# The Maintenance of Top Frameworks and Libraries Hosted on GitHub: An Empirical Study

Yi Huang, Xinjun Mao, Zhang Zhang
National University of Defense Technology, Changsha, China
{huangyi20, xjmao, zhangzhang14}@nudt.edu.cn

***Abstract*—The number of repositories on GitHub is huge and growing rapidly. However, most repositories are inactive, while active maintenance is essential in choosing a project. In this paper, we study the maintenance of top (i.e., most starred) frameworks and libraries hosted on GitHub, for they can be widely reused and in critical positions in the dependency network, so their maintenance status is significant. Furthermore, their maintenance practices may inspire other projects to thrive on the collaborative development platform. By investigating their adoption of recommended Open Source Software (OSS) maintenance practices and recent maintenance activities, and the association between maintenance status and usage, we find that: (1) more than 20% of the top frameworks and libraries have no commit for more than one year; (2) Some maintenance practices (e.g., codes of conduct) have relatively low adoption rates, while continuous integration has a high adoption rate of around 80%; (3) the maintenance status may have an effect on the usage frequency.**

***Index Terms*—software maintenance, open source guide, software dependency**

## I. Introduction

Software maintenance is critical and lasts the longest in the software life cycle. Also, it is an essential factor for developers to consider when choosing software [1]. Over the years, with the rise of open source and the emergence of code hosting platforms, more and more companies and developers maintain their projects on code hosting platforms, especially on GitHub. As the world's largest code hosting platform, GitHub has hundreds of millions of repositories and tens of millions of users, and the number is still proliferating. However, according to research [2], though the total number of repositories is vast, most of the repositories hosted on GitHub are inactive. The finding inspired us to think: are the top (e.g., most starred) projects hosted on GitHub active? Furthermore, how well are they maintained? Especially the top frameworks and libraries—while they are just the tip of the iceberg, they can be critical—they rank high, attract considerable attention, and are more likely to be reused by other projects to save development costs and increase efficiency. Therefore, their maintenance status can significantly affect related projects. Additionally, their practical experience in development and maintenance may inspire and guide other projects to thrive on GitHub.

This paper conducts an empirical study of the top frameworks and libraries in GitHub to investigate their adoption of recommended OSS maintenance practices, recent maintenance activities, and the association between maintenance status and usage. We believe that on the collaborative development platform GitHub, the maintenance complexity is greater than the maintenance complexity of individual or single team development. Attracting and retaining contributors and the interaction between developers are critical. Besides, the interaction between developers is not limited to maintaining code but also includes documentation maintenance, issue resolution, and project discussion. These aspects reflect the maintenance of the project and are worthy of in-depth exploration. In this paper, we address the following three research questions:

***RQ1. (Maintenance Practices Adoption) How well do the top frameworks and libraries follow the recommended OSS maintenance practices?*** The purpose of the question is to investigate the adoption of some recommended maintenance practices by top frameworks and libraries. These practices are important factors for contributors to decide whether to contribute to the project [3].

***RQ2. (Maintenance Activity) What is the level of recent activity present in the top frameworks and libraries?*** While these frameworks and libraries received many stars, they may be dormant, decayed, or even deprecated. Once developers use the deprecated libraries or frameworks, this can be a potential risk to their projects. Therefore, it is necessary to investigate their recent maintenance activities.

***RQ3. (Usage) How often are the top frameworks and libraries used by other projects in GitHub?*** The goal of the question is to investigate the usage of top frameworks and libraries, and the association between maintenance status and usage.

**Paper Organization.** The rest of the paper is organized as follows. The information about our dataset is present in Section II. Our research methods and results are presented in Section III for RQ1, Section IV for RQ2, and Section V for RQ3. In Section VI, we provide an overview of related work. Section VII concludes our work.

## II. Dataset

First, we referred to the previous work [4] and took 5,000 as the threshold to select the top-5,000 most starred repositories hosted on GitHub (In November 2021). Then referring to the classification criteria and results of the existing paper [5], we

DOI reference number: 10.18293/SEKE2022-092

manually selected frameworks and libraries repositories from the top-5,000 repositories, and we used the following strategies to select them: first, we selected the repositories that have keywords such as *framework* and *library* in their descriptions or README files; second, we selected the repositories that can be found in package managers (e.g., Maven, PyPI); after the above procedures, we then manually checked the repositories to assure the correctness of the selection.

