *YAML-based Template* has a mean of 3.71.

Mann-Whitney U test is used for statistical testing, and Cohen's d is used for measuring the effect size.

> Issues created when the project has an issue template have less time to resolution and fewer comments. The difference is statistically significant. However, the effect of the conformance on three metrics is insignificant. Also, when YAML-based templates are used, the time to resolution is significantly lower, and the number of comments and reopening events is less.

5.5 RQ5: Perception of Project Maintainers

The responses of project maintainers related to the benefits of issue templates are shown in Figure 12. The majority of the project maintainers agree that issue templates help to improve the quality of the issues in terms of less time to resolution, less discussion, fewer comments, and less likely to be reopened. On the other hand, they do not think that issue templates affect the likelihood of being duplicated.

Peer Review

Table 3. p-values and effect sizes of the hypotheses

Hypothesis	p-value	Cohen's d
Conformance to the templates		
Less time to resolution	1.00	0.03
Fewer reopens	0.44	0.01
Fewer comments	1.00	0.09
Whether the project has a template or not		
Less time to resolution	0.00	0.59
Fewer reopens	1.00	0.03
Fewer comments	0.00	0.07
YAML-based template over Markdown-based template		
Less time to resolution	0.00	0.37
Fewer reopens	0.00	0.06
Fewer comments	0.00	0.10

Table 4. Summary of the statistical analysis

		Mean	Median	STD
TTR (in days)	Zero Conformance	116.56	6.00	272.58
	Non-Zero Conformance	108.43	6.52	241.05
	No Template Available	374.56	23.14	746.01
	Template Available	103.46	6.22	238.89
	Markdown-based Template	81.76	6.39	170.77
	YAML-based Template	23.52	3.67	47.75
# of Comments	Zero Conformance	4.23	2.00	8.39
	Non-Zero Conformance	4.90	3.00	7.15
	No Template Available	4.90	3.00	9.36
	Template Available	4.34	3.00	7.17
	Markdown-based Template	4.37	3.00	6.61
	YAML-based Template	3.71	2.00	5.44
# of Reopens	Zero Conformance	0.05	0.00	0.32
	Non-Zero Conformance	0.05	0.00	0.25
	No Template Available	0.04	0.00	0.21
	Template Available	0.05	0.00	0.31
	Markdown-based Template	0.05	0.00	0.31
	YAML-based Template	0.03	0.00	0.23

For Peer Review

In detail,

- 79% of the respondents agree that issues created from a template require less time to resolution.
- 85% of the respondents agree that issues created from a template require less discussion and fewer comments.
- 33% of the respondents agree that issues created from a template are less likely to be reopened.
- 23% of the respondents agree that issues created from a template are less likely to be duplicated.

In addition to the Likert-scale questions, we asked the project maintainers to comment on the issue templates. One respondent summarizes the benefits as follows:

> The biggest, most practical benefit of issue templates is that they address a key problem of fully free-form: you often lack key pieces of information. If an issue template was ignored/not filled in by a reporter there is a high chance that it will be of low quality (i.e., missing key pieces of information) and tricky to bucket and resolve.

On the other hand, another project maintainer summarizes the challenges related to issue templates as follows:

> Templates can accrue a lot of cruft, be overly specific/rigid and worse-case scenario dissuade potential reporters from even creating an issue. A good template should be short and provide clear guidance about the kind of input it needs from a user.
> They are mostly challenging for existing contributors who know when they're already providing enough information. It's a bit of wasted time on their part when they have to fill in unneeded parts of the form. That's why it's important to be flexible: it should be possible to delete some parts of your comment or leave them blank.

When their issue tracking system preference is asked, 16 respondents (41%) stated they prefer GitHub Issues with YAML-based templates, while 9 respondents (23%) prefer Jira or another structured system, 6 respondents (15%) prefer GitHub Issues with Markdown templates, 4 respondents (10%) prefer GitHub Issues without any templates and the rest state other alternative solutions.

Fig. 12. Survey responses. 1: Strongly disagree 5: Strongly agree

6 DISCUSSION

In this section, we discuss the analysis of the results. We also provide a few suggestions for practitioners to improve issue tracking workflow and for researchers for future work.

Our analysis shows that issue templates are widely used (99% of the observed projects). Also, we observed the majority of the projects are using two templates (one for reporting bugs and one for requesting new features) with 29%.

We also observed that project maintainers tend to redirect users to external sites for Q&A (43%) and documentation (11%). The sites usually include a dedicated forum, Stack Overflow, or GitHub Discussions page. This way, the issue tracking system is only used to track bugs and feature requests.

There are many common components between the issue templates of different projects (see Table 2). For example, *Reproduction Steps* are usually needed when reporting a bug.

When GitHub introduced a new version of the Markdown issue templates in 2018, projects were using the older version of the Markdown issue templates. Since projects started using the new version instead of the old one, while the number of projects using the new version increased, the ones using the old version decreased.

Similarly, when GitHub introduced YAML issue templates in 2021, usage of Markdown issue templates started to decrease in the same manner. Since Markdown and YAML issue templates can be used in the same project, we observed increases in the usage of Markdown and YAML issue templates together in the same project. However, usage of only Markdown issue templates decreased while only YAML issue templates increased.

In terms of the effect of the templates, we observed that using templates reduces the time to resolution and the number of comments. Also, using YAML-based templates reduces the time to resolution and the number of comments and reopens.

6.1 Suggestions for Project Maintainers

6.1.1 Use issue templates. Large-scale projects can significantly benefit from structured issue-tracking workflows, and a key tool for this structure is issue templates. For example, consider a project where bugs are reported haphazardly without clear reproduction steps or system details. This leads to time-consuming clarifications and delays in resolution. Now, by implementing a structured "Bug" template that requires specific fields, this ambiguity is largely resolved, streamlining the fixing process. Thus, maintainers should have primary templates like one for reporting bugs and another for suggesting new features, as per the patterns in Table 2.

