AN EMPIRICAL ANALYSIS OF ISSUE TEMPLATES ON GITHUB

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Emre Sülün
January 2023

An Empirical Analysis of Issue Templates on GitHub
By Emre Sülün
January 2023

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Eray Tüzün (Advisor)

Uğur Doğrusöz

İsmail Sengör Altıngövde

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

AN EMPIRICAL ANALYSIS OF ISSUE TEMPLATES ON GITHUB

Emre Sülün
M.S. in Computer Engineering
Advisor: Eray Tüzün
January 2023

Many open-source software projects use GitHub Issues for issue tracking. Unlike other issue trackers, the initial versions of GitHub Issues were highly flexible and had no standard way of using it. Its unstructured nature may have made it prone to incomplete issue reports that may negatively affect software development and maintenance productivity. To potentially address these problems, GitHub introduced issue templates in 2016. This thesis aims to reflect the current status of issue template usage by mining open-source projects. Also, we analyze how the templates have evolved since their introduction in 2016 and further investigate the impact of issue templates on several issue tracking metrics, such as time to resolution, the number of reopens, and the number of comments. We evaluated 350 templates and their previous versions from 100 large-scale and popular open-source projects. We also analyzed 1,916,057 issues to understand their conformance to templates and the impact of issue templates. Lastly, we conducted a survey with open-source software maintainers to understand their opinions about issue templates. We found that issue templates are almost always used (99 out of 100 projects). The historical analysis suggests that issue forms, which are more structured issue templates, started to gain popularity over vanilla issue templates. We also observed that issues created when the project has an issue template are statistically resolved faster (p-value 0.00, effect size 0.59) and have less number of comments. Similarly, when issue forms are used, time to resolution, the number of reopenings and the length of discussion significantly decrease. According to the survey, 85% of project maintainers agree with the benefits and they believe issue templates construct a balanced midpoint between the flexibility of vanilla issues and the strictness of other issue tracking systems such as Jira.

Keywords: issue templates, issue forms, issue tracking, github, open-source software, empirical analysis.

ÖZET

GITHUB ISSUES İÇİN KULLANILAN ŞABLONLARIN DENEYSEL BİR ANALİZİ

Emre Sülün
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Eray Tüzün
Ocak 2023

Pek çok açık kaynaklı yazılım projesi, olay kaydı takibi için GitHub Issues kullanmaktadır. Diğer olay kaydı takip araçlarının aksine, GitHub Issues kullanıcılara esneklik sağlamakta ve kullanımı için standart bir yol sunmamaktadır. Bu esneklik ve standart eksikliği, yazılım geliştirme ve bakım süreçlerini olumsuz etkileyebilecek eksik olay kayıtlarına yol açabilmektedir. GitHub, bu problemi engellemek için 2016'da olay kaydı şablonlarını kullanıma sundu ve birçok proje bu özelliği kullanmaya başladı. Bu tez, açık kaynaklı projeleri inceleyerek olay kaydı şablonu kullanımının mevcut durumunu ve tarihsel gelişimini ortaya koymayı amaçlamaktadır. Bu amaçla, 100 popüler açık kaynaklı yazılım projesinden 350 şablonu ve bu şablonların önceki sürümlerini analiz ettik. Açılan olay kayıtlarının şablonlara uygunluklarını ve şablonların etkisini anlamak içinse 1,916,057 olay kaydını inceledik. Son olarak, şablonlar hakkındaki görüşlerini anlamak için açık kaynak yazılım geliştiricilerle bir anket gerçekleştirdik. Analiz sonucunda, şablonların neredeyse her zaman kullanıldığını gözlemledik (100 projeden 99'u). Tarihsel analizimiz, daha yapılandırılmış ve kısıtlı şablonların normal şablonlara göre popülerlik kazanmaya başladığını gösteriyor. Ayrıca yaptığımız istatiksel analiz, şablon kullanımı durumunda olay kayıtlarının istatistiksel olarak daha hızlı çözüldüğünü (p değeri 0.00, etki büyüklüğü 0.59) ve daha az etkileşimle tamamlandığını göstermektedir. Ankete göre ise, geliştiricilerin %85'i şablonların faydaları konusunda hemfikir ve geliştiriciler şablon kullanımının orijinal GitHub Issues olay kaydı sisteminin esnekliği ile Jira gibi diğer olay kaydı takip sistemlerinin katılığı arasında dengeli bir orta nokta oluşturduğuna inanıyorlar.

Anahtar sözcükler: olay kaydı şablonları, olay kaydı formları, olay kaydı takibi, github, açık kaynaklı yazılım, deneysel analiz.

Acknowledgement

I would like to express my sincere gratitude to my thesis advisor, Eray Tüzün, for his invaluable guidance, encouragement, and support throughout my graduate studies. His expertise and support were essential to the success of my research. I would also like to thank the reviewers of my thesis, Uğur Doğrusöz and İsmail Sengör Altıngövde, for their constructive feedback and helpful suggestions.

I would like to extend my appreciation to my colleagues and friends at Bilkent University Software Engineering and Data Analytics Research Group for their support and companionship during my graduate studies. Special thanks to Metehan Saçakçı and Erdem Tuna for their help in various aspects of my research and writing process. I would also like to thank my family and girlfriend for their unwavering love, support, and encouragement throughout my educational journey.

I am genuinely grateful to have had such a supportive group of people around me. This research would not have been possible without the support.

Contents

List of Figures

List of Tables

Chapter 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. 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. 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 [1].

Although the unstructured notion of GitHub Issues provides a flexible and faster way to create issues, in the long run, it hinders maintaining the issue tracking system [1]. This is especially true for large projects where a substantial number 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 [2, 3]. However, GitHub projects were suffering from the absence of these features because of its lightweight design, and the maintainers 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 [4]. One of the problems that were expressed as; "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."

1.1 Research Questions

To address the problems mentioned earlier, GitHub introduced the issue templates in 2016, aiming to improve the issue handling process [5]. Although the issue templates have been around for more than six years now, their effects are still unknown to the software engineering community. Based on this motivation, 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?

  RQ1.1 What is the ratio of projects that started to use 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?

1.2 Contributions of the Thesis

The main contributions of this thesis are as follows:

- We present an overall picture of the current status of issue templates. Moreover, we present a historical analysis and show how the use of templates evolves over time.
- We statistically analyze the effects of issue templates on issue tracking metrics and demonstrate the benefits.
- We list recommendations for project maintainers to improve their issue tracking process based on the statistical analysis and the survey responses.
- Throughout the study, we encountered some incorrect use of issue templates and helped open-source projects fix their templates.

