Time-to-close: an analysis of GitHub Issue/Pull Request Templates

Yogeshvar Senthilkumar
The University of Adelaide
Adelaide, Australia
yogeshvar.senthilkumar@student.adelaide.edu.au

ABSTRACT

This research paper explores the usage and characteristics of GitHub templates in open-source projects. We examined a dataset of 538 repositories with both Issue and Pull Request templates using the GitHub API. We discovered some valuable insights for effective communication among the community. On average, repositories had $\approx$ 2.68 and 1 issue and pull request templates respectively. Contents of Templates contained labels such as welcoming contributors, project guidelines, and additional data. We also investigated variations and differences in templates across different languages. Javascript and Go repositories exhibited distinct trends, whereas Assembly language despite fewer repositories, had the highest average number of issue templates. This research analysis gives us an understanding of templates for better effective communication. Future directions include evaluating the impact of templates, template customization, and understanding the coherence between template usage and community engagement.

1 INTRODUCTION

Over the years, GitHub's issue and pull request templates have gained widespread adoption among software projects, and act as a provider to standardize information from contributors [10] In this work, to achieve a better understanding of the template feature in GitHub, we conduct a mixed-methods empirical study of issue/PR templates, by answering the following questions: **RQ1:** *What are the common types of GitHub issue and pull request templates used in the OSS Projects?* **RQ2:** *What are the contents/taxonomy of these templates?* **RQ3:** *How do the contents of GitHub templates differ across programming languages?* The insights obtained from this study can inform the best practices and provide guidelines for the community to understand the template feature, and leverage the potential of GitHub's template feature to streamline workflows, better issue resolution, and perform effective collaboration among the contributors [14] [15]. This understanding will ultimately contribute to the development of more efficient and productive open-source software projects [16].

2 MOTIVATION

Github, a widely used platform for hosting, version controlling, and collaborating on Open source software projects (OSS) plays a crucial role in facilitating the distributed and collaborative agile environment for building software solutions [10]. Acts as a central hub where programmers and project managers, stakeholders can engage, propose, discuss, report through issues and pull requests [2]. The global nature of OSS development often means that developers are geographically dispersed, making effective communication for successful collaboration on GitHub [19]. Within this constraint, everyone in the community majorly relies on GitHub's features to interact and build things together [14]. Although, the description of issues and pull request contributions can be challenging [2]. Various issues and pull requests are stagnated because of the inadequate description, furthermore, the communication between the user and maintainer for requesting more information makes the issue disappear or less important over time [3]. GitHub analyzed the shortcomings of the informational quality of issues and pull requests, and introduced the *Issue/Pull Request Templates*, which will ensure to maintainers have a proper informational quality as they can customize the template. [12] The template will be prompted to the contributors who are expected to include the customized information requested by the maintainers. Although, this feature has been widely used by various popular and trending projects on GitHub. An analysis of templates showcases the potential for improving collaboration in the OSS community.

3 BACKGROUND

In the majority of GitHub projects, both issue discussion and code changes proposal/discussions are conducted primarily through text-based communication. [9] For an issue discussion, the users who encountered an issue will report the issue in the form of an issue in GitHub under the repository. An issue will be composed of an issue description and unexpected behavior of the software. While a pull request would be a code change proposal by a developer which could potentially be solving an issue or introduce a new feature to the software. Both the crucial part of better software is initiated communication, the maintainers and the users will move on back-and-forth to collaboratively improve the software. [19] But the description of an issue and pull request submission is difficult. Without a proper description, the maintainers cannot understand the unexpected behavior reported by a user. Similarly, if a code changes proposal does not have a detailed description. It would be a tedious task for the maintainer to review/understand the code without a proper description of the proposal. The analysis of these templates can offer insights that can be useful in future projects for streamlining effective communication practices.

