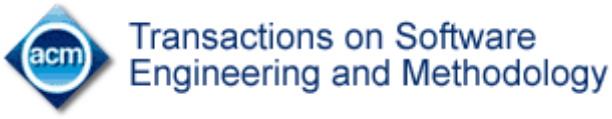


# An Empirical Analysis of Issue Templates Usage in Large-Scale Projects on GitHub

| | |
|---|---|
| Journal: | *Transactions on Software Engineering and Methodology* |
| Manuscript ID | TOSEM-2023-0169 |
| Manuscript Type: | Paper |
| Date Submitted by the Author: | 12-Jun-2023 |
| Complete List of Authors: | Sülün, Emre; Bilkent University, Computer Engineering<br>Saçakçı, Metehan; Bilkent University, Computer Engineering<br>Tüzün, Eray; Bilkent University, Computer Engineering |
| Computing Classification Systems: | Software and its engineering, Software development process management, Open source model |
| | |

# 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 significant 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. Use of issue forms further significantly decreased resolution time, the number of reopenings, and discussion extent. Thus, our research underscores issue templates' positive impact in 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 (June 2023), 24 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.


Manuscript submitted to ACM

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. 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?

The rest of the paper is organized into the following sections. In Section 2, we provide a brief overview of issue templates in 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

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

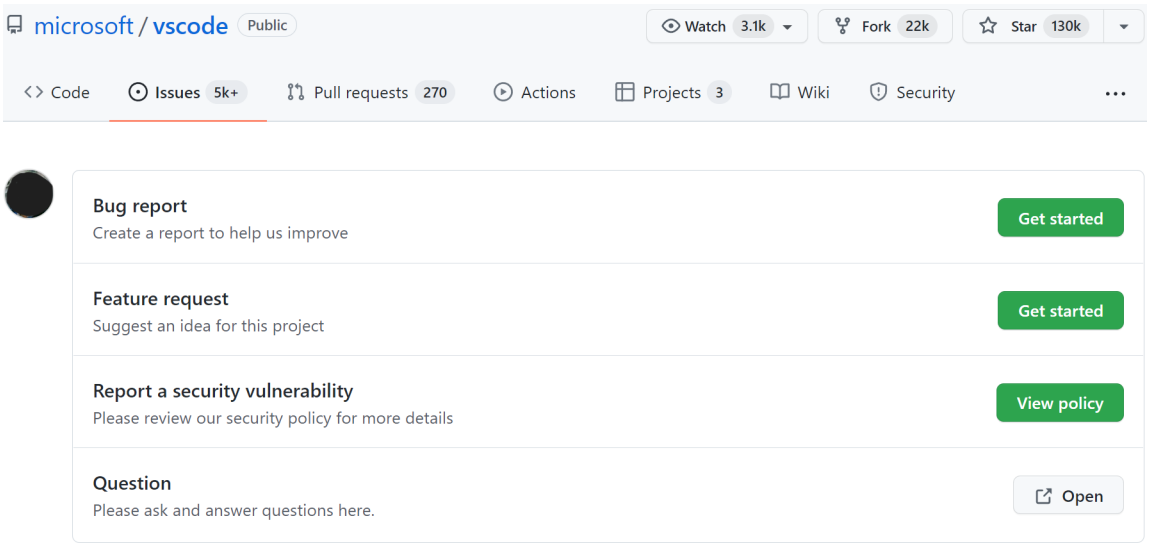


Fig. 1. Issue template chooser menu of Microsoft VS Code

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.
- 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 to 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].