Finally, we got 2,092 repositories. We collected the basic information (e.g., the number of stars, forks, commits, contributors). Besides, we collected events of issues, pulls, and commits for the repositories for the recent year between November 2020 and October 2021 through GitHub REST API. To investigate the usage of top frameworks and libraries by other repositories hosted on GitHub, we further selected repositories with more than 500 stars and had at least one commit in the past three months (for quality assurance), resulting in an initial set of 24,605 repositories. Then we selected the repositories with manifest files from the initial set, and we ended up with 14,666 repositories as potential client projects. We used libraries.io[1] to gather their dependency information. The more detailed data for analysis and its extraction process will be elaborated in the following *methodology* subsections. The data and scripts for the replication of this study are available[2].

TABLE I: Data Basic Statistics

| Metric | Min | 25% | Median | 75% | Max | Average |
|---|---|---|---|---|---|---|
| **Star** | 5,319 | 6,667 | 8,955 | 13,501 | 189,629 | 12,593 |
| **Fork** | 2 | 712 | 1,291 | 2,290 | 85,709 | 2,229 |
| **Age (weeks)** | 12 | 276 | 372 | 475 | 717 | 376 |
| **Commit** | 3 | 449 | 1,158 | 2,862 | 121,196 | 3,292 |
| **Contributor** | 1 | 34 | 85 | 181 | 4,377 | 163 |

Table I shows statistics on the number of stars, number of forks, age, number of commits, and number of contributors for the repositories in our dataset.

## III. RQ1: Maintenance Practices Adoption

### A. Methodology

TABLE II: Studied Maintenance Practices

| Dimension | Practices | Rationale |
|---|---|---|
| **Documentation** | contribution guidelines | To guide contributors do good work |
| | codes of conduct | To define community standards to facilitate healthy community behavior |
| | template | To provide guidance for opening issues or pull requests |
| | good first issues | To highlight opportunities for people to contribute |
| | homepage | To show all kinds of information related to the project |
| **Community** | discussion | To provide a collaborative communication forum to talk about the project |
| | chat platform | Similar to the discussion |
| **Automation** | continuous integration | To improve the productivity of the project team |

[1]https://libraries.io/

[2]10.6084/m9.figshare.18665744

Table II shows the recommended OSS maintenance practices studied in our paper. Some practices are recommended by GitHub, and some are recommended by previous work [6], [3], [4]. We divided them into three dimensions: *documentation*, *community* and *automation*.

**Documentation Dimension.** In this dimension, we considered five practices: adoption of *contribution guidelines*, *codes of conduct*, *template*, *good first issues* and *homepage*. These are practices recommended by GitHub to set projects for healthy contributions. To investigate their adoption, we searched for the relevant files (e.g., CONTRIBUTING.md, CODE_OF_CONDUCT.md, ISSUE_TEMPLATE.md) from the repositories' directories and checked the issue labels.

**Community Dimension.** In this dimension, we considered the usage of *discussion* and *chat platform*. The *discussion* refers to the new feature—GitHub Discussions. As for the usage of *chat platform*, we investigated whether the frameworks and libraries repositories have adopted Slack, Discord, or Gitter.

**Automation Dimension.** In this dimension, we considered the usage of *continuous integration*, and we investigated whether the top frameworks and libraries repositories have adopted Travis CI, GitHub Actions, or CircleCI.

First, we classified the collected repositories into five groups: the *top* group, which refers to the collection of top-500 repositories by the number of stars; the *bottom* group; the *active* group, which refers to the collection of repositories with at least one commit in the past month (the count is 1,034, 49.43%); the *inactive* group, which refers to the collection of repositories with no commits in the past year (the count is 448, 21.24%); the *all* group, which refers to the collection of all repositories in our dataset. The purpose of this grouping is to present the statistical differences between the top and bottom repositories, active and less active repositories, and the overall statistical characteristics of all collected repositories. Then, according to the classification of the groups, we calculated and analyzed the corresponding adoption rates of the above practices. The grouping shows that although these top frameworks and libraries received many stars and ranked high, more than 20% of them have not committed for more than one year.