About choosing the right template type, issue forms—which are YAML-based templates—exhibit a higher degree of structure and organization compared to their Markdown-based counterparts. According to Table 4, YAML-based templates significantly reduce the Time to Resolution (TTR) when compared with Markdown-based templates. Therefore, for those seeking a stricter, more regimented workflow, YAML-based templates could be an advantageous choice.

6.1.2 Prevent users from creating issues without a template. If users are permitted to create issues from scratch, it renders the issue templates essentially redundant.

Imagine a scenario where ten issues are raised, but only three follow the template. This inconsistency can lead to critical bugs being missed or delayed in resolution because the issues lack clarity. Therefore, it is imperative that creating issues without templates should be discouraged, and if an issue does not fit any template, platforms like Q&A forums could be more appropriate.

For Peer Review

6.1.3 Thoroughly explain the components in your templates. When analyzing the content of different templates, we observed some templates with cryptic components. The cryptic components usually have a non-obvious title with no description available, which would allow different interpretations of the components by different developers. Writing more descriptive texts inside the templates can help issue reporters and increase the conformance scores.

For example, consider two components: one labeled "SysInfo" and another labeled "System Information (OS, RAM, CPU)". The latter, being more descriptive, leaves less room for ambiguity, speeding up resolution times. Vague components can lead to confusion. For instance, without clear instructions, two reporters might interpret "SysInfo" differently—one might list only the OS, while another might detail the entire hardware setup. Hence, templates should be explicit and detailed, aiding both issue reporters and developers.

Furthermore, project maintainers can reference Table 2 to decide which components should be added to their templates.

6.2 Suggestions for Project Contributors

When opening an issue, providing sufficient information is critical, and issue templates help contributors to ensure they provide sufficient information. Otherwise, project maintainers may request more information, which can cause a longer process (See Figure 13 for an example).

Therefore, we suggest the contributors conform to the templates as much as possible, especially for the required fields.

Opening an issue is more than just pointing out a problem; it's about communicating efficiently. Let's examine two contrasting situations:

Scenario A (Without Adhering to Templates): Bob, a contributor, raises an issue describing a software crash. However, he either skips the template or not providing the detailed information required by the template, and writes a generic description. This lack of detail forces the maintainers to seek more information: Which software version was she using? Under what conditions did the crash occur? Was it replicable? As a result, precious time is spent in back-and-forths (much like the extended discussion seen in Figure 13), delaying the resolution.

Scenario B (Properly using the Issue Templates): On the other hand, Alice encounters a similar crash. She opts to use the provided template, detailing the software version, the specific sequence leading to the crash, and other required fields. The maintainers immediately understand the crux of the problem and swiftly move to rectification. Given these examples, the benefits of adhering to issue templates become evident. Not only do they streamline the communication between contributors and maintainers, but they also expedite the problem-solving process. Thus, we strongly recommend contributors to meticulously fill out the templates, paying special attention to the required fields to ensure the most efficient and effective communication.

6.3 Suggestions for GitHub

During the analysis, we noticed that some templates written in YAML had syntax issues [15]. Unlike Markdown, YAML has a strict syntax, and syntax issues can cause problems such as miss renderings. In case of a syntax error, GitHub does not show any warning message, causing errors to be missed. A good example of the disadvantage here can be seen when a project uses a YAML template with a minor error: Instead of a proper render, the issue form can be disjointed or even unusable. Yet, GitHub currently lacks a mechanism to flag these errors. We attempted to fix five errors

[15] https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/common-validation-errors-when-creating-issue-forms

For Peer Review

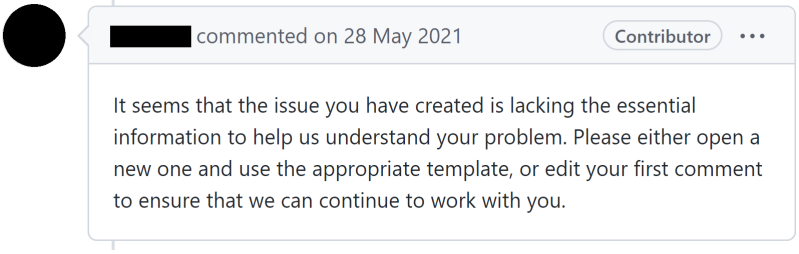

Fig. 13. A comment written by a project maintainer under an issue created with missing details

in the analyzed projects by opening pull requests directly[16]. Some of them were merged successfully by the project maintainers, while others are still open. Based on this observation, we suggest GitHub display warning messages when there are syntax errors in templates. This will prevent broken or mis-rendered templates, enhancing the user's issue reporting experience.

Additionally, we observed some issues do not conform to a template even though the project maintainers disable blank issue creation as described in Section 4.2.3. Though this case seems impossible to happen, we realized that it could be done by editing the issue body after creating it.

To illustrate, imagine a contributor initially using a template but then subsequently editing the issue to remove all template traces. Such a workaround negates the template's purpose. This behavior may hinder project maintainers from enforcing the use of templates. Therefore, we suggest GitHub should implement validation checks not just during issue creation but also during subsequent edits. This ensures persistent adherence to the chosen template, ensuring structured and useful issue submissions.

A couple of projects innovatively designed external issue forms [17], [18] to guide systematic issue creation, hinting at a desire for more sophisticated issue tracking features. For example, while Project A might want to introduce a custom field for "Device Type" to streamline issue categorization, GitHub's current system does not permit the addition of such fields. Similarly, the inability to introduce varying issue statuses or set workflow regulations based on them is a limitation. GitHub should consider augmenting its issue tracking system, allowing for the creation of new fields and the ability to define custom workflows. Projects can achieve a more tailored and efficient issue management process, potentially reducing resolution times.