[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

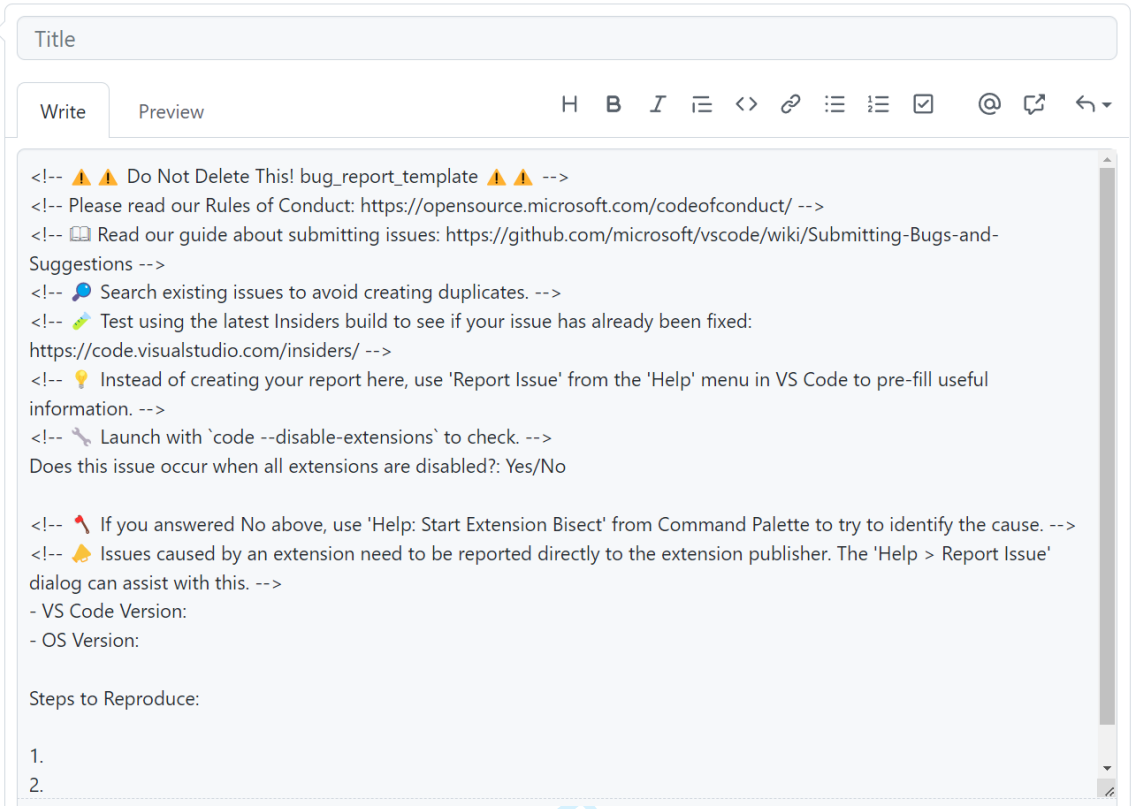


Fig. 2. Bug reporting issue template of Microsoft VS Code

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. 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

[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/

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.

### 3.2 Template Usage in Software Development

Similar to issue templates, 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

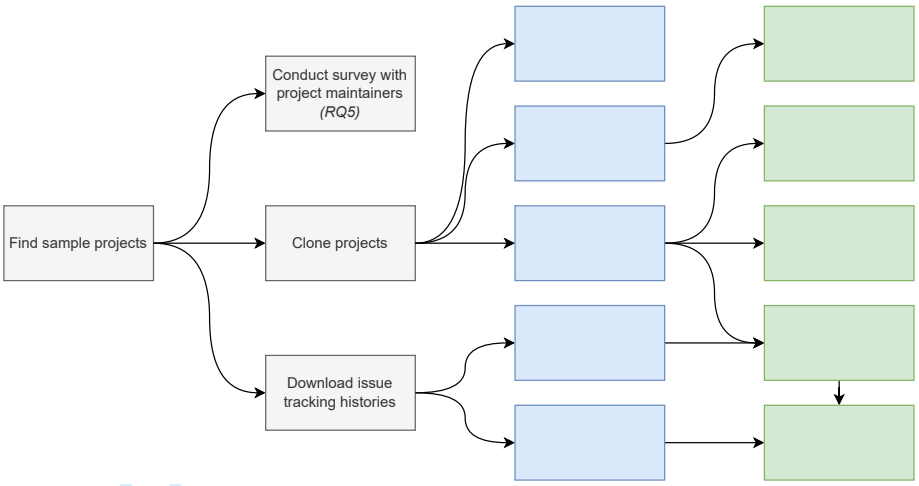


Fig. 3. The overview of the study

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.