4 METHODOLOGY

4.1 Research Questions

RQ1: *What are the common types of GitHub issue and pull request templates used in the OSS Projects?* We have created our dataset using GitHub API to obtain 538 repositories which contain both *Pull Request Templates* and *Issue templates*. We first analyzed the *average number of issue templates* was around **2.6877** whereas for *pull request templates* it was $\approx$ 1. Furthermore, we have noticed **82.1%** of

repositories have more than one issue file template and only **3.71%** for pull request templates. **RQ2:** *What are the contents/taxonomy of these templates?* With the repository data, we qualitatively classify the contents of the template files, significantly exploring the elements of templates among various projects. We concluded that templates have a common structure such as greeting contributions, and demonstration of project guidelines, descriptive informational questions to collect meaningful data. **RQ3:** *How do the contents of GitHub templates differ across programming languages?* We have a total of 25 unique languages from the 436 repositories, a noticeable difference between the *Javascript* and *Go* languages repositories have an average number of issue templates as ***2.52*** and ***3.18*** respectively. On the other hand, *Assembly* languages has only 2 repositories but the average number of issue file templates is ***4***, greater than any other 25 unique languages in the dataset. We will also explore valuable insights into in-differences between templates based on the primary language.

4.2 Data Collection

For this empirical study, we have collected the necessary data by combining several steps. Firstly, we explored different open-source software that enables us to collect or visualize data. Noticeably, we have tried *MergeStat* and it was promising. Although, a deeper understanding of a single repository was the major objective of the software. So we decided to go for a holistic approach as it is evident *GitHub* provides *Application Programming Interface* to interact with GitHub. For communicating with GitHub, we need to obtain a GitHub token that will grant access to read, and write repository information, commits, and pull requests. With many versions of API available, we utilized *GraphQL* endpoint to retrieve the required data. One of the key elements in choosing this version over the classic *REST* API endpoint, was the exposure to the templates on *GraphQL*. Under the nodes of *GraphQL*, we have *issueTemplates* and *pullRequestTemplates* which would be handy for our research [7].

```
{
    search(query: "QUERY",
           type: REPOSITORY,
           first: 100) {
        pageInfo {
            hasNextPage
            endCursor
        }
        nodes {
            ... on Repository {
                nameWithOwner
            stargazerCount
                primaryLanguage {
                    name
                }
                issueTemplates {
                    filename
                    body
                    name
                    title
                }
                owner {
                    login
                }
                pullRequestTemplates {
                    filename
                    body
                }
                hasWikiEnabled
                description
            }
        }
    }
}
```

Listing 1: Query to retrieve repository information

With the help of the query mentioned above, we accumulate the repositories with a significant number of stars (greater than 1000) and also in the popular languages such as *Python, Java, Javascript, C++, C, Go, Ruby, PHP, Rust, Swift* and *Typescript*. From the response, we utilise the `hasNextPage` and `endCursor` parameters to iterate through the paginated results. For each repository, we consider only if the *issue templates* and *pull request templates* are present in the repository. The accumulated data will be converted into a ***C**ommo **S**eperated **V**alues* to store the specific repository information.

On the *pull request template files* and *issue template files* [6] from API was given as a list, so to facilitate analysis we convert it to multiple file names separated by commas. The resulting dataset comprised ***538* repositories**, a total of ***6574* individual issue files** and ***2601* pull request templates.**

To ensure the reproducibility of the research results, the `python` script used for data creation is publicly available for anyone to reproduce on *GitHub* (https://github.com/yogeshvar/GH-Templates. Detailed technical documentation is provided to reproduce the data collection process for this research project, this enables verification of the findings and promotes transparency in research methodology.

- ***Obtain a Github Token:*** For communication with Github's Application programming interface, we would require a *Github* token which should grant read access to repositories, commits, files, and pull requests.
- ***GraphQL Endpoint:*** With the *Github Token*, we can make API calls to the GraphQL endpoint to retrieve the data. Specify the desired filters, such as specific programming languages, in descending or ascending order based on *created date.*
- ***Fetch & Process Data:*** The retrieved repositories in batches are accumulated into a single entity. Store the collected data in a suitable format, such as *TSV* or *CSV* files.

By following the given steps, researchers can reproduce the data collection process and obtain a dataset similar to this research, which could be essential for replacing the results and facilitating further analysis.