7 THREATS TO VALIDITY

In the following, we discuss the threats to the validity of this study.

7.1 Construct Validity

The main threat in this category is the accuracy of the heuristic algorithms. We used heuristic algorithms for three different purposes:

- Extracting the template components from Markdown files: We developed a text parser that extracts the template components by checking the headings. However, not all templates use headings to define titles, we observed

[16] https://github.com/querydsl/querydsl/pull/3287, https://github.com/ant-design/ant-design/pull/36518, https://github.com/elementor/elementor/pull/19154, https://github.com/saltstack/salt/pull/62330, https://github.com/sourcegraph/sourcegraph/pull/39488
[17] https://issues.vuetifyjs.com/
[18] https://new-issue.ant.design/

For Peer Review

some templates prefer writing titles in different formats, such as writing headings with bold or using "·", "○" at the beginning of headings, but our parsing algorithm cannot handle such situations.

- Extracting the headings from issue bodies and comparing them with template headings: Similar to the technique in the previous threat, we used the same parser to extract headings. Then we compared the issue components with template components to calculate a conformance score. This step may not work as expected if Markdown headings are not used to represent components.
- Classifying a template component: Similar to the previous classification task, we used a keyword search algorithm to classify template components into categories. To mitigate this threat, we manually reviewed a random sample of components.
- In terms of using scripts for data collection and analysis, we are aware of the idea that potential errors in this process can affect the obtained result. That is why we performed a few rounds of code review and manual result checking throughout the study. During the implementation of the data mining, we cared to check each other's work continuously. Also, this helped us determine whether we should analyze a particular data manually or write a code to make the analysis. For example, to categorize the external links, we first tried coding that will check the description of the external links. However, during the review, we found that this approach made mistakes while categorizing, and that is why we went with the manual investigation. In the end, we can confirm that our study has a double-check mechanism.

7.2 Internal Validity

A threat to the internal validity of the study can be discussed for RQ4, where we find that issues created when at least one template existed resolved faster. However, other than the introduction of templates, there could be other possible explanations for this effect. Over the lifecycle of a project, the projects have an improved and more streamlined process. This could also have affected the time to resolution positively. Additionally, over time the contributors get more experienced in the domain, and they can solve the issues faster than before. A decrease in the TTR could also result from recruiting more developers to the projects. Finally, with automated bots, projects may have a less average time to resolution. For example, some bots automatically close issues after a certain period of inactivity, which may prevent issues from having a longer time to resolution. Correlation does not imply causation and further research is needed to explore the real effects of the issue templates.

Also, in RQ4, we compared the issue tracking metrics of zero conformance and non-zero conformance issues. The soundness of this comparison relies on the accuracy of the conformance calculation. As discussed in Section 7.1, the conformance calculation algorithm may not be reliable in certain cases.

7.3 External Validity

To understand the overview of issue templates, we sampled 100 projects from GitHub. The projects may not reflect the overall status of issue templates as GitHub hosts millions of projects. Our selection of projects is already biased since they are prominent and large-scale projects. In these projects, the maintainers naturally look for more organized solutions, such as issue templates. To get a more balanced view of issue template usage, we plan to run our analysis for all GitHub projects in our future work.

Similarly, a selection bias may affect the validity of RQ5 since we sent the survey to people who have worked on issue templates before and this group may not be representative of all software practitioners who interact with issue

For Peer Review

templates. Also, only 39 of 685 participated in the survey. Therefore, nonresponse bias may occur and the validity can be affected because of the lack of representation.

8 CONCLUSION AND FUTURE WORK

GitHub Issues, once appreciated for its flexibility in issue creation, recognized the need for standardization as this very flexibility sometimes posed challenges, especially in large-scale open-source projects. While this simplicity encouraged users to report issues promptly, it inadvertently generated challenges for maintainers, especially those handling large-scale projects, due to the inconsistency and lack of standardized details in issue reporting. Recognizing these challenges, GitHub introduced issue templates as a mechanism to facilitate standardized reporting. In this study, we explored and evaluated the use and impact of these templates within large-scale open-source projects, providing valuable insights through various research questions (RQs):

Current Usage (RQ1): 99% of the sampled large-scale projects employ issue templates. We observed and identified common template components recurring across diverse projects.

Evolution Over Time (RQ2): We studied the historical progression of template usage, noting that projects tend to leverage new template features over time, as graphically demonstrated in Figure 8.

Conformance to Templates (RQ3): A notable proportion (33%) of issues raised by contributors deviated from the template guidelines, indicating areas for further refinement.

Impact of Templates (RQ4): We discovered a significant difference in the *time to resolution* of issues in projects where templates were in place compared to those without. We realized that *No Template Available* (374.56 in days) has three times larger mean value than *Template available* (103.46 in days) Moreover, the positive influence of YAML-based templates over markdown-based templates on the time-to-resolution, the number of comments, and the number of reopened issues were statistically significant.

Maintainer Perspective (RQ5): Our survey with open-source software maintainers provided valuable insights into the perceived merits and pitfalls of using templates. Most of our respondents agree that issue templates have a positive effect on reducing time to resolution (79% of the respondents), and the number of comments (85% of the respondents) on GitHub.

In light of our findings, future research endeavors will extend the analysis to incorporate a broader range of GitHub repositories. This expansion aims to provide a more holistic understanding of the dynamics and implications associated with the utilization of issue templates within the open-source project environment. This deeper insight will hopefully guide more effective and efficient issue-tracking and resolution processes within the open-source community, fostering enhanced project development and maintenance outcomes.

Additionally, a comparative analysis of GitHub's issue template structure with those of other issue-tracking systems, such as Jira and Bugzilla, can be conducted across various dimensions. Both Jira and Bugzilla offer issue template features that, while similar, exhibit distinct differences from each other and from GitHub's offerings. Understanding these subtle disparities is essential, as they can influence the effectiveness of an issue-tracking system's templates under particular conditions. Identifying which system's issue template proves most effective, and under which specific circumstances, can provide developers with crucial insights, enabling them to manage issue-tracking processes more proficiently.

