On the Adoption of a TODO Bot on GitHub: A Preliminary Study

Hamid Mohayeji
h.mohayeji.nasrabadi@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Felipe Ebert
f.ebert@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Eric Arts
e.m.a.arts@student.tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Eleni Constantinou
e.constantinou@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Alexander Serebrenik
a.serebrenik@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

ABSTRACT

Bots support different software maintenance and evolution activities, such as code review or executing tests. Recently, several bots have been proposed to help developers to keep track of postponed activities, expressed by means of TODO comments: *e.g.*, `TODO Bot` automatically creates a GitHub issue when a TODO comment is added to a repository, increasing visibility of TODO comments. In this work, we perform a preliminary evaluation of the impact of the `TODO Bot` on software development practice. We conjecture that the introduction of the `TODO Bot` would facilitate keeping track of the TODO comments, and hence encourage developers to use more TODO comments in their code changes.

To evaluate this conjecture, we analyze all the 2,208 repositories which have at least one GitHub issue created by the `TODO Bot`. Firstly, we investigate to what extent the bot is being used and describe the repositories using the bot. We observe that the majority (54%) of the repositories which adopted the `TODO Bot` are new, *i.e.*, were created within less than one month of first issue created by the bot, and from those, more than 60% have the issue created within three days. We observe a statistically significant increase in the number of the TODO comments after the adoption of the bot, however with a small effect size. Our results suggest that the adoption of the `TODO Bot` encourages developers to introduce TODO comments rendering the postponed decisions more visible. Nevertheless, it does not speed up the process of addressing TODO comments or corresponding GitHub issues.

KEYWORDS

TODO, bots, technical debt, code comments

ACM Reference Format:
Hamid Mohayeji, Felipe Ebert, Eric Arts, Eleni Constantinou, and Alexander Serebrenik. 2022. On the Adoption of a `TODO Bot` on GitHub: A Preliminary Study. In *Fourth International Workshop on Bots in Software Engineering (BotSE 2022), May 9, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3528228.3528408

BotSE 2022, May 9, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9333-1/22/05…$15.00
https://doi.org/10.1145/3528228.3528408

1 INTRODUCTION

Software bots are used in software development to create a more efficient software development workflow [1]. In this regard, GitHub has recently introduced GitHub Apps, (user-created) bots that can interact with GitHub API. Such apps are becoming quite popular, *e.g.*, the Coveralls[1] is a bot to support the code review process which has more than 15k installations.

`TODO Bot`, created in September 2017, is one of such apps [4]. It is a GitHub App that automatically creates an issue if a push to the repository's master branch contains the word "TODO" (or another pre-configured keyword). It also places a comment in the Pull Request containing a commit with one of these pre-configured keywords. Bots can as well be associated with self-admitted technical debt (SATD) [10], the non-optimal or incomplete solutions during the software development process. [10] showed that mining code comments looking for patterns such as TODO comments is useful for detecting technical debts. Having this association in mind, the `TODO Bot` can be expected to help managing technical debt by increasing the visibility of TODO comments.

To begin with, we want to investigate the extent that the `TODO Bot` is being used on GitHub. Thus, we formulate our first research question:

RQ1. *To what extent is* `TODO Bot` *being used in GitHub?*

We identified 2,208 GitHub repositories using the `TODO Bot` belonging to 1,033 repository owners. We observed that the `TODO Bot` is more popular on recently created repositories than in older ones, and amongst those recent repositories, the majority have issues created by the bot with the first three days.

Furthermore, we also aim at understanding the characteristics of the projects using the `TODO Bot` in terms of popularity measure by the number of stars, forks, watchers, commits, and also the number of TODO issues created by the bot. This helps us to comprehend the adoption level of `TODO Bot` with respect to the popularity or activity of repositories. As such, we formulate our second research question:

RQ2. *What are the characteristics of repositories on which* `TODO Bot` *operates?*

We found that repositories using the `TODO Bot` usually have few stars, forks, and watchers. The `TODO Bot` has been mostly working on repositories with at most a few hundred commits.