### 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 used 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 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. Then, we parse the template content and identify the components. In addition to the templates, we also analyze the content of the template chooser menu and classify its use cases. Finally, we analyze the conformance of issues made by contributors.

*4.2.1 Template Category.* A template can be classified into multiple categories, such as bugs or feature requests. To determine those categories, the 350 templates are independently and manually labeled by two authors of the study. After the two authors separately determined their categories, three authors compared the results and resolved the conflicts in a meeting. Finally, we came out with four categories: *Bug*, *Feature Request*, *Task*, and *Question*.

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

[6] https://github.com/type-challenges/type-challenges
[7] https://figshare.com/s/822a14149ba38e9ae9f8

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*.

*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.
- *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. 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 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. 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.

```
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

### 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. 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 three 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.

### 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 it 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.

## 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[8] uses organization-level templates, 1 project[9] 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[10] 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 4.

[8] https://github.com/atom/atom
[9] https://github.com/vuetifyjs/vuetify
[10] https://github.com/sympy/sympy

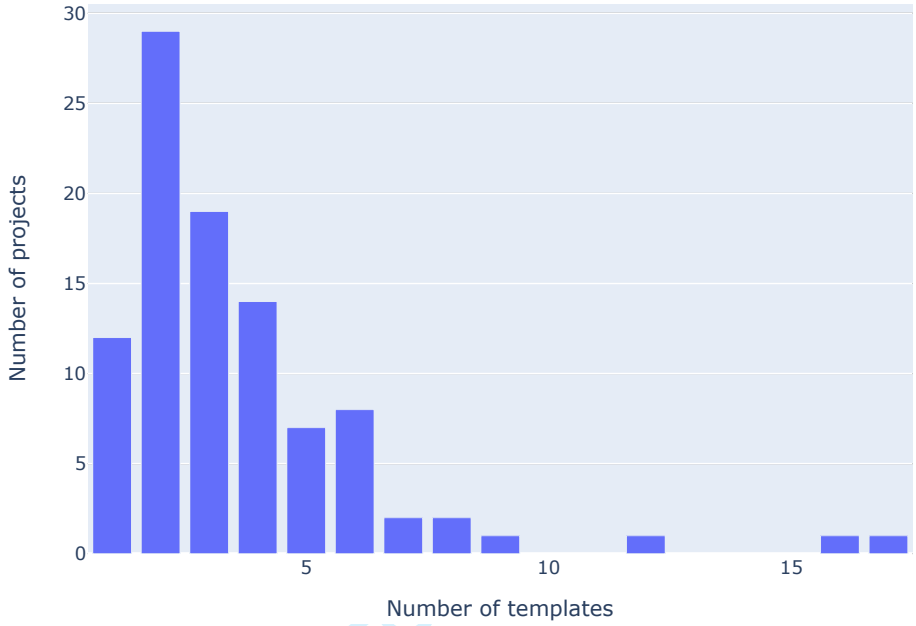


Fig. 4. The distribution of the number of templates per project

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 5.

Figure 6 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 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”

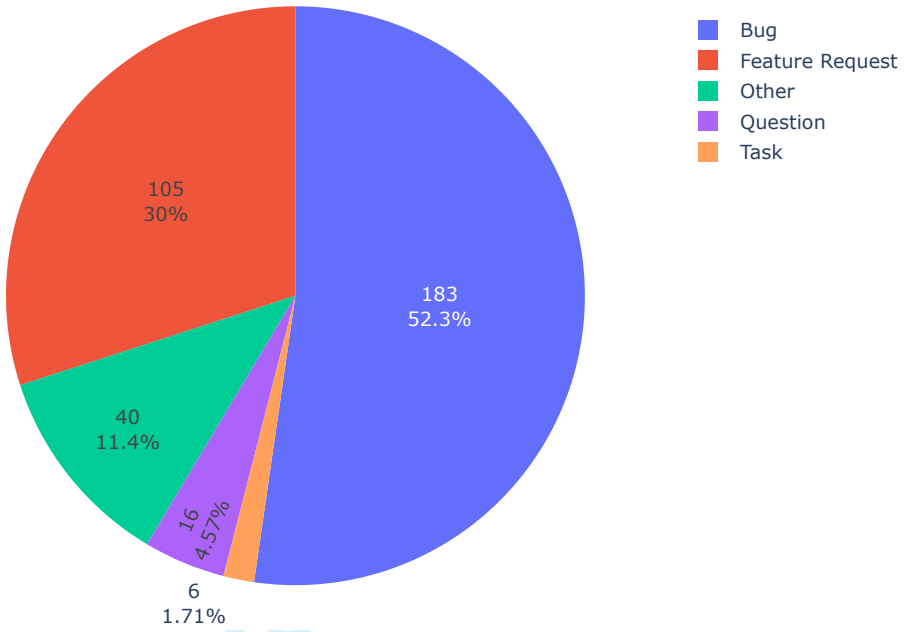


Fig. 5. The distribution of the template categories

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 |

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

### 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 7 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 8. A trend of upgrading the template type is observed in this figure.

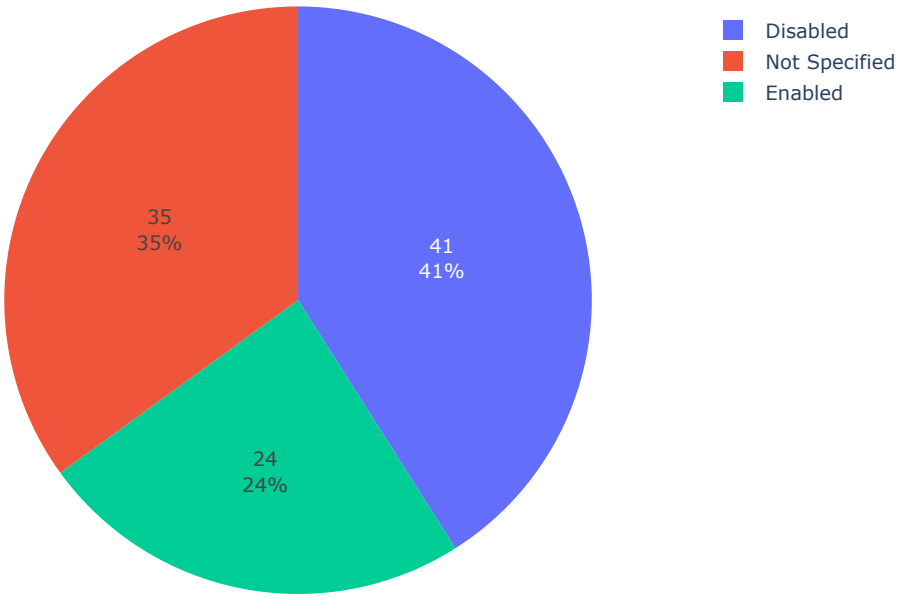


Fig. 6. Project's blank issue allowance distribution *(If not specified, it is enabled by default)*

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 |

Large-scale projects tend to use new generation issue templates over time. As Figure 7 and 8 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

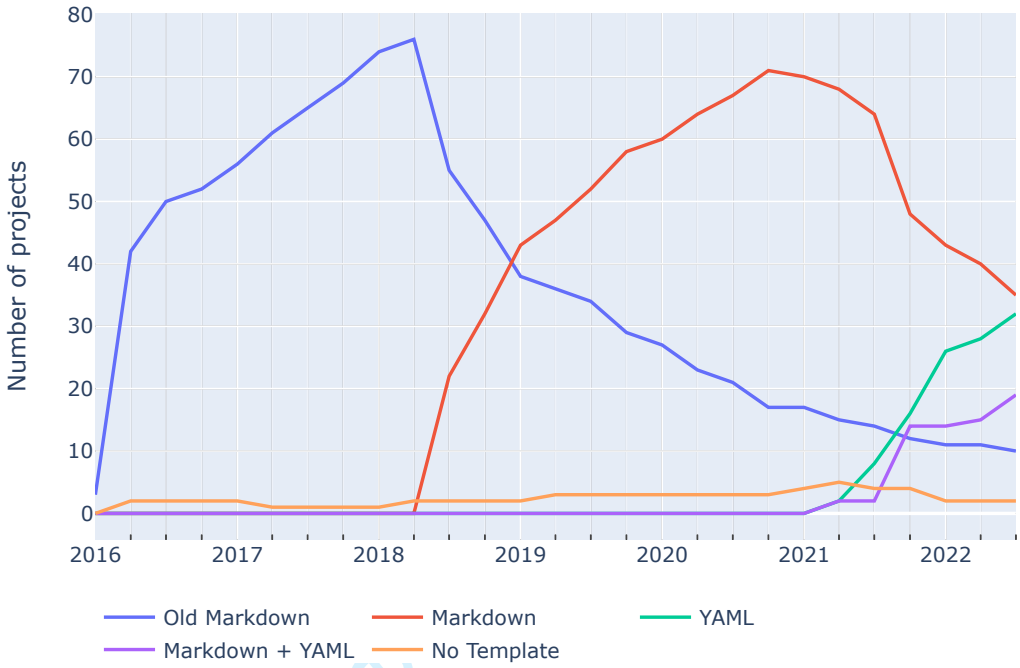


Fig. 7. The historical evolution of issue template usage

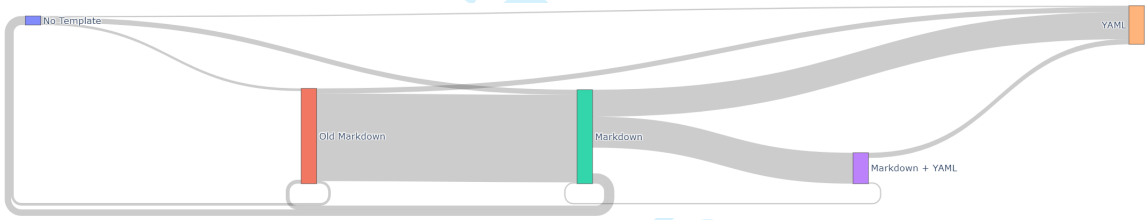


Fig. 8. Transition between different template types

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 is 0.75. Figure 10 presents the conformance analysis results. A visual summary of the distribution of 1,916,057 issues is given in Figure 9.

### 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*

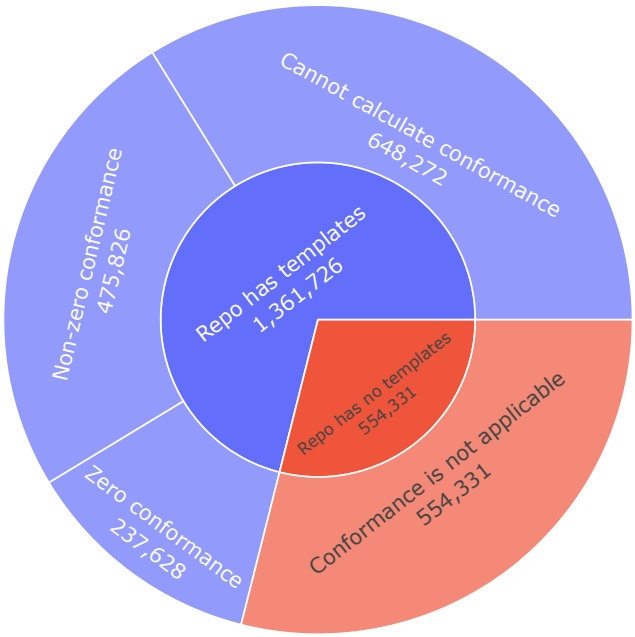


Fig. 9. Distribution of the issues. The inner circle shows whether projects use issue templates when an issue is created, while the outer circle shows the conformance category.



*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.

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. |
|---|

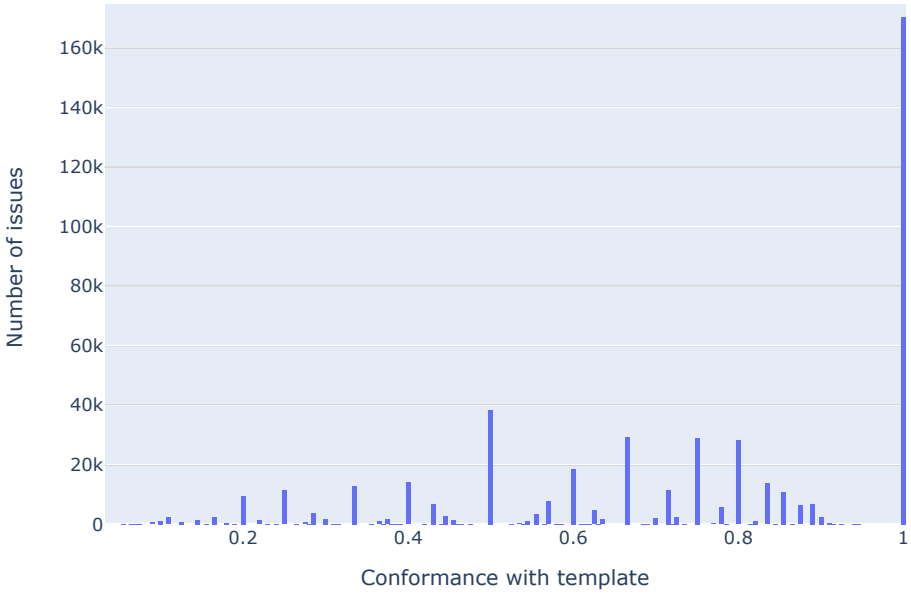


Fig. 10. The conformance of issues (excluding zero conformance)

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 |
| Less number of reopens | 0.44 | 0.01 |