.

For Peer Review

REFERENCES

[1] Ben Bleikamp. 2016. Issue and Pull Request templates. https://github.blog/2016-02-17-issue-and-pull-request-templates/

[2] Oscar Chaparro, Carlos Bernal-Cárdenas, Jing Lu, Kevin Moran, Andrian Marcus, Massimiliano Di Penta, Denys Poshyvanyk, and Vincent Ng. 2019. Assessing the quality of the steps to reproduce in bug reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA). ACM, 86–96. https://doi.org/10.1145/3338906.3338947

[3] Songqiang Chen, Xiaoyuan Xie, Bangguo Yin, Yuanxiang Ji, Lin Chen, and Baowen Xu. 2020. Stay professional and efficient: automatically generate titles for your bug reports. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (New York, NY, USA). ACM, 385–397. https://doi.org/10.1145/3324884.3416538

[4] Bryan Clark. 2018. Multiple issue and pull request templates. https://github.blog/2018-01-25-multiple-issue-and-pull-request-templates/

[5] Jailton Coelho, Marco Tulio Valente, Luciano Milen, and Luciana L. Silva. 2020. Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects. *Information and Software Technology* 122 (6 2020), 106274. https://doi.org/10.1016/j.infsof.2020.106274

[6] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling Projects in GitHub for MSR Studies. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 560–564. https://doi.org/10.1109/MSR52588.2021.00074

[7] Santiago Dueñas, Valerio Cosentino, Jesus M. Gonzalez-Barahona, Alvaro del Castillo San Felix, Daniel Izquierdo-Cortazar, Luis Cañas-Díaz, and Alberto Pérez García-Plaza. 2021. GrimoireLab: A toolset for software development analytics. *PeerJ Computer Science* 7 (7 2021), e601. https://doi.org/10.7717/peerj-cs.601

[8] GitHub. [n. d.]. About issue and pull request templates. https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/about-issue-and-pull-request-templates

[9] Aigerim Issabayeva, Ariadi Nugroho, and Joost Visser. 2012. Issue handling performance in proprietary software projects. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 209–212.

[10] Maliheh Izadi, Kiana Akbari, and Abbas Heydarnoori. 2022. Predicting the objective and priority of issue reports in software repositories. *Empirical Software Engineering* 27 (3 2022), 50. Issue 2. https://doi.org/10.1007/s10664-021-10085-3

[11] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21 (10 2016), 2035–2071. Issue 5. https://doi.org/10.1007/s10664-015-9393-5

[12] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2019. Ticket Tagger: Machine Learning Driven Issue Classification. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 406–409. https://doi.org/10.1109/ICSME.2019.00070

[13] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2021. Predicting issue types on GitHub. *Science of Computer Programming* 205 (5 2021), 102598. https://doi.org/10.1016/j.scico.2020.102598

[14] Zhixing Li, Yue Yu, Tao Wang, Yan Lei, Ying Wang, and Huaimin Wang. 2022. To Follow or Not to Follow: Understanding Issue/Pull-Request Templates on GitHub. *IEEE Transactions on Software Engineering* (2022), 1–16. https://doi.org/10.1109/TSE.2022.3224053

[15] Bart Luijten, Joost Visser, and Andy Zaidman. 2010. Assessment of issue handling efficiency. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. IEEE, 94–97.

[16] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18 (3 1947), 50–60. Issue 1. https://doi.org/10.1214/aoms/1177730491

[17] Rahul Mohanani, Paul Ralph, Burak Turhan, and Vladimir Mandic. 2021. How Templated Requirements Specifications Inhibit Creativity in Software Engineering. *IEEE Transactions on Software Engineering* (2021), 1–1. https://doi.org/10.1109/TSE.2021.3112503

[18] Maintainers of Open-Source Projects. 2016. Dear GitHub. https://github.com/dear-github/dear-github.

[19] Sebastiano Panichella, Gerardo Canfora, and Andrea Di Sorbo. 2021. Won't We Fix this Issue? Qualitative characterization and automated identification of wontfix issues on GitHub. *Information and Software Technology* 139 (11 2021), 106665. https://doi.org/10.1016/j.infsof.2021.106665

[20] Henrique Rocha, Guilherme de Oliveira, Marco Tulio Valente, and Humberto Marques-Neto. 2016. Characterizing bug workflows in mozilla firefox. In *Proceedings of the XXX Brazilian Symposium on Software Engineering*. 43–52.

[21] Tommaso Dal Sasso, Andrea Mocci, and Michele Lanza. 2016. What Makes a Satisficing Bug Report?. In *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 164–174. https://doi.org/10.1109/QRS.2016.28

[22] Mozhan Soltani, Felienne Hermans, and Thomas Bäck. 2020. The significance of bug report elements. *Empirical Software Engineering* 25 (11 2020), 5255–5294. Issue 6. https://doi.org/10.1007/s10664-020-09882-z

[23] Yang Song and Oscar Chaparro. 2020. BEE: a tool for structuring and analyzing bug reports. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA). ACM, 1551–1555. https://doi.org/10.1145/3368089.3417928

[24] Mengxi Zhang, Huaxiao Liu, Chunyang Chen, Yuzhou Liu, and Shuotong Bai. 2022. Consistent or not? An investigation of using Pull Request Template in GitHub. *Information and Software Technology* 144 (4 2022), 106797. https://doi.org/10.1016/j.infsof.2021.106797

[25] Thomas Zimmermann, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schroter, and Cathrin Weiss. 2010. What Makes a Good Bug Report? *IEEE Transactions on Software Engineering* 36 (9 2010), 618–643. Issue 5. https://doi.org/10.1109/TSE.2010.63

For Peer Review

A SURVEY QUESTIONS

(1) What kind of issue templates have you used?
- ☐ Old Markdown: This is when a project has a single template written in Markdown.
- ☐ Markdown: This is when a project has multiple templates for different purposes written in Markdown.
- ☐ YAML (aka issue forms): This is when a project has more structured (dropdowns, required fields etc.) templates.