4.3 Data Analysis

4.3.1 ***What are the common types of GitHub issue and pull request templates used in the OSS Projects?*** For answering

the RQ1, we conducted an analysis of the data we collected. Our major objective was to focus on several key aspects such as the identification of the most common issue and pull request templates, the percentage of issue and pull request templates in the repository, and the average number of sections in the commonly used templates.

Top 3 Issue template	
pull_request_template	90.31%
bugfix	1.11%
translation_checklist	0.58%

Top 3 Pull Request Template	
bug_report	22.18%
feature_report	20.61%
question	4.02%

Table 1: Top 3 issue & pull request template

We investigate the *average number of issue templates* and *pull request templates*, and we concluded the ***bug_report.md*** and ***pull_request_template.md*** are the most commonly used issue and pull request templates. Additionally to understand the popular common templates, our results showcase every repository that adopts the templates has two generic templates. One for the *issue_template* and other for the *pull_request_template*. We identify the repository code owners use *bug_report.md* as the common template which takes us 3 sections on average collecting relative information about the issue or the issue experienced by the user.

Most Common Sections in Issue Templates	
Expected Behaviour	71.97%
Steps to Reproduce	59.83%
Environment	55.23%

Most Common Sections in PR Templates	
Description	22.85%
Checklist	22.27%
Summary	4.37%

Table 2: Most 3 common sections in the commonly used template

To have more insights into the sections of the template in the common templates, we adopted a classic natural language processing to understand the percentage of each section across the commonly used templates. Notably, sub-sections such as ***"Current/Expected Behaviour"***, ***"Steps to reproduce"*** and ***"Environment"*** were highly present and indicating the frequent usages in the widely used templates. On the contrary, sections such as ***"Description"***, ***"Checklist"*** and ***"Summary"*** take us a huge percentage in the pull request templates. These findings on the common templates present in the widely used templates give us insights into incorporating these templates and common sections for future contributions to OSS projects.

Overall, our data analysis for RQ1 identifies the top templates, percentage of common templates, average number of sections, and also the prevalent sections of the both issue and pull request templates, which can be served as a guideline for the developers and other stakeholders to provide a comprehensive and structured communication for the OSS projects. To conclude, the bug report template consisted of ***3.74*** sections on average while the pull request template has an average of ***2.55***. By answering our first research question, we can understand the necessary sections and importance

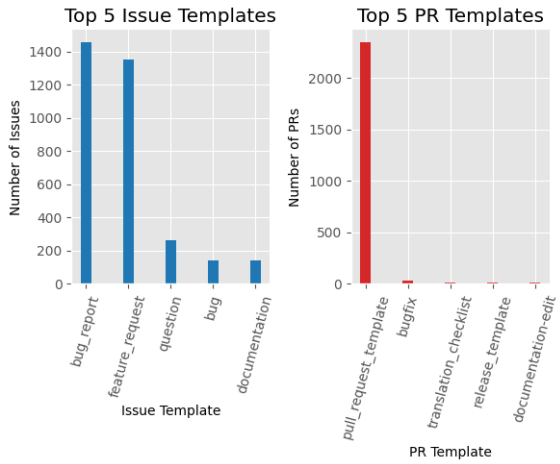

Figure 1: Top 5 issue/pull request templates

of the templates for streamlining issue and pull request management and also improve the performance of project communication and efficiency.

4.3.2 *What are the contents/taxonomy of these templates?*

In analyzing the contents of these templates, we need to understand the semantic meaning and context of the template on a sentence level to have an accurate and flexible classification. Unlike the traditional approaches such as pattern matching or regex matching can capture the variations on the template structures, but it would not a suitable methodology for the diverse and evolving nature of the GitHub Templates. Previous works on [9] did an open coding approach, taking a random sample of template files got classification. For this research, we have utilized all the repositories rather than sampling the template files. Then, all the categories are classified based on [9] but ignore a few taxonomies as we are focusing on the widely used templates. Although, we have come to a conclusion to adopt the higher-level categories from [9].