The rest of the thesis is organized into the following sections. In Chapter 2, we provide a brief overview of issue templates in GitHub. In Chapter 3, we present the related work to our study. In Chapter 4, we describe how we conduct the study. In Chapter 5, we present the results and in Chapter 6 we discuss them. In Chapter 7, we discuss the threats to the validity of our study. Finally, in Chapter 8, we present our concluding remarks and recommendations.

Chapter 2

Background on GitHub Issue Templates

GitHub is an online platform that enables developers to store and manage their software development projects. It utilizes the Git version control system and offers a variety of functionalities, such as collaboration tools, code review, and issue tracking. The platform was introduced in 2008 and has since become one of the most widely used code-hosting platforms globally.

GitHub Issues is a feature within the platform that allows users to keep track of bugs, feature requests, and other tasks related to software development projects. It offers tools for issue tracking, including labels, milestones, and assignees, which aid in organizing and prioritizing tasks. GitHub Issues is commonly used by software development teams in both open-source and commercial projects. It serves as a powerful tool for managing the development process, and many organizations rely on it to keep track of bugs, feature requests, and other tasks.

An issue template is a pre-defined set of fields with descriptions that can be used to create an issue. It allows teams to set up a consistent structure for the issues they raise and to provide guidance on the information that should be included. These templates can standardize the process of raising bugs, feature

requests, and other tasks related to software development projects.

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 2.1. 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 2.2. Besides the templates, project maintainers restrict users from creating issues for questions and security problems, and they prefer forwarding the users to external pages.

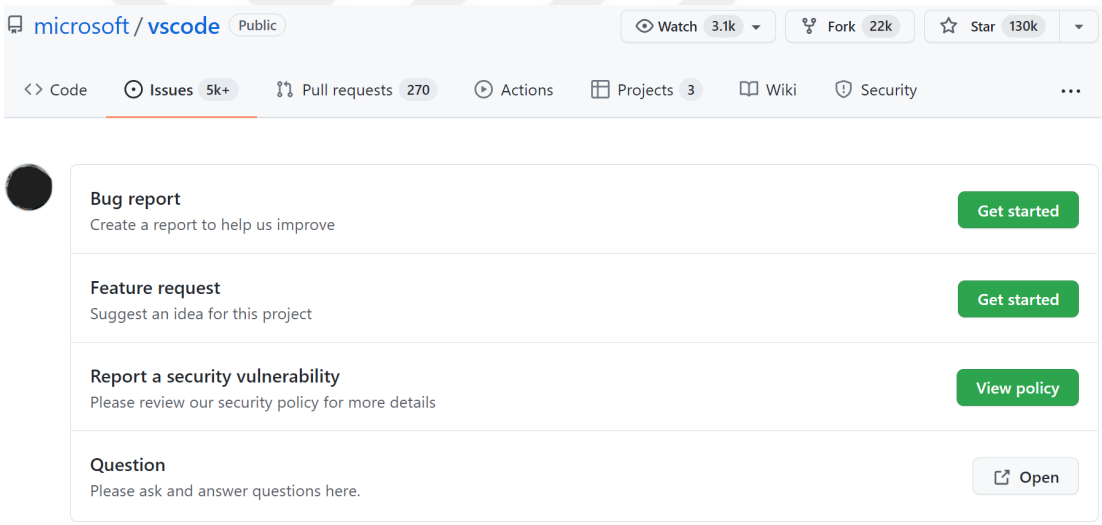


Figure 2.1: Issue template chooser menu of Microsoft VS Code

The details of the issue templates and how to use them are described in GitHub documentation [6]. 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

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

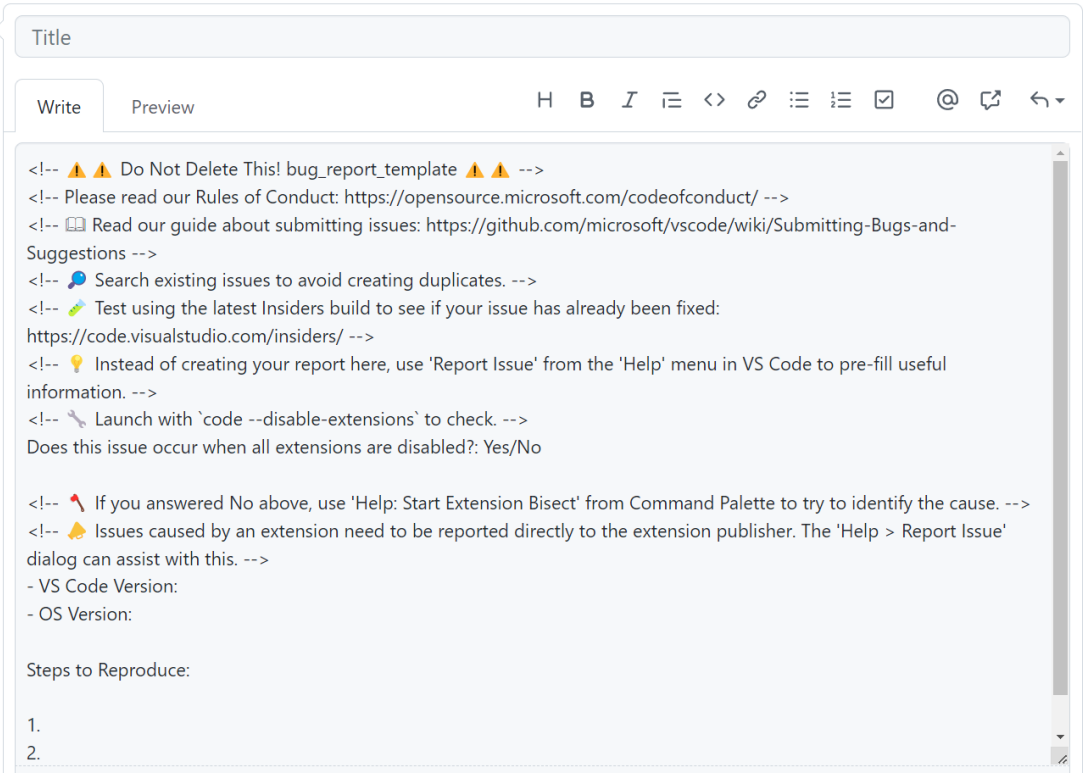


Figure 2.2: Bug reporting issue template of Microsoft VS Code

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]

[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`

- 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.