In the following questions, please rate the benefits of issue templates from 1 to 5. 1: Strongly disagree 2: Disagree 3: Neither agree nor disagree 4: Agree 5: Strongly agree

(2) Issues created from a template require less time to resolution.

(3) Issues created from a template require less discussion and less number of comments.

(4) Issues created from a template are less likely to be reopened.

(5) Issues created from a template are less likely to be duplicated.

Historically the earlier and more traditional issue tracking systems, such as Bugzilla and Jira, can be categorized as structured issue tracking systems. Structured issue tracking systems have the ability to define extra fields for different purposes and provide customizable workflows by introducing validation rules and custom issue states. On the other hand of the spectrum, original version of GitHub Issues which would only require a title and a description to file an issue can be considered as an unstructured issue tracking system. The recent efforts by GitHub by introducing Markdown and YAML based templates are semi-structured issue tracking systems. In the following question, please state your preference in issue tracking systems.
- ☐ Jira (or another structured issue tracking system)
- ☐ GitHub Issues with issue forms (YAML based templates)
- ☐ GitHub Issues with templates (Markdown templates)
- ☐ GitHub Issues without any templates

(6) Please explain your reasoning for the previous question.

(7) Do you have any extra comments related to the benefits of issue templates?

(8) Do you have any extra comments related to the challenges of issue template usage?

(9) Would you like to be notified when this study is published?
- ☐ Yes

(10) Please provide your email address if you want to be notified or participate in the raffle.

Received 2023; revised 2023; accepted 2023

For Peer Review

Dear Editor,

We would like to thank all of our reviewers for their valuable comments. We have carefully addressed each comment and updated the paper accordingly. Please find below the revised paper and the response to the reviewers' comments.

Thanks to the reviewers' and your comments, we believe that our paper has significantly improved.

Kind regards,

Emre Sülün
Metehan Saçakçı
Eray Tüzün

Revision plan according to the reviewers' comments

In this revision, we tried to address the review comments and improve the paper accordingly. Below, we describe the revisions that we have made to address the comments of the reviewers. The comments (C) are numbered and answered. Also, they are grouped by the reviewer.

Handling Editor

This paper presents an interesting empirical study on issue template usage, but reviewers agree that this paper is not ready due to its following issues: (1) unclear motivation, novelty, and contribution (2) some unclear setup and analysis in the empirical study. Please carefully address the issues pointed out by the existing reviews.

Answer: *Thanks for your valuable comments. Per each main issue, we performed the following:*

(1) We clarified the motivation and underlying novelty of the work in the introduction section.

We added a summary of contributions to the introduction to highlight the contributions of the study.

(2) We have added details related to our setup and analysis per reviewers' comments.

Reviewer 1

Summary

This paper studies 100 large-scale projects with 350 templates found widespread usage. Authors found projects with templates experienced reduced resolution time (374.56 days to 103.46 days) and comment count (4.90 to 4.34) compared to those without. Additionally, issue forms can help reduce resolution times, reopening, and discussions. Therefore, they concluded that issue templates can have a positive impact on large-scale projects.

Detailed points

C1. In Section 2 Background, the authors just listed the details of the issue templates described in GitHub documentation without any findings and analysis.

Answer: *Thank you for your feedback regarding Section 2: Background. Our intention in this section was to lay a foundational understanding of the issue templates as described in GitHub's documentation. We believe that providing a clear and comprehensive background is crucial for readers who may not be intimately familiar with GitHub's issue templates. This ensures that subsequent sections, especially the results section where our primary findings and analysis are detailed, are understood in the correct context. We further perform analysis on the Discussion section as well. Also, in the related work section, we mentioned findings and analyses that have been performed by other studies.*

C2. There are only a few boxes with different colours and arrows in Figure 3, and it is unclear what they mean. There should be a paragraph summarizing the workflow.

Answer: *We realized that during the submission of our document, the format of the mentioned figure was compromised, and the uploaded figure was incomplete. We fully recognize that this issue could have been identified by re-reviewing the document during submission, and for this oversight, we sincerely apologize. In the updated figure, white color is used to indicate the data mining process, blue color is used to indicate the analyzing process, and finally, green color is used to indicate documenting the final result for research questions.*

We also added a paragraph summarizing the workflow.

C3. In Section 4.2.1, the authors proposed four template categories and manually labelled 350 templates. Why are the four categories divided in this way, and what is the basis for judging these template categories, or is it just relying on intuition?

Answer: *Our intuition is mainly based on Jira and GitHub's predefined issue template categories. By default, you can create issue templates for "Bug" and "Feature Request" on GitHub and "Task" on Jira [1][2]. We determined our three categories in this way. Also, while analyzing the issues, we realized that asking project-related questions is a common practice as well. Thus, we added the "Question" category.*

We also updated the references and the text in the paper as well.

C4. To describe the template content and components in Section 4.2.2, practical examples should show what each component looks like.

Answer: *We provided an example template and highlighted its components in a screenshot in Figure 4 in Section 4.2.2*

C5. For Section 4 Methodology, it is necessary to clarify the overall structure of the method, the importance of each component of the technique, and the corresponding problems to be solved. In addition, where did each subsection come from? Why do the authors need to analyze them? The authors do not explain these clearly.

Answer: *We added a new paragraph that clearly explains the motivations of each subsection.*

C6. For Algorithm 1, what are the expected title headings? To compare two strings, some similarity algorithms can be used.