For understanding the semantic and contextual meaning of the templates, we approach a transformer-based language model trained on a large corpus of text data. ***Facebook/bart-large-mnli*** [8] model, a state-of-the-art-transformer fine-tuned on the Multi-Genre Natural Language Inference (MNLI) dataset, which can be useful to comprehend the semantics between sentences and perform classification tasks. A notable feature of the model is ***Zero-shot classification*** [17] which enables us to classify our templates as the model is already trained on a set of labeled examples for classification.

With [9] categorization, we have concluded to have the following categories for understanding the contents of the templates.

- ***Related Issue:*** This category will be useful for tracking and providing context and understanding to the issues related based on the issue/pull request.
- ***Additional Context:*** This type will give more background information and context for the contributors to understand the problem in a deeper level.

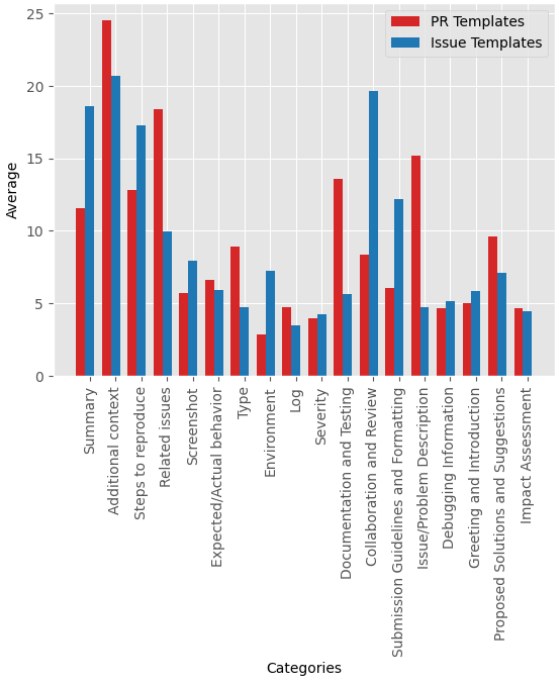

Figure 2: Distribution of Categories across PR/Issue Templates

- ***Steps of Reproduce:*** Highlighting the reproducibility of the work or an issue is encouraged by the code owners to resolve it.
- ***Expected/Actual Behaviour:*** Identifies the inconsistency and discrepancies in the repository.
- ***Type:*** Type provides information and nature of the problem such as *documentation, translation, bug, bug fix, feature.*
- ***Screenshot:*** Visual representation of the issue or pull request is encouraged for better understanding and debugging.
- ***Summary:*** A detailed description or summary of the issue/pull request can help the code owners to understand the core idea for the changes presented.
- ***Environment:*** Providing information about the system configurations and environment of the application/software.
- ***Log:*** Including the relevant logs and stack trace can be beneficial for identifying the bugs.
- ***Severity:*** Impact of the issue/pull request is specified, so the importance of the changes is noted.
- ***Impact Assessment:*** Breaking changes or updating a library needs to be mentioned for the OSS community.
- ***Documentation & Testing***: For a complex feature or a bug fix, a proper write-up needs to record for future reference.
- ***Collaboration & Review:*** Few collaboration checklists would be given for the contributors to maintain consistency within the repository.
- ***Submission Guidelines & Formatting:*** Each Project has its own styling guidelines and custom liners which need to be satisfied.
- ***Issue/Problem Description:*** While the summary gives a detailed explanation of the changes/discrepancies, this label will provide the issue description.
- ***Debugging Information:*** Provides the debugging information, such as error messages and scenarios.
- ***Greeting & Introduction:*** Importance of a greeting to contributors and gratitude for their contributions.
- ***Proposed Solution & Suggestions:*** Proposing potential solutions and suggestions for opening a discussion with the community.

An analysis of the templates reveals the distinct contents of the templates, in *issue_templates* we tend to see *"Additional Context" **(20.67%)**, "Issue/Problem Description" **(19.63%)**, "Related Issues" **(18.58%** and "Steps of Reproduce" **(17.25%)*** as a significant part. On the other hand, in *pull_request_templates* we have *"Additional context" **(24.48%)**, "Documentation and Testing" **(13.6%)**, "Summary" **(11.55%)**, "Issue/Problem Description" **(15.16%)*** as significant percentage. However, the distribution varies, and the taxonomy gives us insights into what templates provide to a repository. In issue templates, the context and ability to reproduce the issue are emphasized while the documentation and collaboration are focused in the pull request templates.