Issue templates have become a popular tool among software development teams as they help to streamline the process of raising and managing issues. They can help to ensure that all issues raised include the necessary information, reducing the need for follow-up questions and clarifications. Additionally, they can help to improve the quality of issues raised, making it easier for developers to understand and address the issues.

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 [5]. Then in 2018, GitHub introduced multiple issue templates [7]. 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 thesis, 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] 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/

Chapter 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 [8]. GitHub Issues is also used to study issue tracking systems as a built-in feature of GitHub. For example, Kallis et al. [9] developed a GitHub app to predict the issue types automatically. Panichella et al. [10] conducted a similar study to identify *won't fix* issues on GitHub. Likewise, Izadi et al. [11] proposed a method to predict the objective and priority of the issues using feature engineering methods and text classifiers.

Chen et al. [12] 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. [13] 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. [2]. 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. [3] 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.

Qamar et al. [14] defines a taxonomy of anti-patterns in bug tracking process. They define twelve process smells and three of them *(missing priority, missing severity and missing environment information)* are related to the lack of essential attributes in bug reports. Furthermore, they assess the negative effects of such smells by quantitatively analyzing the time to resolution and the number of reopenings.

Chaparro et al. [15] 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 [16] 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.

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. [17] 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. [18] used mixed methods to empirically analyze the issue and pull request templates and it aims to explore contents, impacts, and perceptions. 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. [19] examined the maintenance levels of open-source projects to understand their inactivity. One of the research questions aims to compare the adaption 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. [20]. Their results suggest that using templated requirements specification inhibits creativity by reducing critical evaluation and critical thinking.

Chapter 4

Methodology

In this section, we describe how we conducted the study. The methodology can be summarized in Figure 4.1.

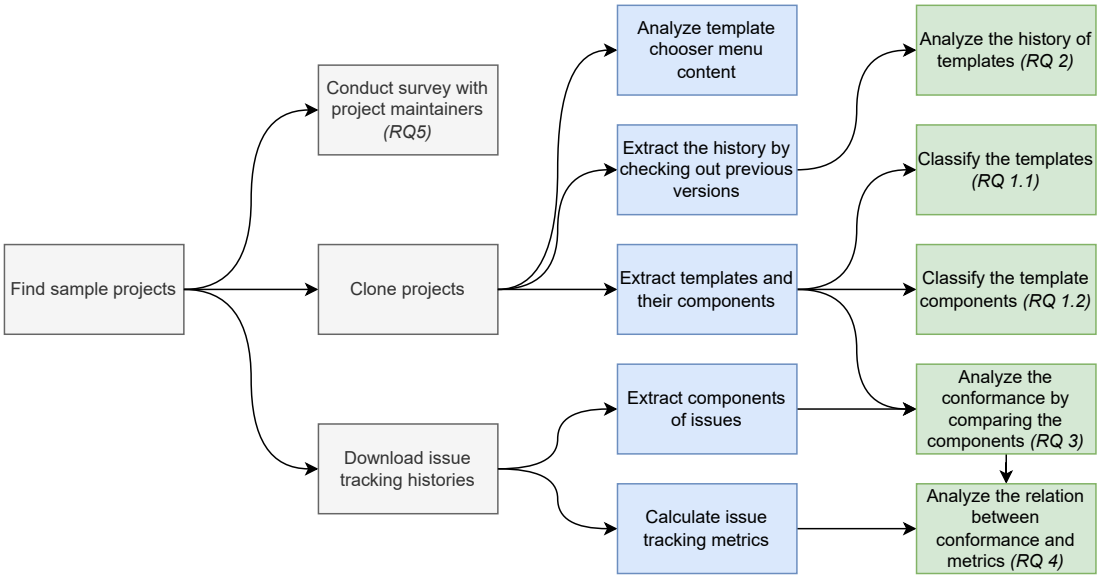


Figure 4.1: The overview of the study

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. [21]. A list of the sampled projects is available in Appendix A.

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[1] 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 [2].

We collected all the issues of these projects with the help of the Perceval tool [22]. The data collection step led to 100 repositories, 1,916,057 issues, and 350 issue templates.

[1] `https://github.com/type-challenges/type-challenges`
[2] `https://esulun.com/msc-thesis-replication-package`

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 thesis, 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 contributors to the templates.

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 investigated by the author and the advisor of the thesis. After they separately determined their categories, the author, the advisor and an undergraduate research volunteer compared the results and resolved the conflicts in a meeting. Finally, we came out with four categories with corresponding keywords:

- *Bug*: bug, defect, error, problem, issue, failure, crash, inconsistent, unexpected, regression, incident
- *Feature Request*: feature, enhancement, improvement, user story, epic, propose, proposal, roadmap
- *Task*: task
- *Question*: question, support, help

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.

Based on these keywords, we created a script for the automatic classification of template categories that were shared in our replication package [3].

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*.

[3] `https://esulun.com/msc-thesis-replication-package`

We extracted 1,458 headings from 350 templates in total. To classify components, the author and the advisor of the thesis independently went over the 1,458 headings and classified them. During their manual analysis, they listed the keywords associated with each component category. In a separate meeting, the author, the advisor and an undergraduate research volunteer 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, Information Text, 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 2.1). 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. The author and an undergraduate research volunteer 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. For the analysis, we first run a Git command 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 are 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 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 [23] 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.

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.

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

Chapter 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[1] uses organization-level templates, 1 project[2] 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[3] 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 5.1.

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).

[1] https://github.com/atom/atom
[2] https://github.com/vuetifyjs/vuetify
[3] https://github.com/sympy/sympy