Answer: *The algorithm expects the titles in the issue body to be the same as the titles in the issue template. We performed the same basic preprocessing such as case insensitive comparing, and stripping white spaces. In our manual analysis, we have seen instances of completely removing the headers to avoid the use of templates however, only renaming the title of template components while editing the issue would be an unexpected behavior. Although we agree that a similarity algorithm can be used to improve the comparison, we think that that method brings its own challenges like selecting the right thresholds and possible false positives. Therefore, we did not consider using similarity algorithms and mention this problem as a threat to validity.*

C7. For the results, the authors state their findings but lack analysis for each part of the results (RQ1-RQ3) and unclear significance of these findings.

Answer: *RQ1.1 was exploring the status of GitHub Issue Templates usage in a sample of 100 large-scale projects. As we stated in the summary boxes, we found that only one out of 100 projects does not use issue templates. The most frequent template combination uses two templates (One for reporting bugs and one for suggesting new features). RQ 1.2 was exploring the common components of issue templates, which was listed in Table 2. We were able to identify 11 most common components that was used in the templates. This could be helpful for practitioners while definining new issue templates. We added a new summary box to Section 5.1.2 to highlight this finding.*

RQ2 was exploring the evolution of issue templates over time, as we have indicated in the Summary box, large-scale projects tend to use new generation issue templates over time. As Figure 8 and 9 suggests there is a trend over using more structured /newer templates. (YAML and a combination of markdown and YAML templates are getting more popular) Per your comment, we added more details about how to interpret Figure 8 and 9.

RQ3 was exploring the conformance ratios of practitioners to the issue templates. We provided more discussion about the implications of results related to RQ1/RQ2/RQ3 in the discussion section which was extended via C8 as well.

C8. For Section 6 Discussion, when giving some suggestions, the author should show the benefits or current disadvantages and compare the two by using some examples.

Answer: *For every suggestion in this section, we provided concrete examples that would highlight the benefits of the proposed approach compared to the current version of GitHub. Thanks to your comment, this section is more clear for the readers.*

C9. The Section Conclusion and future work should be modified to summarize the overall result and significance. More details related to future work should be added.

Answer: *Based on your comments, we have rewritten the conclusion and future work, adding the corresponding numerical results as you suggested.*

C10. The structure and presentation of the article need to be overhauled, and its significance is unclear. ie. The motivation and the novelty of this work is unclear. What adds to the body of technical knowledge in the field?

Answer: *This study examines the effectiveness of structured templates in issue tracking systems, with a specialized focus on software development projects on GitHub Issues, which is highly popular in software industry. Our investigation involves a pragmatic comparison of GitHub Issues managed with and without these structured templates, intending to unravel clear, empirical evidence of their overall impact. The exploration encompasses various dimensions such as the current utilization status of issue templates, their evolutionary trajectory, and a quantitative assessment of their influence on pivotal metrics like time to resolution, frequency of issue reopening, and the number of comments generated. Complementing our analytical findings, we also incorporate insights gleaned from surveys conducted with project maintainers, enriching the depth and breadth of our conclusions. The findings from this research will offer valuable insights, helping software developers make informed decisions based on empirical data to improve issue tracking processes.*

We have edited the introduction according to the above paragraph as well. We believe this clarifies the overall aim and highlights the significance of the study.

Reviewer 2

Summary

This study examines the use, evolution, and impact of issue templates in large-scale open source projects. The authors analyze 350 templates and their past versions from 100 projects. They evaluate 1.9 million issues for template conformance and impact. They also surveyed maintainers. They note reduced resolution time and comment count from the use of templates, and a preference for issue forms, as those further reduce resolution time, number of reopenings, and discussion extent.

Detailed points

C11. I thank the authors for their submission. Overall, there is quite a lot I liked in the paper. This study examines a very interesting topic, and has the potential to be of use to software development practitioners. The motivation for performing the study is clear and evident. The research questions and study design are largely clear and appropriate. In addition, the paper is nicely written. The discussion offers practical advice. The Threats to Validity section is also good - I highly appreciate the discussion of how analyses were checked for correctness. I also thank the authors for providing a replication package.

Answer: *Thank you for your kind words, in the following, we have addressed your queries.*

C12. The replication package is provided as a Figshare link. Figshare offers an option to assign a DOI. I recommend using the DOI link instead so that the link will still work even if Figshare changes their own link structure. It would be nice to add a README to the replication package to give an overview of the data and how it should be examined or used.

Answer: *As the reviewer advised, we started to use the DOI link instead of the Figshare link. We added a readme file that explains the contents of our replication package. The doi link of figshare is the following: https://doi.org/10.6084/m9.figshare.21587088*

C13. Figure 3 is confusing. Most of the boxes are empty. Is this on purpose? If not, the figure should be fixed. If so, it should be clarified.

Answer: *We realized that during the submission of our document, the format of the mentioned figure was compromised, and the uploaded figure was incomplete. We fully recognize that this issue could have been identified by re-reviewing the document during submission, and for this oversight, we sincerely apologize. In the updated figure, white color is used to indicate the data mining process, blue color is used to indicate the analyzing process, and finally, green color is used to indicate documenting the final result for research questions.*

C14. The authors assigned templates to four categories. It would be nice to provide a definition of each category to clarify the authors' views on each. In particular, it is not clear why "Task" is different from "Feature Request" (additionally, some of the examples of the Other category could be considered as Tasks, e.g., a library change). Similarly, it would be nice to see similar definitions for the components of templates.

Answer: *Thanks for your suggestion. Per your comment, we have now added the following category descriptions to the paper.*

1. Bug: An issue type that reports unexpected or unintended behavior in a software project.

2. Feature Request: An issue type used to suggest new functionality or improvements to existing features in an open-source project.

3. Task: An issue type representing a specific action or piece of work that needs to be completed within the project, often related to maintenance or development.