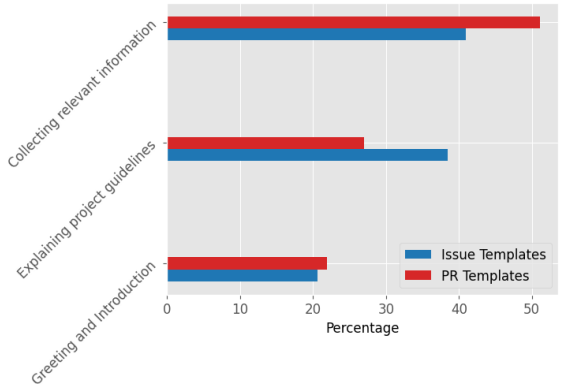

Figure 3: Distribution of Contents based on [9] Higher Categories

From [9], comparing the two sets of templates on higher category levels (***"Greeting and Introduction", "Explaining Project guidelines", "Collecting relevant information"***). It is evident that the templates seek to ***collect relevant information*** as it is found predominantly in both templates displaying the significance of comprehensive information required for the maintainers. In other categories, expressing gratitude, and introducing one to the templates takes up approximately ***20%*** in both templates. And finally, the ***"Explaining project guidelines"*** sums up to ***27.02%*** and ***38.41%*** in pull request and issue templates respectively.

Overall, we investigated and provided some valuable information on the contents of the templates by leveraging the capabilities of the language model to provide an accurate and flexible means of classification of template contents, also showcasing the distribution of different categories within the templates.

4.3.3 ***How do the contents of GitHub templates differ across programming languages?*** For the next research question, we incorporated the previous research questions analysis and further investigate the repository's language level. There were ***25*** unique languages in the dataset, we have listed a few of the statistics of the templates based on the language 3.

For simplicity, we have characterized the languages based on the Github tags. And we have concluded to have ***Frontend Languages, Backend Languages, System Languages*** and ***Other languages.*** All the client-facing languages and repositories are considered to be *front-end languages* such as *Javascript, Typescript, Dart, CSS, HTML* and *SCSS*, while languages like *Python, Go, Ruby, PHP, Kotlin, Java, Rust, C#, C++, and C* are categorized as *backend languages*. And *Assembly, Swift* and *OCaml* is said to be Assembly, the rest of the languages are put under the label as *Other Languages*.

We investigated further into the templates based on the categorization, we find frontend languages have a significant percentage for the ***"Issue/Problem Description", "Additional Context"*** and ***"Steps to Reproduce"***. This establishes and gives insights that the

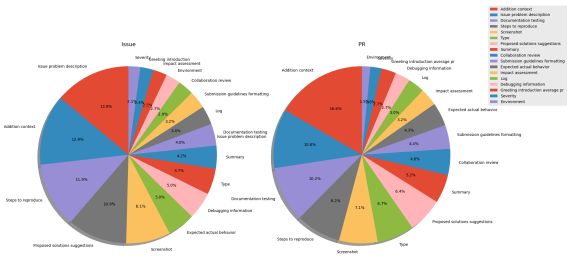

Figure 4: Distribution of Front end Language Templates

front-end community needs to encourage the contributor to provide a brief explanation and reproducible steps for fixing an issue. On the pull requests, we have the *"Documentation & Testing"* on the top 3 as the maintainers expect the contributors to give documentation and record on what changes are done as part of an issue or feature enhancement.

Back-end languages tend to prioritize content related to issue tracking, however, the back-end languages give more importance to the *"Proposed Solutions/Suggestions"* and *"Expected/Actual Behaviour"*, which gives an insight that the back end developers primarily focus on analysis and discussing the potential solutions and scenarios.