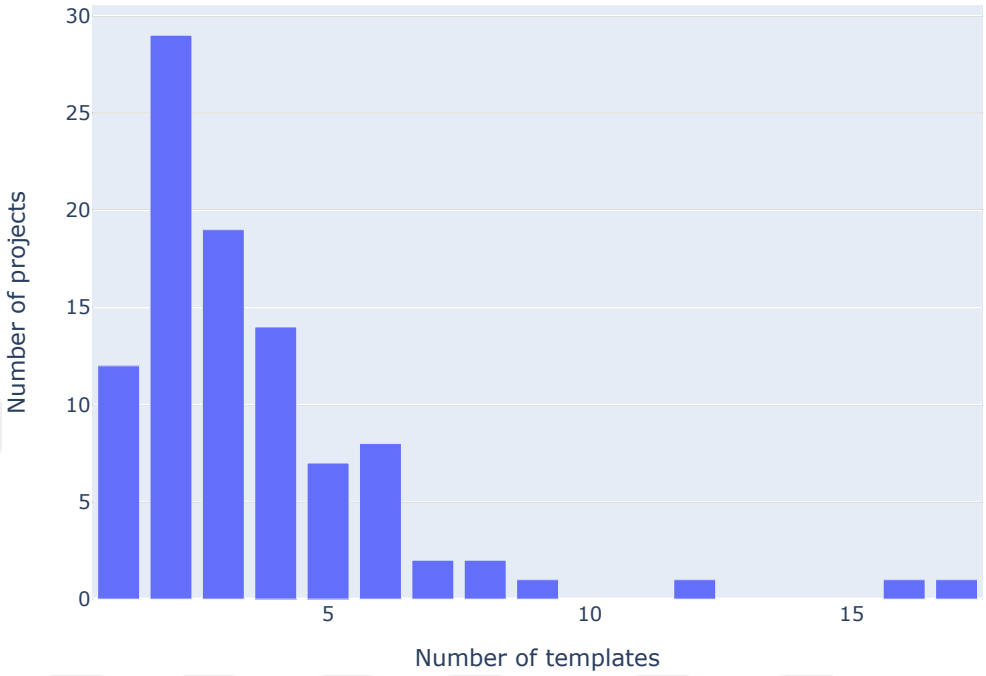


Figure 5.1: The distribution of the number of templates per project

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 5.2.

Figure 5.3 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 5.1. 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.

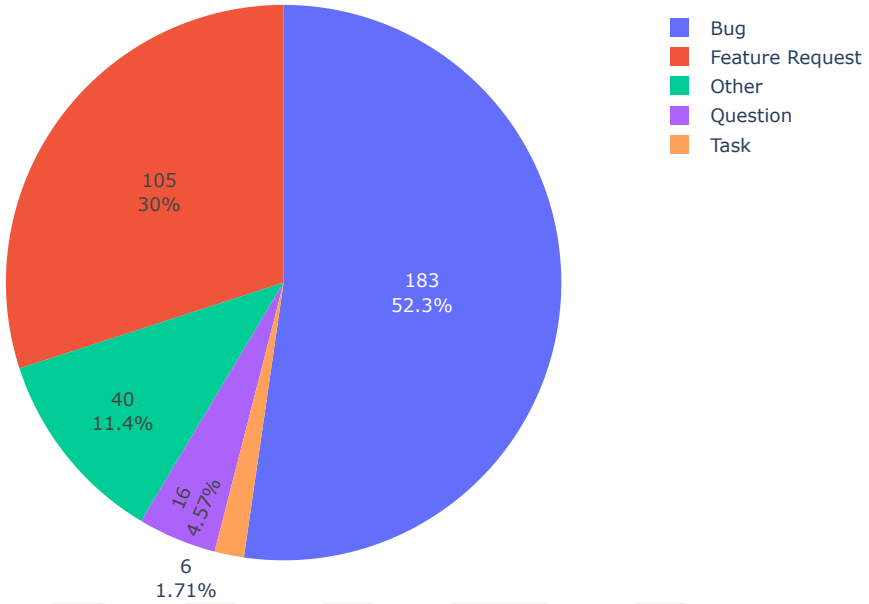


Figure 5.2: The distribution of the template categories

Table 5.1: 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

5.1.2 RQ1.2 Template Components

The result of the analysis of the template content and the used components is shown in Table 5.2. 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. The other category consists of not frequent items (such as priority and severity) or components that are only applicable to that specific project's domain.

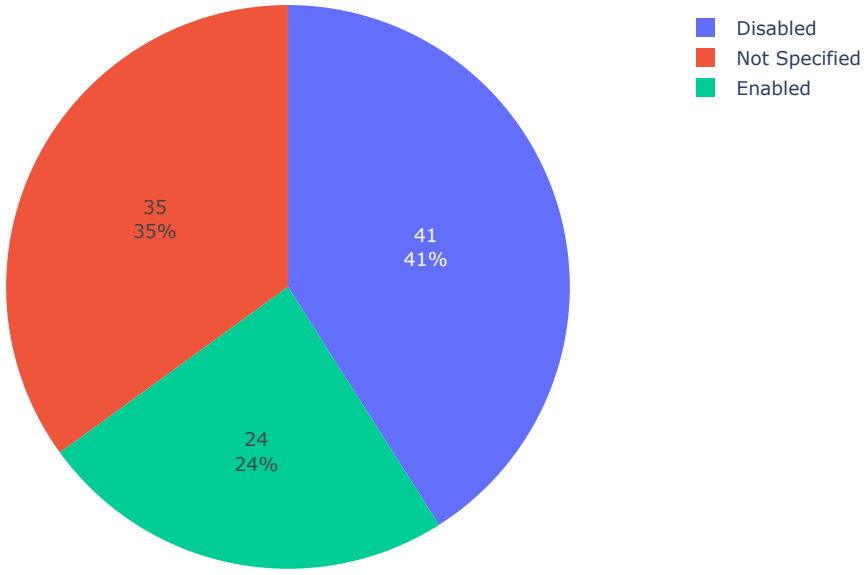


Figure 5.3: Project's blank issue allowance distribution *(If not specified, it is enabled by default)*

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 5.4 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 5.5. A trend of upgrading the template type is observed in this figure.

Projects tend to use new generation issue templates over time. As Figure 5.4 and 5.5 suggest, more projects increasingly prefer issue forms that are more strict but still configurable.

Table 5.2: 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

5.3 RQ3: Conformance of 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 12% 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 5.7 presents the conformance analysis results. A visual summary of the distribution of 1,916,057 issues is given in Figure 5.6.

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 filters led to 1,661,788 issues. 209,654 issues have zero conformance to templates, while 394,285 issues have non-zero conformance.