| Less number of comments | 1.00 | 0.09 |
| Whether the project has a template or not | | |
| Less time to resolution | 0.00 | 0.59 |
| Less number of reopens | 1.00 | 0.03 |
| Less number of comments | 0.00 | 0.07 |
| YAML-based template over Markdown-based template | | |
| Less time to resolution | 0.00 | 0.37 |
| Less number of reopens | 0.00 | 0.06 |
| Less number of comments | 0.00 | 0.10 |

### 5.5 RQ5: Perception of Project Maintainers

The responses of project maintainers related to the benefits of issue templates are shown in Figure 11. 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, less number of comments and less likely to be reopened. On the other hand, they do not think that issue templates affect the likelihood of being duplicated.

In detail,

- 79% of the respondents agree that issues created from a template require less time to resolution.

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 |

- 85% of the respondents agree that issues created from a template require less discussion and less number of 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.

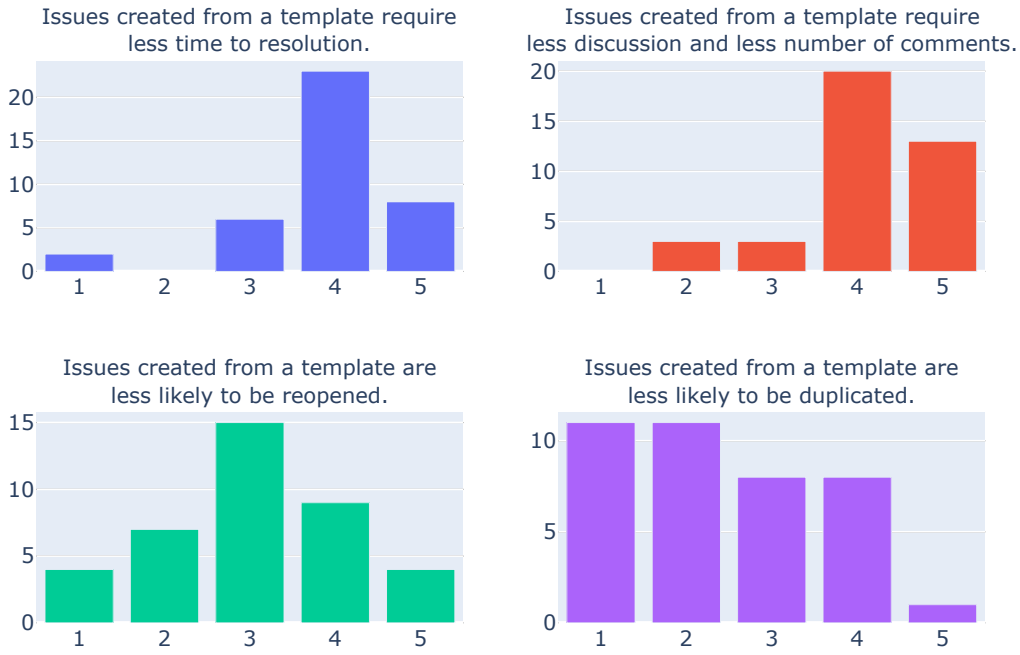


Fig. 11. Survey responses. 1: Strongly disagree 5: Strongly agree

## 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.

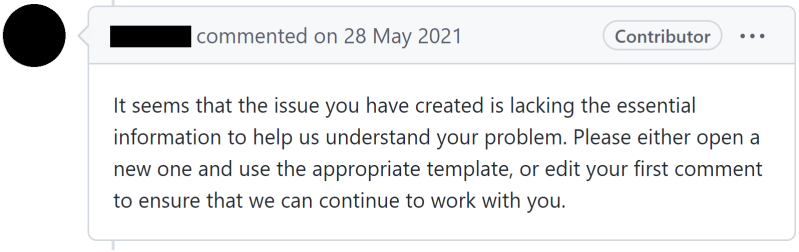


Fig. 12. A comment written by a project maintainer under an issue created with missing details

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.* On large-scale projects, issue templates help project maintainers to organize the issue tracking workflows. We suggest they use at least two templates: one for reporting bugs and one for requesting new features since introducing new features and fixing bugs are the most common activities in all projects. As Table 2 demonstrates, different types of templates (such as Bug, Feature Request, Question, Task, or other) require different types of information to be entered. It is a good practice to have multiple templates to acquire adequate information.