### B. Results

Table III shows the adoption rate of each practice by each group. Regardless of the group, the most followed practices is *continuous integration*. For *continuous integration*, the *inactive* group has the lowest adoption rate at 58.48%, while the *active* group has the highest adoption rate at 88.88%. Nevertheless, the adoption rates of *codes of conduct*, *good first issues*, *discussion* and *chat platform* are relatively low, not exceeding 50% in each group. Moreover, for each practice, the *inactive* group has the lowest adoption rate; and the adoption rates of *codes of conduct*, *template*, *good first issues* and *discussion* in *inactive* group are significantly low, which are 1/6, 1/4, 1/4 and 1/34 of the highest adoption rates.

Figure 1 shows the frequency distribution of the number of practices adopted by each group, where the *y*-axis repre-

TABLE III: Percentage of Repositories Following Recommended Maintenance Practices

| Practice | Active | Inactive | Top | Bottom | All |
|---|---|---|---|---|---|
| contribution guidelines | 65.57% | 31.70% | 71.20% | 39.60% | 53.11% |
| codes of conduct | 34.43% | 6.03% | 36.00% | 16.60% | 23.57% |
| template | 71.18% | 17.19% | 73.00% | 37.60% | 51.77% |
| good first issues | 36.94% | 9.15% | 32.60% | 23.80% | 26.86% |
| homepage | 76.11% | 54.24% | 80.60% | 58.00% | 67.97% |
| discussion | 40.14% | 1.79% | 36.40% | 17.20% | 24.81% |
| chat platform | 38.39% | 15.83% | 40.60% | 20.40% | 29.78% |
| continuous integration | 88.88% | 58.48% | 88.40% | 73.20% | 79.73% |

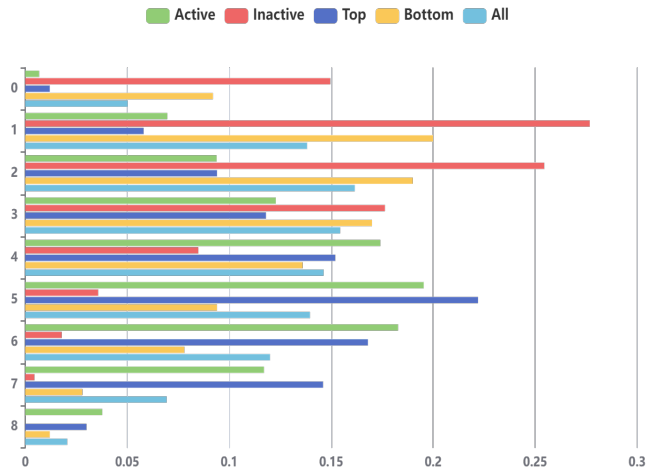


Fig. 1: Frequency Distribution of The Number of Practices Adopted by Different Groups

sents the number of practices and the *x*-axis represents the frequency. For the *inactive* group, the number of practices adopted mainly (84%) falls into the range 0 to 3. For the *active* and *top* groups, the top-3 number of practices adopted are in the range of 4 to 6. Few frameworks and libraries have adopted all the above practices though they are in active status. For *active* group, only 39 (3.77%) of the 1,034 repositories have adopted all eight studied practices.

## IV. RQ2: MAINTENANCE ACTIVITY

### A. Methodology

TABLE IV: The Metrics of Maintenance Activities

| Dimension | Metric | Description |
|---|---|---|
| **Commit** | commit_count | The number of commits |
| **Release** | release_count | The number of releases |
| **Issue** | issue_count | The number of issues |
| | issue_closed_average_time | The average time of closing an issue |
| | issue_closed_ratio | The ratio of closed issues |
| | issue_replied_ratio | The ratio of replied issues |
| | issue_replied_average_time | The average time of replying to an issue |
| **Pull Request** | pull_count | The number of pull requests |
| | pull_closed_ratio | The ratio of closed pull requests |
| | pull_closed_average_time | The average time of closing a pull request |
| | pull_merged_ratio | The ratio of merged pull requests |
| | pull_merged_average_time | The average time of merging a pull request |
| | pull_replied_ratio | The ratio of replied pull requests |
| | pull_replied_average_time | The average time of replying to a pull request |

As shown in Table IV, we investigated the maintenance activities of the top frameworks and libraries repositories in the past year (from November 2020 to October 2021) from four dimensions: *commit*, *release*, *issue*, and *pull request*. And we used the groups introduced in RQ1.