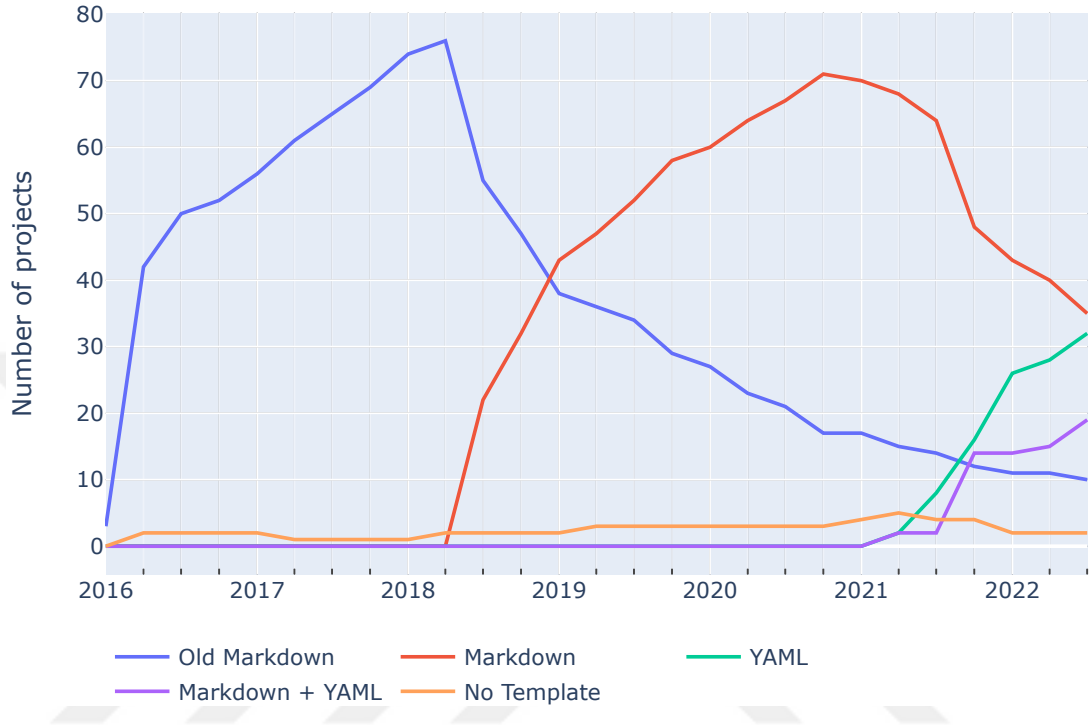


Figure 5.4: The historical evolution of issue template usage

We have nine hypotheses for three metrics *(TTR (time to resolution), 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 5.3, and the statistical summary of the comparison is given in Table 5.4.

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.

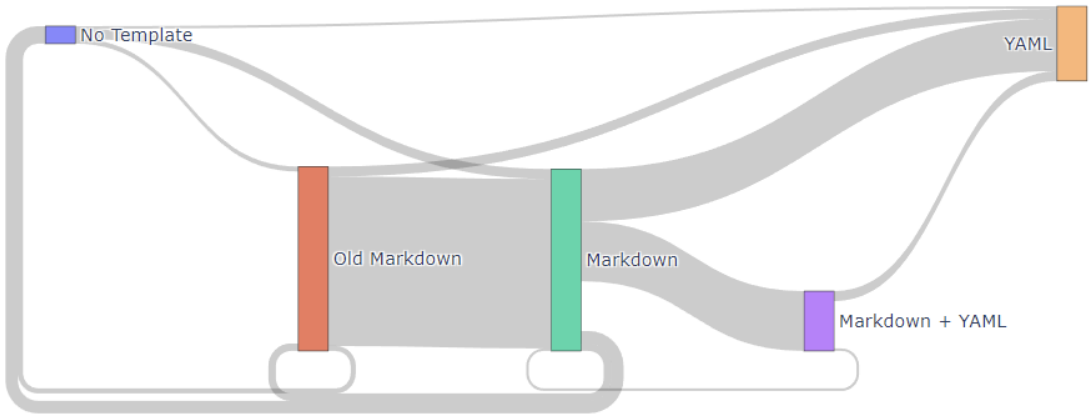


Figure 5.5: Transition between different template types

Table 5.3: 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 5.9. 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,

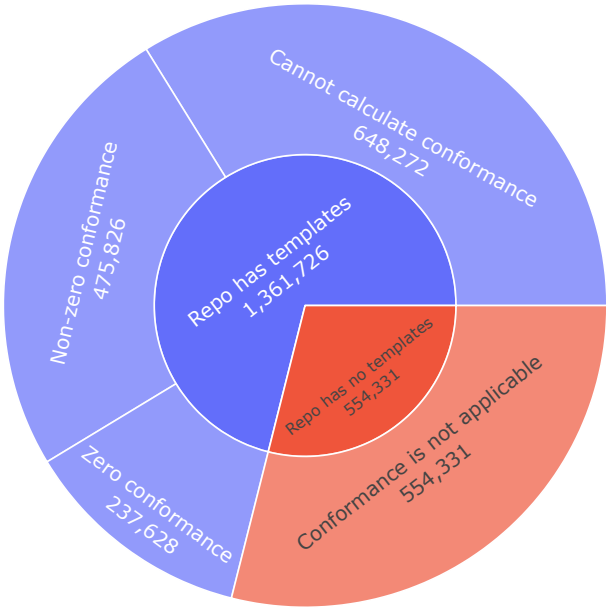


Figure 5.6: Distribution of the issues. The inner circle shows whether projects use issue templates when an issue is created, while the outer circle shows the conformance category.

- 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 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