[1] https://github.com/marketplace/coveralls

Finally, we aim at verifying whether the adoption of the `TODO Bot` affects the way developers use and address TODO comments. As such, we formulate our third research question:

RQ3. *How does the adoption of the* `TODO Bot` *affect the introduction of TODO comments and dealing with them?*

The results of this preliminary study show that, the `TODO Bot` adoption is affected by the frequency developers are adding TODO comments in their commits. In the set of repositories created more than one month before and after the first issue created by the bot, there is a statistically significant increase in the number of TODO comments after the adoption of the bot. These results confirm our expectation, *i.e.*, we expect that such a bot would encourage developers in the management of (self admitted) technical debt as the creation of GitHub issues would increase the visibility and management of the TODO comments. While this paradigm shifts from TODO comments to GitHub issues is fruitful in increasing the visibility of technical dues, our experiment reveals that the process of resolving those issues is not generally faster.

2 RELATED WORK

TODO comments are a specific type of task annotation which developers can use to manage their programming tasks. Storey *et al.*[14] conducted an empirical study on how task annotations in source code are used by developers and teams. They found that TODO comments are used for a wide range of tasks, such as splitting up a large task in smaller subtasks, or denoting edge cases so that handling their implementation can be deferred to a later point in time. Nie *et al.*[9] developed a Java framework called TrigIt that developers can use to write trigger-action TODO comments using Java code. These trigger-action TODO comments are then automatically resolved (by executing the provided action) when a certain condition (*i.e.*, the trigger) is met. A different approach was taken by Shridhara [13], who created a tool to check the up-to-date status of TODO comments.

However, without ways to manage TODO comments, they can still be lost track of. For instance, Storey *et al.*[14] also reported several TODO comments that were not revisited and left in the codebase of one of the projects they analyzed. This can lead to technical debt [3] later on in the project, more specifically, to so called self-admitted technical debt (SATD) [10]. SATD refers to situations where developers are aware the actual implementation is not optimal and they use this task annotation to alert the inadequacy of the solution, and TODO comments is one way of expressing SATD. SATD has been extensively investigated in the last few years. Maldonado *et al.*[8] investigated how much SATD is removed and who removes it. Their results showed that most of the SATD is removed also by the same person who introduced it. Iammarino *et al.*[5] investigated the relationship between refactoring and SATD removal. They found that refactoring is more likely to co-occur with the SATD removal than with other commits.

In this paper, we conduct a preliminary analysis of a bot which aims at helping developers manage TODO comments by increasing their visibility, *i.e.*, by creating GitHub issues. We argue this study is complementary to the others as we aim at understanding at what degree `TODO Bot` is adopted by repository owners and how effective it is in addressing technical debts expressed in the form of TODO comments. There are other GitHub Apps that provide a similar functionality as `TODO Bot`. We, however, decided to specifically focus on the `TODO Bot` in this study as it seemed to be the most popular variant when looking at i) the number of stars, forks, and watchers of the bot repository and ii) the search results on the GitHub marketplace and Google when searching for "todo" and "github todo", respectively.

3 METHODOLOGY

To answer our questions, we need to obtain collections of: (1) the issues created by the `TODO Bot`, along with the repositories related to those issues and (2) TODO comments that were created before `TODO Bot` was introduced to a repository.