### B. Results

TABLE V: Maintenance Activities of *active*, *inactive*, *top*, *bottom*, and *all* Groups.

| Metric | Active | Inactive | Top | Bottom | All |
|---|---|---|---|---|---|
| commit_count | 475 | 0 | 416 | 181 | 248 |
| release_count | 25 | 0 | 17 | 6 | 13 |
| issue_count | 314 | 14 | 376 | 91 | 178 |
| issue_closed_ratio | 63.92% | 15.20% | 58.09% | 41.50% | 47.62% |
| issue_closed_average_time | 172.34h | 257.65h | 165.40h | 218.91h | 215.23h |
| issue_replied_ratio | 76.58% | 31.95% | 72.48% | 57.83% | 62.97% |
| issue_replied_average_time | 48.10h | 472.99h | 84.29h | 245.95h | 180.07h |
| pull_count | 346 | 4 | 389 | 100 | 183 |
| pull_closed_ratio | 89.29% | 11.37% | 74.53% | 57.80% | 64.78% |
| pull_closed_average_time | 108.38h | 412.90h | 165.24h | 228.00h | 246.49h |
| pull_merged_ratio | 66.72% | 0.09% | 50.17% | 39.47% | 44.55% |
| pull_merged_average_time | 100.04h | 183.55h | 137.26h | 205.19h | 171.50h |
| pull_replied_ratio | 60.09% | 15.67% | 56.63% | 41.00% | 47.26% |
| pull_replied_average_time | 69.76h | 508.34h | 159.89h | 216.09h | 234.37h |

The results in Table V show that the *active* group outperforms the other groups in most metrics while the *inactive* group lags behind the other groups in most metrics. For the average number of commits and releases, the *active* group has the maximum values, 475 and 25, respectively, while the *inactive* group has the minimum values, both zero. For the average number of issues and pull requests, the *top* group has the maximum values of 376 and 389, while the *inactive* group has the minimum values of 14 and 4, respectively. Besides, the *active* group has the highest ratios of closed issues, replied issues, closed pull requests, merged pull requests, and replied pull requests, which are 63.92%, 76.58%, 89.29%, 66.72%, and 60.09%, respectively, while the *inactive* group has the lowest which are 15.20%, 31.95%, 11.37%, 0.09%, and 15.67% respectively.

Interestingly, we found that: in *inactive* group, the average time of replying to an issue is longer than the average time of closing an issue; the average time of replying to a pull request is longer than the average time of closing or merging a pull request; however, the above situation is reversed in *active* group.

## V. RQ3: USAGE

### A. Methodology

In this section, as for the usage of top frameworks and libraries, we investigate their usage frequency and outdated usage. We used libraries.io to extract the dependencies of each potential client project. The libraries.io has organized all the manifest files (e.g., pom.xml, package.json) for the project. Finally, we extracted 216,504 dependencies.

### B. Results

TABLE VI: Statistics on the usage frequency and outdated usage ratio

| Metric | Min | 25% | Median | 75% | Max | Average |
|---|---|---|---|---|---|---|
| **Usage Frequency** | 0 | 0 | 4 | 33 | 13,069 | 102 |
| **Outdated Usage Ratio** | 0 | 9.22% | 49.77% | 80.00% | 100.00% | 47.10% |

Table VI shows statistics on the usage frequency and outdated usage ratio. We define the outdated usage ratio as the number of outdated uses of a framework or library divided by its total uses. Of the 2,092 studied top frameworks and libraries, 1,362 are used at least once, only 285 do not have outdated usage, and 146 projects are wholly used with outdated versions.