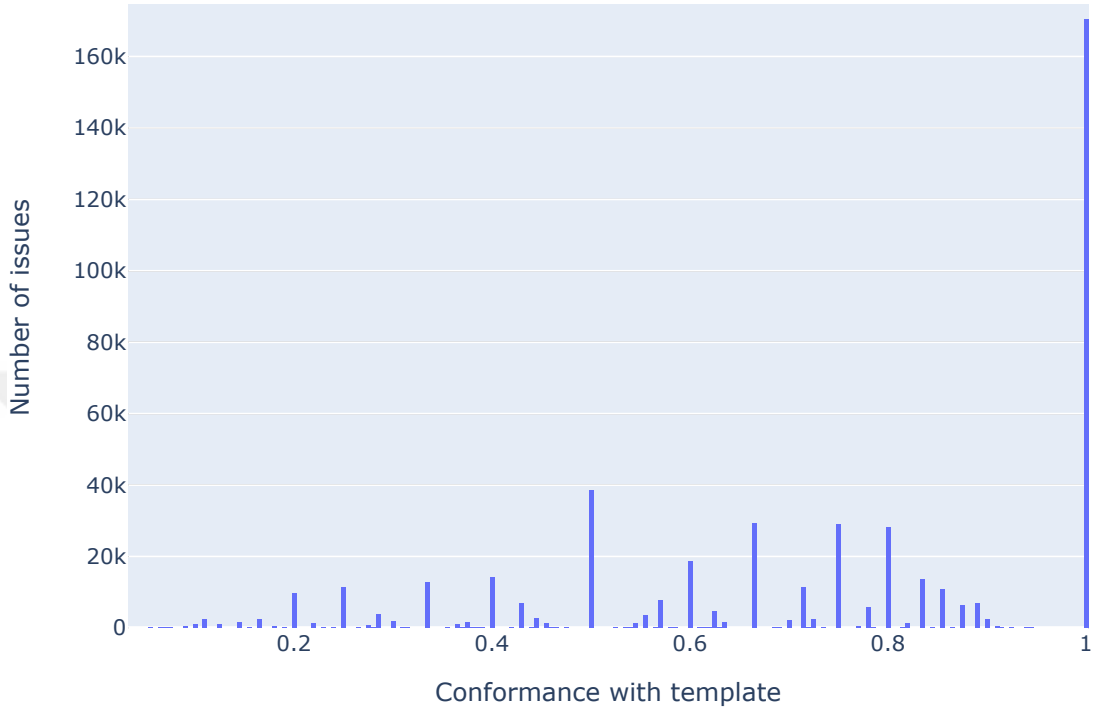


Figure 5.7: The conformance of issues (excluding zero conformance)

> 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.

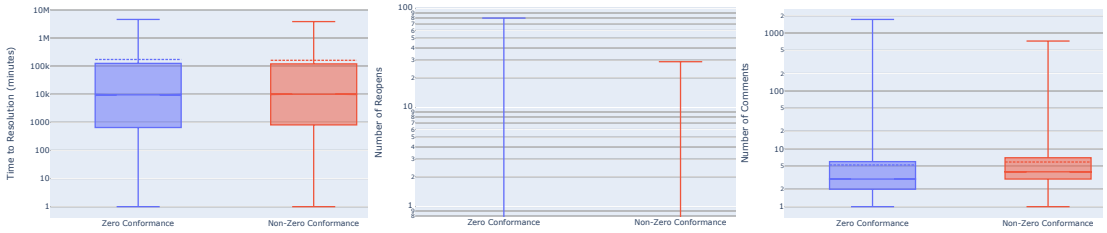
On the other hand, another project maintainer summarizes the challenges related to issue templates as follows:

> Templates can accrue a lot of cruft, be overly specific/rigid, and worst-case scenario dissuades 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.

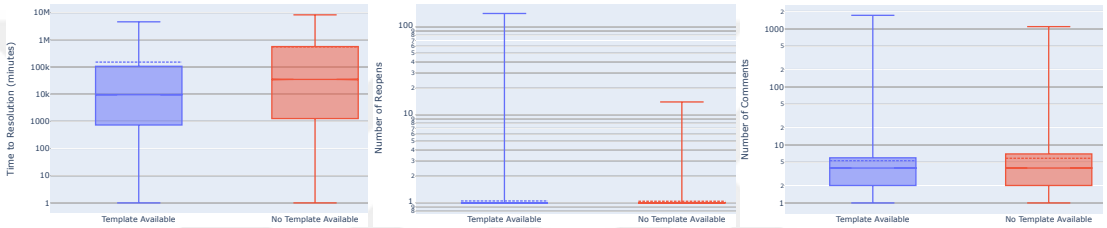
Table 5.4: Summary of the statistical analysis

		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

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.



(a) Comparison by the conformance to the templates



(b) Comparison of whether a project has a template or not

Figure 5.8: Three issue tracking metrics compared by conformance values and the existence of a template

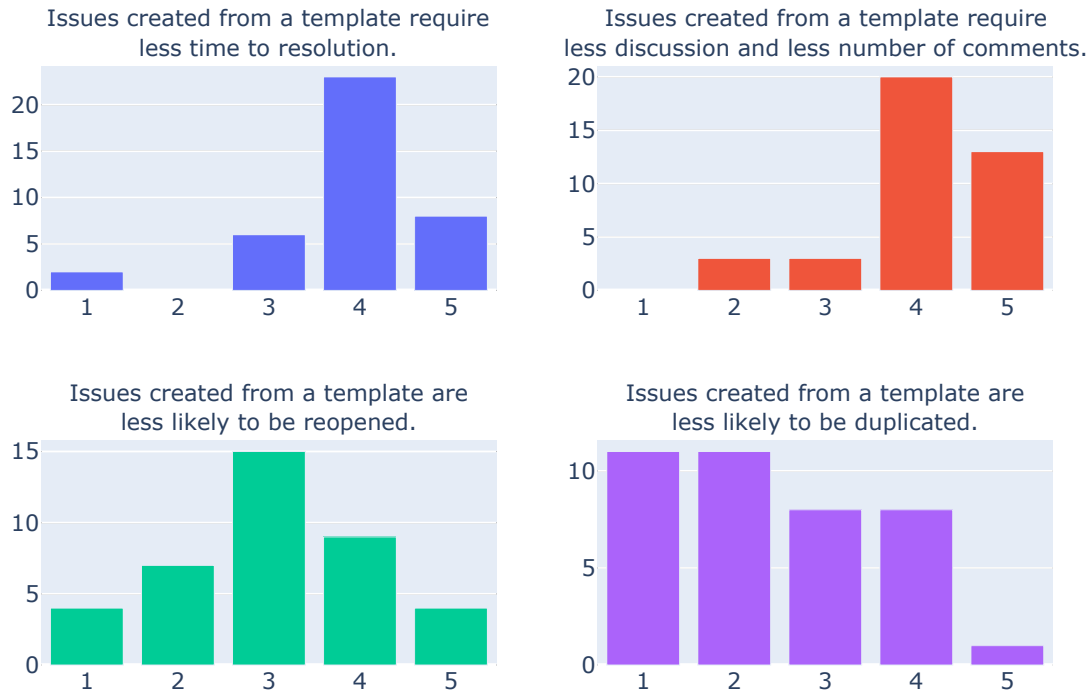


Figure 5.9: Survey responses. 1: Strongly disagree 5: Strongly agree

Chapter 6

Discussion

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

The analysis shows that issue templates are widely used (99% of the 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 5.2). 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 positively impacts reducing 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. The results of the survey also support this finding. The majority of the respondents agree about the benefits on the time to resolution and the length of discussions. Also, the YAML-based templates are the first choice of project maintainers when their issue tracking system preference is asked.

6.1 Suggestions for Project Maintainers

6.1.1 Use issue templates

Issue templates help project maintainers to organize the issue tracking systems. 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 5.2 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.