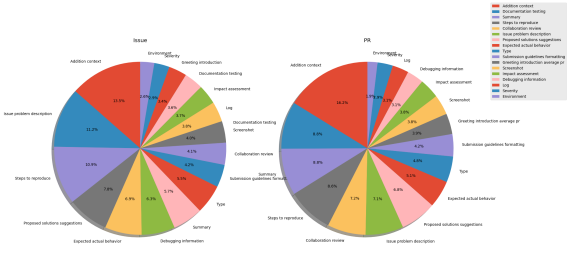

Figure 5: Distribution of Backend Language Templates

Furthermore, from 5 we can see the importance of *"Debugging Information"* and *"Impact Assessment"* compared to the front-end languages.

Languages like *Markdown, Jupyter Notebook, Shell, Clojure and None* prioritize *"Debugging Information"* and *"Expected/Actual Behaviour"*, mostly these languages focus on the thorough debugging information. On the other hand, System languages such as *Assembly, Swift, and OCaml* share similar characteristics with back-end languages, but they tend to lean on the *"Environment"* as the maintainers need to address environmental-specific issues. The detailed and more visualization are available at https://github.com/yogeshvar/GH-Templates

5 FINDINGS

In the study, we created a dataset of *538* repositories using the GitHub API and we selected only the repositories that included both *Pull Request Templates* and *Issue Templates*. Our analysis gave some interesting insights into the Github templates across languages, commonly used templates, and the contents of the templates. The average number of issue templates and pull request templates was $\approx$ 2.6 and 1.0. Moreover, we have found that the most commonly used issue templates are named as ***bug_template.md*** while the pull request template is named as ***pull_request_template.md***. We also found that ***82.1%*** of repositories has more than one issue template.

In addition to quantifying the template usage, we also investigated the taxonomy of the templates. Through language model classification of the template files, we identified common sections of the template files. With [9], we formulated the categories for classifying the template contents. Notably, templates exhibit ***Greeting and Introduction", "Explaining Project guidelines", "Collecting relevant information"***. Additionally, we investigated the templates based on the languages which lead us to discover interesting insights such as *Javascript* and *Go* repositories had an average of **2.52** and **3.18** issue templates respectively.

6 DISCUSSION

The findings from our analysis provide some valuable insights into the usage and contents of the Github templates. This aligns with the motivation of our research, as the communication is effective with these templates. We can provide general guidelines with these findings and streamline the development process. Although, the findings give us valuable insights into the templates. But it does not provide us the information on the impact of the templates on a repository.

6.1 Limitations

- *GPU Access*: A language Model like *Facebook/bart-large-mnli* comes with huge parameters, so running on a huge dataset requires GPU for better results. We have utilized Cloud GPUs from *Kaggle* platforms as our workaround.

7 THREATS OF VALIDITY

The following section shows the potential threats to the validity of the research findings,

Table 3: Issue Statistics

Language	Number of Repos	Number of Issues	Average Issue Templates per Repo	Number of PR	Average PR Templates per Repo
JavaScript	28	48	1.71	8	1.33
Python	23	51	2.22	9	2.25
Dart	1	3	3.00	1	1.0
TypeScript	16	38	2.38	7	1.75
Markdown	1	2	2.00	1	1.0
Go	21	51	2.43	3	1.0
C++	14	51	3.64	2	1.0
Jupyter Notebook	2	10	5.00	2	1.0
C#	3	8	2.67	3	1.5
Assembly	1	2	2.00	1	1.0
Ruby	11	21	1.91	2	1.0
CSS	1	2	2.00	1	1.0
Shell	1	1	1.00	1	1.0