In order to answer **RQ1** and **RQ2**, we use the GitHub search API to identify issues created by the `TODO Bot` and its repositories names, *i.e.*, we consider only public repositories with at least one issue created by the bot. Moreover, we fetch additional repository information (such as the number stars, number of forks, etc.) using GitHub's standard API. To answer **RQ3**, we need to identify TODO comments created before `TODO Bot` was introduced to a repository. The code that was used is publicly available.[2]

We use CLI-tool provided by the `TODO Bot` to check whether, given a repository and a commit SHA, if it would create an issue for that commit. However, the generated issues contained duplicates, as `TODO Bot` was unable to verify whether an issue already existed (*e.g.*, when a file is renamed). These duplicates were removed by keeping the commit date of the earliest duplicate commit. Furthermore, some TODO comments already had related issues and were therefore identified before. Each issue that was created by the `TODO Bot` relates to a commit, which must have been created before the issue was created. As such, the commit that caused the creation of the first TODO issue was also found in this search and discarded. To study statistical difference between the number of TODO comments before and after the adoption of the bot, firstly we check for normality with Shapiro-Wilk test [11], then we use the t-test [15], otherwise we use the non-parametric Wilcoxon test [2]. Finally, to compare the lifespan of the issues with the TODO comments, we run another experiment, in which for each repository, we leveraged Pydriller [12] library to extract the time difference between adding and removing TODO comments in the source code before the bot adoption, as well as calculating the same lifespan for issues based on their creation and termination dates. We clone the repositories and find commits touching TODO comments using regular expression "^\s*(#|--|<!--|//|/*+)+\s*(@todo|todo):?\s*(?P<text>.*)". Regarding issues, we calculated the elapsed time between the creation and closing date, provided by the GitHub API. Subsequently, we used Survival Analysis [7] on both TODO comments and issues to gain insight into the expected duration for each of those to be resolved by the developers.

4 RESULTS

The process of identifying the GitHub issues created by the `TODO Bot` resulted in 11,837 issues, belonging to 2,208 unique repositories.[3] The usage of the CLI-tool to identify the TODO comments

[2] https://figshare.com/s/d530438dd83035662a8c
[3] It was performed on September 6, 2020.

before the adoption of the TODO Bot returned a total of 34,948 TODO comments, which then were reduced to 20,809 after removing duplicates, belonging to 872 unique repositories. The following sections present the results of the **RQs**.

RQ1: To what extent is TODO Bot being used in GitHub?

We found 2,208 repositories which use TODO Bot, that are owned by 1,033 GitHub users. The fact that owners use TODO Bot in multiple of their repositories is not surprising given the ease of installing a GitHub app across their repositories with a single click. Developers seem also to be quickly adopting the TODO Bot as it is more popular on recently created repositories (54%), *i.e.*, where the creation date is less than one month before the first issue created by the bot. Also, more than 60% of the repositories have TODO Bot issues created with three days after the creation of the project.

RQ2: What are the characteristics of repositories on which TODO Bot operates?

We report the characteristics of repositories on which TODO Bot operates regarding the number of stars, forks, and watchers , commits, and TODO issues.

Stars, Forks, and Watchers: Our analysis reveals that 1,990 out of 2,208 repositories have less than 25 stars, forks and watchers. Moreover, around half of the repositories have no stars or forks at all. Similarly, more than half of the repositories only have one watcher. Given that GitHub provides functionality to automatically watch a self-created repository, it seems reasonable that most repositories have one watcher as opposed to zero watchers.

Commits: More than half of the repositories have less than 50 commits. Figure 1 presents the number of repositories having a certain number of commits; repositories with at least 2K commits are excluded from Figure 1 for readability, but they only account for 60 out of 2,208 repositories. The fact that TODO Bot mostly operates on small repositories is not surprising as according to Kalliamvakou *et al.* [6], more than 90% of GitHub projects have less than 50 commits.

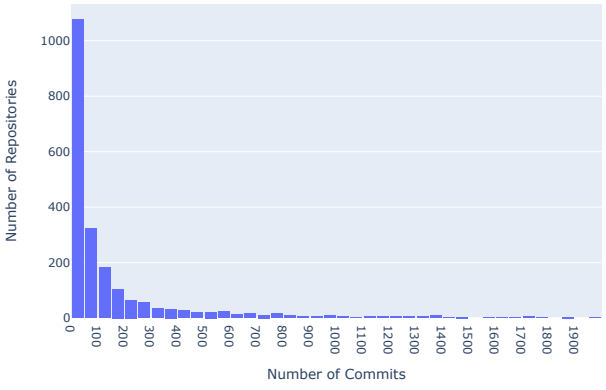

Figure 1: Number of Commits per repository.