Additionally, issue forms (YAML-based templates) are more structured and organized than Markdown-based templates. Therefore, they can be helpful if a more strict workflow is needed.

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 5.2 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 6.1 for an example). Therefore, we suggest the contributors conform to the templates as much as possible, especially for the required fields.

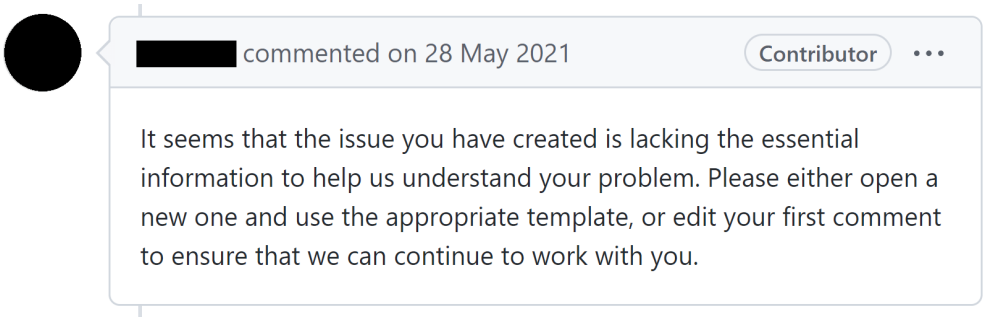


Figure 6.1: A comment written by a project maintainer under an issue created with missing details

6.3 Suggestions for GitHub

During the analysis, we noticed that some templates written in YAML had syntax issues [1]. 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[2] [3], [4], [5], [6]. 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.

[1] `https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/common-validation-errors-when-creating-issue-forms`
[2] `https://github.com/querydsl/querydsl/pull/3287`
[3] `https://github.com/ant-design/ant-design/pull/36518`
[4] `https://github.com/elementor/elementor/pull/19154`
[5] `https://github.com/saltstack/salt/pull/62330`
[6] `https://github.com/sourcegraph/sourcegraph/pull/39488`

Interestingly, two projects developed external issue forms[7], [8] 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] https://issues.vuetifyjs.com/
[8] https://new-issue.ant.design/

Chapter 7

Threats to Validity

In the following, we discuss the main 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.

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, a project may 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 be the result of recruiting more developers to the projects. Finally, with the use of 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 more 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.

Chapter 8

Conclusion and Future Work

GitHub Issues offers a lightweight issue tracking solution for open-source projects. In the original design, anyone can create an issue by filling in only a description. This flexibility was helpful for contributors since it would be faster to create new issues. However, in the long run, this can lead to maintainability problems for open-source maintainers, especially in large-scale projects. To mitigate this problem, GitHub developed issue templates feature that helps project maintainers standardize their issue reporting process. In this study, we analyzed the current status of issue template usage by mining open-source projects. The summary of our results is as follows;

In RQ1, we summarized the current status of the issue templates. The result shows that templates are widely started to be used in sampled large-scale projects (99 of 100 projects). We also presented the set of common components across different projects.

In RQ2, we analyzed the historical evolution and found that the projects tend to use the new features of the templates, and the transition can be observed in Figure 5.4.

In RQ3, we evaluated the conformance of the contributors to the templates. We found that 12% of the issues do not conform at all and the mean conformance

for the rest is 0.71.

In RQ4, we evaluated the impact of the templates in detail. We observed a significant difference (p-value 0.00, effect size 0.59) between the *time to resolution* of issues opened when at least one template existed in a project compared to a project that did not have a template at the time. Similarly, the positive impact of YAML-based templates on time to resolution (p-value 0.00, effect size 0.37), the number of comments and the number of reopenings is statistically significant.

Finally in RQ5, we conducted a survey and presented the benefits and challenges of templates from the perspective of open-source software maintainers who used them recently. In the survey, 85% of respondents stated they agree with the fact that issues created from a template require less discussion and less number of comments. Also, 41% of them prefer GitHub Issues with YAML-based templates when compared with other alternatives.

To the best of our knowledge, this study is the first to analyze only issue templates. We plan to mine more GitHub repositories in the future to get a more complete picture related to the issue templates. Further studies may also focus on tool development to address the challenges of the templates.

Bibliography

[1] R. Kallis, A. D. Sorbo, G. Canfora, and S. Panichella, "Ticket tagger: Machine learning driven issue classification," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 406–409, IEEE, 9 2019.

[2] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?," *IEEE Transactions on Software Engineering*, vol. 36, pp. 618–643, 9 2010.

[3] M. Soltani, F. Hermans, and T. Bäck, "The significance of bug report elements," *Empirical Software Engineering*, vol. 25, pp. 5255–5294, 11 2020.

[4] M. of Open-Source Projects, "Dear github," 2016.

[5] B. Bleikamp, "Issue and pull request templates," 2 2016.

[6] GitHub, "About issue and pull request templates."

[7] B. Clark, "Multiple issue and pull request templates," 1 2018.

[8] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Software Engineering*, vol. 21, pp. 2035–2071, 10 2016.

[9] R. Kallis, A. D. Sorbo, G. Canfora, and S. Panichella, "Predicting issue types on github," *Science of Computer Programming*, vol. 205, p. 102598, 5 2021.

[10] S. Panichella, G. Canfora, and A. D. Sorbo, "Won't we fix this issue? qualitative characterization and automated identification of wontfix issues on github," *Information and Software Technology*, vol. 139, p. 106665, 11 2021.

[11] M. Izadi, K. Akbari, and A. Heydarnoori, "Predicting the objective and priority of issue reports in software repositories," *Empirical Software Engineering*, vol. 27, p. 50, 3 2022.

[12] S. Chen, X. Xie, B. Yin, Y. Ji, L. Chen, and B. Xu, "Stay professional and efficient: automatically generate titles for your bug reports," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 385–397, ACM, 12 2020.

[13] T. D. Sasso, A. Mocci, and M. Lanza, "What makes a satisficing bug report?," in *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 164–174, IEEE, 8 2016.

[14] K. A. Qamar, E. Sülün, and E. Tüzün, "Taxonomy of bug tracking process smells: Perceptions of practitioners and an empirical analysis," *Information and Software Technology*, vol. 150, p. 106972, 10 2022.

[15] O. Chaparro, C. Bernal-Cárdenas, J. Lu, K. Moran, A. Marcus, M. D. Penta, D. Poshyvanyk, and V. Ng, "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*, pp. 86–96, ACM, 8 2019.