- ***Selection Bias:*** The studies is probably affected by selection bias as the repositories blanketed inside the evaluation is primarily based on unique standards along with stars greater than a thousand and top programming languages. This should bring about a little pattern that won't represent all software program tasks on GitHub.
- ***Maturation:*** If we take a look at the *GitHub* templates over a long time frame, then Maturation could be a potential threat. The templates and practices utilized by repositories on GitHub may also evolve or exchange, leading to exclusive results at only points in time. Changes within the templates over the years may confound the analysis and interpretation of the findings. For instance, recent updates of *GitHub* templates have exchanged the templates with *customizable* form that supports *YAML* language.
- ***History:*** History can be a risk to validity if some significant changes or activities occur at some point in the observed duration. For instance, if GitHub introduces new functions or updates its platform's functionalities related to the issue and pull request templates, it can affect how repositories use and preserve those templates. This historic context wishes to be considered when deciphering the findings.
- ***External Validity:*** For most of the open-source projects are maintained in *GitHub*, but there are noticeable OSS projects handle their version control elsewhere than *GitHub*. This raises questions about the generalizability of the findings. The analysis carried out within the have a look at might not seize the template utilization, styles, and traits of repositories that do not adhere to the usual practices on *GitHub*. The repository might have a different technique or mechanism to handle their effective communication.
- ***Measurement Bias:*** There might be bias if the information accumulated from GitHub, including issue and pull request templates, isn't accurately recorded or interpreted. If any error or misinterpretation can affect the analysis and findings of the study. For example, a mistake in the conversion of `list` based issue templates to *string* based would change the total number of issue templates.

8 RELATED WORK

Our research contributes to the existing body of knowledge on GitHub templates by addressing specific research questions related to their taxonomy and the variations across programming languages. A study conducted by [9], focused on examining the content of templates using a traditional natural language processing coding approach without any use of language models or classification algorithms, but it provides a baseline for the structure and elements of templates. Similarly, a study by [18] investigated the consistency of the issue templates and provided some importance of standardized formats for effective issue tracking. However, our research stands out by exploring the contents of GitHub templates on a language level, finding how templates vary across programming languages. This novel approach adds a new pathway to explore the language-specific patterns in the templates. By considering the broader research landscape, there are several research conducted on customizing templates [4], the introduction of templates [6], and the impact of templates [9].

9 CONCLUSIONS

In conclusion, our research on GitHub templates has enabled us to understand the usage and characteristics of OSS projects. By analyzing a diverse dataset of *538* repositories, we have revealed that GitHub repositories incorporate multiple issue templates for collaboration and widely used templates across the repositories. And the average number of templates on the repositories as well. In terms of the taxonomy, our qualitative analysis identified common sections present in the templates. The overall higher-order categorization *"greeting contributors, providing project guidelines, and collecting relevant data."* shows some results on the sections and labels which was focused on the project.

Furthermore, our research discovered trends specific to programming languages. For instance, Backend specific languages prioritize *"Proposed Solutions/Suggestions", "Expected/Actual Behaviour"* while the front-end languages prioritize *"Steps to Reproduce"* and *"Additional Information"*. As the field of open-source development evolves

every day, further research on the impact of template customization and the effectiveness and impact of the templates will provide deeper insights for refining best practices for Github Templates in OSS projects.

10 FUTURE WORK

With the current progress, the upcoming work should be on *Understanding the Impact of Templates*. This could be an interesting research topic as the researcher would be focused on investigating the impact of the pull request templates and how long it takes for the pull requests to be reviewed, accepted, rejected, or merged. From this analysis, we can get insights about template complexity, and template customization, and proceed towards optimizing it for effective collaboration and communication among the OSS community. Furthermore, with this research we can identify patterns and characteristics of templates that will lead us to better turnaround times for the pull requests, enabling the project to fix their bugs faster as we will streamline these templates with development workflows.

One of the recent changes to the *GitHub* Templates was the introduction of *YAML* type forms as templates, this investigation can be conducted to examine the extent to which projects customize the templates as form *YAML Template* or the *classic templates*. By studying the level of template customization, and the quality of effective communication, we can enable the shortcomings of these areas to improve the collaboration between project stakeholders and developers. And, this research can give us the significance of presenting available template customization options and encourage the development to equip these techniques and features to facilitate template personalization for their projects and maintain a standard consistency of pull requests and issues documentation for better software development.

These future work ideas are not limited to these two described above, but the provided ones serve as a direction for the exploration of templates on a deeper level. By investigating template usage, impact, customization, and their integration with the software development process, the researcher can gain valuable insights that can be used to form best practices and guidelines for the continuous improvement of the collaborative software development process.