In relation to 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.* Allowing users to create issues from scratch decrease the effectiveness of issue templates. If an issue does not fit into any template, it should be discussed elsewhere, such as on a Q&A platform.

*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, if a software version is needed, a text explaining how to obtain the version can be written in the template. 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 12 for an example). Therefore, we suggest the contributors conform to the templates as much as possible, especially for the required fields.

### 6.3 Suggestions for GitHub

During the analysis, we noticed that some templates written in YAML had syntax issues [11]. 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. We attempted to fix five errors in the analyzed projects by opening pull requests directly[12]. 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.

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. This behavior may hinder project maintainers from enforcing the use of templates. Therefore, we suggest GitHub to enforce the same validation rules while editing.

Interestingly, two projects developed external issue forms[13], [14] that guide users to create issues more systematically. Although it is subject to discussion, this shows that these projects need more advanced features of issue tracking systems. One should also note that with GitHub Issues, creating new fields is still impossible, which would inhibit simple querying operations on fields. It is also impossible to add different issue states and create workflow rules based on states.

## 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 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 checkings throughout the study. During the implementation of the data mining, we cared to check each other's

[11] https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/common-validation-errors-when-creating-issue-forms
[12] URLs Redacted for review
[13] https://issues.vuetifyjs.com/
[14] https://new-issue.ant.design/

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 has worked on issue templates before and this group may not be representative of all software practitioners who interact with issue 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 serves as a streamlined issue tracking tool for open-source projects. Originally, any user could create an issue by simply providing a description. Although this flexibility facilitated rapid issue creation for contributors, it often led to maintenance issues for open-source maintainers, particularly in the context of large-scale projects. To address this, GitHub introduced the issue templates feature to aid project maintainers in standardizing their issue reporting process. This study analyzed the current use of issue templates in large-scale open-source projects and concluded with the following findings:

In RQ1, we explored the present state of issue templates in large-scale projects, revealing their widespread utilization in the sample projects. Furthermore, we identified common components prevalent across varying projects.

Under 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 7.

For RQ3, we assessed the conformance of issues created by contributors and found that 33% of these did not adhere to the guidelines.

In RQ4, we scrutinized the impact of templates. We discovered a significant difference in the *time to resolution* of issues in projects where templates were in place compared to those without. 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.

Finally, in RQ5, we conducted a survey to understand the perceived advantages and challenges of templates from the viewpoint of recent open-source software maintainers.

Moving forward, we intend to expand our analysis to include more GitHub repositories to gain a comprehensive understanding of issue template utilization.

.



## 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

## 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