4. Question: An issue type for seeking clarification, assistance, or information related to the project or its usage.

Task and Feature Request are different because a task is a specific action or piece of work to be completed within a project, whereas a feature request suggests new functionality, focusing on the desired outcome rather than the specific steps to achieve it.

We also provided definitions for the components of templates as you suggested. They are added to paper as well:

1. Description: Concisely outline the bug, providing a clear overview of the problem to set the stage for further details.

2. Environment: Specify where the bug was identified, including details like operating system, browser, and device used.

3. Software Version: Indicate which version of the software the bug is occurring in, helping to identify if the issue is version-specific.

4. Reproduction Steps: Offer a clear, sequential guide, enabling others to replicate the issue accurately for debugging.

5. Expected Result: Clearly describe the intended outcome or behavior in the absence of the bug.

6. Actual Result: Detail the actual, flawed outcome caused by the bug to highlight discrepancies from the expected result.

7. Logs: Incorporate relevant error logs or messages that could shed light on the underlying issues causing the bug.

8. Motivation: Express why it's essential to resolve this bug, outlining its impact and the necessity for its resolution.

9. Alternative Solutions: Suggest temporary workarounds or solutions that could be applied while waiting for a complete fix.

10. Additional Context: Include any supplementary information like screenshots or videos that could aid in understanding and resolving the bug more efficiently.

C15. For answering RQ4, the authors compare issues following a template to issues that do not. How are issues separated into these two categories? Is there, for example, a threshold in the conformance score? In the results, it looks like you used a score of 0, but this could be made more explicit in the methodology section.

Answer: *If an issue has a conformance score greater than zero, that means the corresponding issue template's at least one heading is used while generating the issue. We consider this case as the issue template is used by the issue. Since we care about whether issues are created while considering an*

issue template or not, we are looking for the slightest clue that a portion of the issue template is used while creating an issue. That is why we do not have a threshold value and we categorized the issues into ones with a conformance score of zero (issues that do not use issue templates) and ones with a conformance score greater than zero (issues that use issue templates).

The reason why we do not use threshold value is added to the paper as well.

C16. In addition, the methodology for RQ4 should make it clear that an effect size test was used (I had this comment in my review until I got to the results and saw that a test was used), and the hypotheses for statistical analysis should be stated explicitly. The methodology mentions three hypotheses, but there are nine in the results - and I had to infer them from Table 3, possibly leading to misinterpretation. Please make these hypotheses explicit.

Answer: *Thank you for your comment, now we added a paragraph stating the hypothesis explicitly as well as mentioning the use of effect size in the methodology section for RQ4. We also corrected the "3" to "9".*

C17. How was Mann-Whitney applied? Two-sided, or a single-sided variant? In addition, effect size tests are generally only applied when Mann-Whitney shows significance first.

Answer: *As Table 6 suggests, the 5 of 9 hypotheses show significance results. That's why we kept effect size values for all entries. We applied the single-sided variant test because we tested whether one group of data is statistically less than the other group with respect to some characteristic.*

C18. The interpretation of the effect sizes in RQ4 should be discussed more. Notably, I see that there are no large effect sizes in Table 3. In fact, many of the effect sizes are insignificant, except for less time to resolution when a project has a template (medium) and when a YAML-based template is used (small). I would suggest being careful with making strong statements about the effects when the effect sizes do not support them.

Answer: *Although the effect sizes are not large, the differences between the distributions are significant (according to p-value). In summary boxes, we admitted the cases where the effect size is insignificant.*

C19. For RQ5, how was qualitative analysis of open-ended questions performed? Thematic analysis?

Answer: *In our survey, the open-ended questions were 6th through 8th.*

(6) Please explain your reasoning for the previous question.
(7) Do you have any extra comments related to the benefits of issue templates?
(8) Do you have any extra comments related to the challenges of issue template usage?

In the paper, quotations pertaining to Question #7 and Question #8 were utilized. Since we exclusively selected these quotations, there wasn't a necessity for employing any form of thematic analysis.

C20. It is not very clear how Figure 8 should be read. A little more explanation of what is depicted would be appreciated.

Answer: *The main purpose of Figure 9 (In the final version, Figure 8 the reviewer refers to corresponds to Figure 9) is to propose the different transitions that are made between different issue*

template types. The gray lines on the figure indicate those transitions. For example, it can be seen that a large number of projects stopped using "Old Markdown" templates and started to use "Markdown-based" templates. This also can be verified in Figure 8. Realize that when the number of "Old Markdown" templates started to drop, the number of "Markdown-based" templates started to increase. In general, Figure 9 gives insight into the number of transitions from one template type to another template type.

C21. In the discussion, the authors mention submitting pull requests fixing issues in YAML forms, but the links are redacted. As this submission is not double-blind, there is no need for redaction. It would have been nice to see the suggested fixes. In addition, it would be interesting to discuss briefly what these fixes were (e.g., are there common mistakes made?).

Answer: *Thank you for your comment. Now, we added the following links and a brief discussion.*

- *https://github.com/querydsl/querydsl/pull/3287*
- *https://github.com/ant-design/ant-design/pull/36518*
- *https://github.com/elementor/elementor/pull/19154*
- *https://github.com/saltstack/salt/pull/62330*
- *https://github.com/sourcegraph/sourcegraph/pull/39488*

Three of them are merged, one of them is still open, and another one is closed (The reason for closure was "The path .github/ or scripts/ and CHANGELOG is only maintained by team members. This current PR will be closed and team members will help on this.." We emailed the maintainers and are still waiting for an answer from them.

The most common mistake was related to the special YAML format (required key-values) of the templates. The developers skipped some minor details that were causing the wrong rendering of the templates.