REFERENCES

[1] Lingfeng Bao, Xin Xia, David Lo, and Gail C. Murphy. 2021. A large scale study of long-time contributor prediction for GitHub projects. *IEEE Transactions on Software Engineering* 47, 6 (1 June 2021), 1277–1298. https://doi.org/10.1109/TSE.2019.2918536

[2] Hudson Borges, Andre Hora, and Marco Tulio Valente. 2016. Understanding the Factors That Impact the Popularity of GitHub Repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 334–344. https://doi.org/10.1109/ICSME.2016.31

[3] Silvia Breu, Rahul Premraj, Jonathan Sillito, and Thomas Zimmermann. 2010. Information Needs in Bug Reports: Improving Cooperation between Developers and Users. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work* (Savannah, Georgia, USA) *(CSCW '10)*. Association for Computing Machinery, New York, NY, USA, 301–310. https://doi.org/10.1145/1718918.1718973

[4] John L. Campbell, Charles Quincy, Jordan Osserman, and Ove K. Pedersen. 2013. Coding In-depth Semistructured Interviews: Problems of Unitization and Intercoder Reliability and Agreement. *Sociological Methods & Research* 42, 3 (2013), 294–320. https://doi.org/10.1177/0049124113500475 arXiv:https://doi.org/10.1177/0049124113500475

[5] Vijaya Kumar Eluri, Thomas A. Mazzuchi, and Shahram Sarkani. 2021. Predicting long-time contributors for GitHub projects using machine learning. *Information and Software Technology* 138 (2021), 106616. https://doi.org/10.1016/j.infsof.2021.106616

[6] GitHub. Accessed 2023. 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.

[7] GitHub. Accessed 2023. Configuring issue templates for your repository. https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/configuring-issue-templates-for-your-repository.

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461 [cs.CL]

[9] Zhixing Li, Yue Yu, Tao Wang, Yan Lei, Ying Wang, and Huaimin Wang. 2023. To Follow or Not to Follow: Understanding Issue/Pull-Request Templates on GitHub. *IEEE Transactions on Software Engineering* 49, 4 (2023), 2530–2544. https://doi.org/10.1109/TSE.2022.3224053

[10] Audris Mockus, Roy T. Fielding, and James Herbsleb. 2000. A Case Study of Open Source Software Development: The Apache Server. In *Proceedings of the 22nd International Conference on Software Engineering* (Limerick, Ireland) *(ICSE '00)*. Association for Computing Machinery, New York, NY, USA, 263–272. https://doi.org/10.1145/337180.337209

[11] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. 2017. Curating GitHub for Engineered Software Projects. *Empirical Softw. Engg.* 22, 6 (dec 2017), 3219–3253. https://doi.org/10.1007/s10664-017-9512-6

[12] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. 2002. Evolution patterns of open-source software systems and communities. In *International Workshop on Principles of Software Evolution*.

[13] Devarshi Singh, Varun Ramachandra Sekar, Kathryn T. Stolee, and Brittany Johnson. 2017. Evaluating how static analysis tools can reduce code review effort. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2017), 101–105.

[14] Xin Tan and Minghui Zhou. 2019. How to Communicate When Submitting Patches: An Empirical Study of the Linux Kernel. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 108 (nov 2019), 26 pages. https://doi.org/10.1145/3359210

[15] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Let's Talk about It: Evaluating Contributions through Discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 144–154. https://doi.org/10.1145/2635868.2635882

[16] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. 805–816. https://doi.org/10.1145/2786805.2786850

[17] Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. arXiv:1909.00161 [cs.CL]

[18] 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 (April 2022). https://doi.org/10.1016/j.infsof.2021.106797 Funding Information: The work is funded by Natural Science Research Foundation of Jilin Province of China under Grant Nos. 20190201193JC , supported by Graduate Innovation Fund of Jilin University, China 101832020CX181 , supported by "the Fundamental Research Funds for the Central Universities", China , and Interdisciplinary Research Funding Program for Doctoral Students of Jilin University, China 101832020DJX064 . Publisher Copyright: © 2021 Elsevier B.V..

[19] Thomas Zimmermann, R. Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schröter, and Cathrin Weiss. 2010. What Makes a Good Bug Report? *IEEE Transactions on Software Engineering* 36 (09 2010), 618–643. https://doi.org/10.1109/TSE.2010.63