We applied the Mann-Whitney U test to analyze the statistical significance of the difference between the top-100 most used frameworks and libraries and the unused frameworks and libraries in the metrics mentioned in RQ2, and we used Cliff's delta [7] to show the effect size of the difference. We found a statistically significant difference between them in all metrics, and all 14 metrics have large effect sizes. Further, We calculated the median and the mean of the 14 metrics of the top-100 most used and unused projects and found that the former outperforms the latter in all metrics. For example, the median number of commits in the past year of the former is 74 while the latter is 12, and the issue closed ratio of the former is 70.14% while the latter is 35.15%. Therefore, the maintenance status may have an effect on the usage frequency. Then, we used the same methods above to analyze the projects that do not have outdated usage and projects that are wholly used with outdated versions. We found a statistically significant difference between them in all metrics, and 10 of 14 metrics have large effect sizes, while the other metrics have medium effect sizes. Further, we found that the latter outperforms the former in 13 metrics (except the issue closed average time).

## VI. Related Work

Much work has been done studying software maintenance. There is some work [8], [9] focused on investigating or measuring the maintenance status of projects; some work [10] focused on the barriers faced by contributors (e.g., peripheral contributors); some work [4], [6], [11], [3] focused on the recommended OSS maintenance practices that may guide or automate the maintenance and contribution process. Coelho et al. [8] proposed a machine learning model to identify unmaintained GitHub projects and defined a metric to measure the level of maintenance activity of GitHub projects. Lee et al. [10] conducted an online survey to investigate the barriers one-time code contributors faced when contributing to FLOSS projects and highlighted the significance of timely feedback and guidance through the patch submission process. Hilton et al. [6] studied the usage, costs, and benefits of continuous integration in open source projects. They found that the overall percentage of projects using CI continues to grow, and CI helps projects release more frequently and accept pull requests faster. Alderliesten et al. [11] initially explored the "good first issues" label and found that though they are effective at developer onboarding and considered useful, their types need to be refined to match the types of initial contributions.

## VII. Conclusion

In this paper, we conducted an empirical study to investigate the maintenance of top frameworks and libraries hosted on GitHub. We found that some OSS recommended maintenance practices are not widely adopted even in the top frameworks and libraries. For example, the adoption rates of codes of conduct and good first issues are 23.57% and 26.86%. Further, we used quantity, proportion, and response time as metrics to analyze the recent maintenance activities of the top frameworks and libraries. Moreover, we found that the maintenance status may have an effect on the usage frequency. In future work, we plan to propose a unified measure of open source project maintenance status.

## References


[1] E. Larios Vargas, M. Aniche, C. Treude, M. Bruntink, and G. Gousios, *Selecting Third-Party Libraries: The Practitioners' Perspective*. New York, NY, USA: Association for Computing Machinery, 2020, p. 245–256. [Online]. Available: https://doi.org/10.1145/3368089.3409711

[2] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. German, and D. Damian, "The promises and perils of mining github," 06 2014.

[3] D. Sholler, I. Steinmacher, D. Ford Robinson, M. Averick, M. Hoye, and G. Wilson, "Ten simple rules for helping newcomers become contributors to open projects," *PLoS Computational Biology*, vol. 15, September 2019.

[4] J. Coelho and M. Valente, "Why modern open source projects fail," 08 2017, pp. 186–196.

[5] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2016, pp. 334–344.

[6] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 426–437.

[7] G. Macbeth, E. Razumiejczyk, and R. D. Ledesma, "Cliff's delta calculator: A non-parametric effect size program for two groups of observations," *Universitas Psychologica*, vol. 10, no. 2, pp. 545–555, 2011.

[8] J. Coelho, M. Valente, L. Milen, and L. Silva, "Is this github project maintained? measuring the level of maintenance activity of open-source projects," *Information and Software Technology*, 02 2020.

[9] G. Avelino, E. Constantinou, M. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," 09 2019, pp. 1–12.

[10] A. Lee, J. C. Carver, and A. Bosu, "Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: A survey," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017, pp. 187–197.

[11] J. W. D. Alderliesten and A. Zaidman, "An initial exploration of the "good first issue" label for newcomer developers," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2021.