C22. Minor comments:

- Page 5: "templates, pull request" -> "templates, a pull request"

- Figures 5-6 are not very readable in black and white (e.g., when printed). I suggest differentiating, e.g., the "bug" and "feature request" colors more.

- Page 17: "less number of" -> "fewer"

Answer: *We fixed the mentioned wording issues in our paper. Also, as the reviewer addressed, we tried different color palettes on our figure, such as black-white or different colors. However, we could not find a palette that looks appropriate on both printed black-white and printed colorful versions of our paper. That is why we decided to keep original versions of our figures in this revision, but this problem can be discussed with the editor in the future if our paper is accepted.*

Reviewer 3

Summary

The paper explores the utilization of templates for defining GitHub issues. Based on the conducted analysis, the authors demonstrate that adopting templates substantially reduces issue resolution times, minimizes the occurrence of issue reopenings, and leads to more concise related discussions.

Detailed points

C23. The paper is well-written and well-organized. The underlying hypothesis is logical, and the research questions pursued by the authors are well-defined. Overall, I found the paper interesting

Answer: *Thank you for your kind words, in the following, we have addressed your queries.*

C24. Analysis of Issue Templates: The authors are encouraged to provide additional details regarding the methodology used for parsing issues. It is critical to thoroughly discuss the accuracy of the developed mechanism used to detect the presence of specific templates. Errors in this phase could undermine the entire paper. The section focusing on issue parsing should be expanded and elucidated, incorporating concrete examples of issues and their parsing process. This inclusion could substantiate the validity of the template detection process.

Answer: *We agree with the idea that errors during parsing an issue can potentially affect the result of the research questions That's why, in designing our algorithm, we took great care to identify potential corner cases. If those cases are fixable, we fixed it. However, if those cases are not fixable, we exclude them and mention the details of that case in the threats to validity section. For example, while calculating the conformance scores, we only considered the issues that used the heading structure with the "#" character. Also, we mentioned characters that are used for heading structure but excluded from our analysis such as using "*" symbol or "-" symbol as heading structure.*

Since Markdown-based templates are actually editable free texts, questions in the templates can be deleted or modified. To parse this free text, we used a markdown parser called Marko [3]. By the help of this parser, we were able to parse the headings for each issue and for each markdown-based template. While using this parser, we found a bug and later on this was fixed as well (https://github.com/frostming/marko/issues/119)

On the other hand, the structure is more strict for YAML-based templates, and project maintainers can set validation rules to force users. We did not need a parser for YAML-based templates.

C25. Conformance Measurement Algorithm: Further elaboration is needed in explaining the second conditional statement. What constitutes a "valid body"? The authors mention the identification of the strings "No response" or "N/A," but this approach might require more sophistication. It is advisable to justify the representativeness of such strings or explore alternatives that offer a more comprehensive analysis. Additionally, comparing anticipated title headings and actual titles within issues warrants detailed clarification, especially concerning semantic considerations during this phase.

Answer: *To calculate the conformance scores, we want to understand whether a markdown-based issue template's heading is formed appropriately. By saying appropriately, we mean authors must answer the question/requested information that is given in the heading. Depending on the type of the*

question, it is very difficult / may be impossible to assess if the answer is really a "valid" answer. Thus we mainly checked if the answer is not empty or not filled with a filler sentence for leaving the answer empty as "N/A" or "No response". The reviewer pointed out that, in this case, there might be many other ways to leave a heading empty and how much we can rely on our conformance algorithm. To maximize the accuracy of our conformance algorithm, we applied the following manual test procedure. We selected random 100 conformance score calculations from our algorithm. We also checked the related issue by eye and manually computed the conformance score. Finally, we compare our calculations with our algorithms' results on an Excel file, and if there is a wrong calculation, we note down why it is miscalculated. We repeated this procedure. In the end, we increased the accuracy of our algorithm from 60~% to 80~% by addressing the corner cases we have encountered in several iterations.

C26. Number of Templates: The authors introduce the figure of 350 templates without clear contextualization. It would be beneficial to elucidate the source or rationale behind this number.

Answer: *In the projects that we have selected (100 projects) we have encountered a total of 350 templates. Thus, some projects may have multiple templates and Figure 5 shows the distribution per project.*

C27. Effects of Issue Templates: The authors mention that out of "1,661,788 issues, 209,654 issues have zero conformance to templates, while 394,285 issues have non-zero conformance." The disposition of the remaining issues needs to be addressed. Providing insights into how these unmentioned issues fit into the framework of the study would enhance comprehensiveness.

Answer: *We analyzed 1,661,788 closed issues (a precondition for effect analysis is being closed), of which 209,654 have zero conformance, 394,285 have non-zero conformance, and conformance cannot be calculated for the remaining 1,057,849 (NAN conformance). The reason for this is the same as explained in Section 5.3: "issues are not eligible for conformance analysis because there was no template at that time (554331) or we could not parse the issue to calculate the conformance (648272) due to the usage of non-formal denotation for headings in Markdown-based templates (we expanded the details of such cases in the Construct Validity section).*

C28. Sampled Projects: The extent to which the sampled 100 projects accurately represent large-scale GitHub projects requires clarification. To enhance precision, I recommend refining the title, abstract, and introduction to establish the scope of the study upfront. Defining the specific parameters of analysis early on, rather than waiting until the methodology and results sections, would provide a clearer delineation of the research.

Answer: *Thanks for your comment. The title and the abstract already include the "large-scale" keyword. Considering the reviewer's comment, we defined what we mean by defining "large-scale project" in the introduction section as well.*

References

[1] “Configuring issue templates for your repository,” GitHub Docs, https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/configuring-issue-templates-for-your-repository (accessed Oct. 9, 2023).

[2] “What are issue types?,” Atlassian Support, https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-types/ (accessed Oct. 9, 2023).

[3] “Marko,” PyPI, https://pypi.org/project/marko/ (accessed Oct. 15, 2023).