TODO issues: As depicted in Figure 2, only few repositories have more than 50 TODO issues. Figure 3 shows the total number of

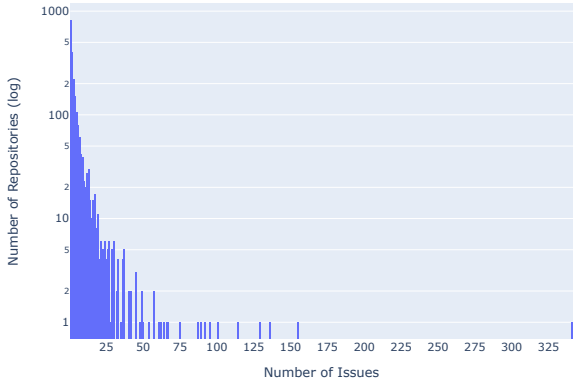

Figure 2: Number of TODO issues per repository.

newly created TODO issues across all repositories per month. From October 2017 to November 2018, TODO Bot only created around 50 issues each month. This changed in the period from November 2018 to June 2019, where the bot usage steadily grew over time. In the period thereafter (until September 2020), this growth halted. Note that this data was fetched on 6 September 2020, meaning that the last bin (September 2020 to October 2020) contains only data for six days. This means that it took TODO Bot around a year after its release before its usage actually started to grow, and that this growth continued for around half a year.

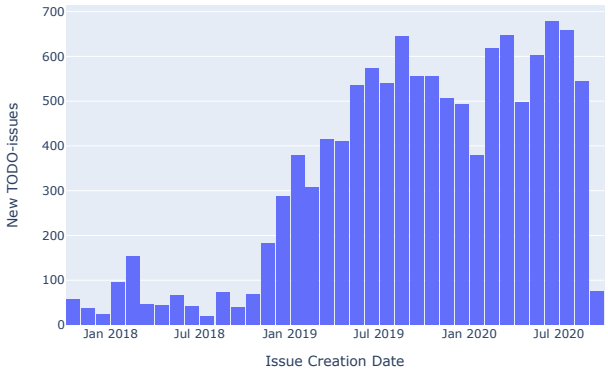

Figure 3: Number of TODO issues per month.

RQ3: How does the adoption of the TODO Bot affect the introduction of TODO comments and dealing with them?

As we established earlier, most repositories only had few TODO comments that were identified by TODO Bot. For the sake of comparison, Figure 4 illustrates the amount of repositories that have created a certain number TODO comments before TODO Bot was introduced in their repository. Repositories with one or no commits before the bot was introduced are not included in this graph, as they cannot have TODO comments without issues.

While we can see a similar distribution as for TODO comments introduced after TODO Bot was adopted (recall that each TODO

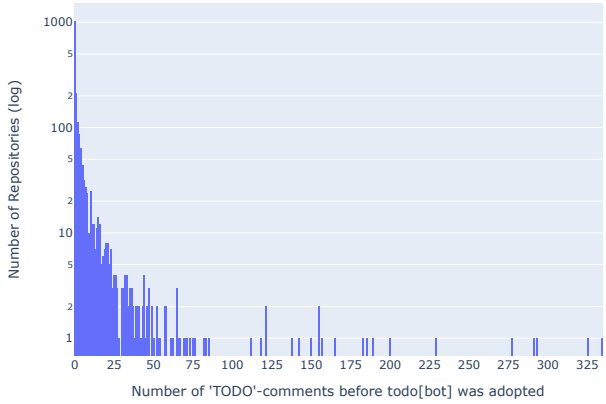

Figure 4: Number of TODO comments before `TODO Bot` was adopted.