[16] Y. Song and O. Chaparro, "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*, pp. 1551–1555, ACM, 11 2020.

[17] M. Zhang, H. Liu, C. Chen, Y. Liu, and S. Bai, "Consistent or not? an investigation of using pull request template in github," *Information and Software Technology*, vol. 144, p. 106797, 4 2022.

[18] Z. Li, Y. Yu, T. Wang, Y. Lei, Y. Wang, and H. Wang, "To follow or not to follow: Understanding issue/pull-request templates on github," *IEEE Transactions on Software Engineering*, pp. 1–16, 2022.

[19] J. Coelho, M. T. Valente, L. Milen, and L. L. Silva, "Is this github project maintained? measuring the level of maintenance activity of open-source projects," *Information and Software Technology*, vol. 122, p. 106274, 6 2020.

[20] R. Mohanani, P. Ralph, B. Turhan, and V. Mandic, "How templated requirements specifications inhibit creativity in software engineering," *IEEE Transactions on Software Engineering*, pp. 1–1, 2021.

[21] O. Dabic, E. Aghajani, and G. Bavota, "Sampling projects in github for msr studies," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pp. 560–564, IEEE, 5 2021.

[22] S. Dueñas, V. Cosentino, J. M. Gonzalez-Barahona, A. del Castillo San Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, and A. P. García-Plaza, "Grimoirelab: A toolset for software development analytics," *PeerJ Computer Science*, vol. 7, p. e601, 7 2021.

[23] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, pp. 50–60, 3 1947.

Appendix A

List of Projects

	Organization/Project	Number of Issues	Main Language
0	ampproject/amphtml	13836	JavaScript
1	angular/angular	23451	TypeScript
2	angular/angular-cli	13754	TypeScript
3	angular/components	12845	TypeScript
4	ansible/ansible	30539	Python
5	ant-design/ant-design	22796	TypeScript
6	apache/echarts	15857	TypeScript
7	appium/appium	12531	JavaScript
8	apple/swift	12607	C++
9	atom/atom	17028	JavaScript
10	automattic/wp-calypso	18265	JavaScript
11	aws/aws-cdk	10102	TypeScript
12	azure/azure-cli	13106	Python
13	ballerina-platform/ballerina-lang	14963	Java
14	brave/brave-browser	21133	JavaScript
15	ckeditor/ckeditor5	10249	JavaScript
16	cleverraven/cataclysm-dda	20382	C++
17	clickhouse/clickhouse	10150	C++
18	cypress-io/cypress	10732	JavaScript

19	dbeaver/dbeaver	12853	Java
20	dotnet/aspnetcore	25823	C#
21	dotnet/aspnetcore.docs	12198	C#
22	dotnet/efcore	17898	C#
23	dotnet/roslyn	28675	C#
24	dotnet/runtime	44941	C#
25	dotnet/sdk	10559	C#
26	eclipse/che	13993	TypeScript
27	eclipse/sumo	10763	C++
28	elastic/elasticsearch	30251	Java
29	elastic/kibana	36400	TypeScript
30	electron/electron	17140	C++
31	elementor/elementor	12833	PHP
32	facebook/react	11046	JavaScript
33	facebook/react-native	22733	JavaScript
34	fastlane/fastlane	12139	Ruby
35	fortawesome/font-awesome	18351	JavaScript
36	frappe/erpnext	12909	Python
37	freecodecamp/freecodecamp	15247	JavaScript
38	gatsbyjs/gatsby	13483	JavaScript
39	godotengine/godot	33323	C++
40	golang/go	47858	Go
41	grafana/grafana	22678	TypeScript
42	grpc/grpc	10290	C++
43	hashicorp/terraform	18677	Go
44	hashicorp/terraform-provider-aws	10239	Go
45	highcharts/highcharts	12997	TypeScript
46	influxdata/influxdb	12477	Go
47	ionic-team/ionic-framework	20128	TypeScript
48	iterate-ch/cyberduck	10948	Java
49	jackett/jackett	10584	C#
50	joomla/joomla-cms	13796	PHP
51	jooq/jooq	13505	Java

52	keras-team/keras	11362	Python
53	laravel/framework	15657	PHP
54	magento/magento2	22059	PHP
55	marlinfirmware/marlin	13255	C++
56	matomo-org/matomo	12340	PHP
57	microsoft/powertoys	15512	C#
58	microsoft/terminal	10107	C++
59	microsoft/typescript	32398	TypeScript
60	microsoft/vscode	117484	TypeScript
61	moby/moby	21436	Go
62	mozilla-mobile/fenix	16256	Kotlin
63	mrdoob/three.js	11174	JavaScript
64	nextcloud/server	12530	PHP
65	numpy/numpy	10597	Python
66	odoo/odoo	14067	JavaScript
67	owncloud/core	19006	PHP
68	paddlepaddle/paddle	14977	C++
69	pandas-dev/pandas	20022	Python
70	phpmyadmin/phpmyadmin	13327	PHP
71	powershell/powershell	10238	C#
72	prestashop/prestashop	12119	PHP
73	pytorch/pytorch	18622	C++
74	qbittorrent/qbittorrent	13415	C++
75	qgis/qgis	27767	C++
76	quarkusio/quarkus	11125	Java
77	rails/rails	15004	Ruby
78	rancher/rancher	27861	Go
79	rocketchat/rocket.chat	14345	TypeScript
80	saltstack/salt	24606	Python
81	sourcegraph/sourcegraph	10731	Go
82	spring-projects/spring-boot	26095	Java
83	spring-projects/spring-framework	22192	Java
84	spyder-ide/spyder	15040	Python

85	symfony/symfony	17397	PHP
86	sympy/sympy	11856	Python
87	telegramdesktop/tdesktop	11306	C++
88	tensorflow/tensorflow	32030	C++
89	trinitycore/trinitycore	21335	C++
90	twbs/bootstrap	21178	JavaScript
91	vector-im/element-web	20180	TypeScript
92	vuetifyjs/vuetify	10931	TypeScript
93	woocommerce/woocommerce	19741	PHP
94	wordpress/gutenberg	17755	JavaScript
95	yiisoft/yii2	11241	PHP
96	yoast/wordpress-seo	11174	JavaScript
97	ytdl-org/youtube-dl	25065	Python
98	yugabyte/yugabyte-db	10818	C
99	zephyrproject-rtos/zephyr	15050	C

Appendix B

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.