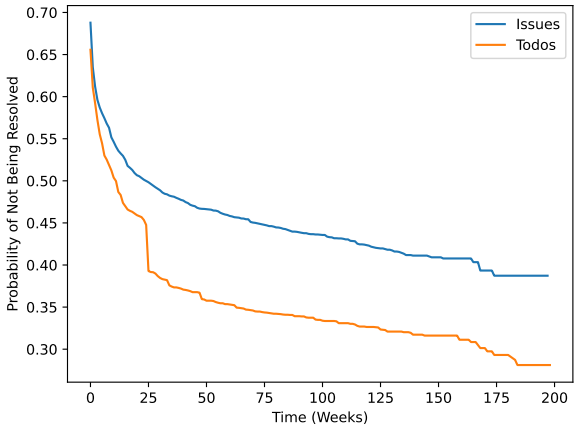

Figure 5: The probability of TODO comments and issues not being resolved after a certain amount of time

comment has an associated issue), this can be because of two reasons. Firstly, repositories might not have commits where `TODO Bot` was not already active (and hence, the bot identified all TODO comments that were made throughout the entire repository history). Alternatively, repositories might have made commits, but those commits did not contain TODO comments.

To investigate whether the `TODO Bot` affected the way developers use TODO comments, we analyzed the period of one month before and after the first issue created by the bot. We identified a total of 591 repositories which had at least one commit with a TODO comment one month before the first issue created by the bot, and also with issues created by the bot one month after its first issue. As the data is not normally distributed ($p < 2.2 \times 10^{-16}$ for both distributions) we perform a paired Wilcoxon test [2]; the p-value of 0.0192 indicates that the number of issues after the bot adoption is statistical higher then before, but with a small effect size (0.0979).

As our final assessment, we conducted another study, aiming to understand whether the adoption of the `TODO Bot` encourages developers to address TODO comments faster. We obtain the lifespan of TODO comments and compare them with the lifespan of issues. We consider a TODO comment introduced in a commit resolved if there is a subsequent commit in which the TODO comment is deleted. The time difference between these two commits is considered as the TODO comment's lifespan. We extract 6,364 TODO comments from the source code and calculate the time difference between the commits adding and removing TODO comments. Figure 5 shows the result of Survival Analysis on those comments and also the issues. The results show that for periods of over 5 weeks, the probability of not being addressed is lower for TODO comments compared to issues. Also, the Wilcoxon [2] test results in a p-value of 0.004 and an effect size of 0.15, which is not significant. The exact reasons for this behavior need to be investigated in future work, however, we can argue that the lifespan of TODO comments and issues are heavily dependent on some factors like the complexity of their objectives, which we did not analyze in this article. Nevertheless, the `TODO Bot` remains successful in increasing the visibility of technical debts.

5 THREATS TO VALIDITY

Internal Validity. When fetching issues that `TODO Bot` would have created before it was introduced, only the keywords "todo" and "@todo" (case insensitive) are considered, meaning that TODO comments introduced by custom-defined keywords are not identified. Given that "TODO" is the most commonly used type of task-annotation by a large margin [14], we do not expect our results to be invalidated. The `TODO Bot` also creates Pull Request comments, which have not been regarded in this research, meaning the repositories which fix all TODO comments before merging are not considered. Besides, the issues created by the `TODO Bot` at the beginning of its adoption might have been for the sole purpose of experimenting with the bot's functionalities, which has not been considered in this work.

External Validity. Only open-source projects on GitHub and `TODO Bot` were studied. Thus, we cannot claim our results hold for different bots with the similar functionality nor to non open-source projects.

6 CONCLUSION

In this study, we investigate the characteristics of the projects which adopted the `TODO Bot` in order to support the management of technical debt. We also investigated whether the adoption of `TODO Bot` encourages the use of TODO comments by developers. We observed the `TODO Bot` more popular among more recently created projects. Furthermore, the initial analysis showed the number of TODO comments increased after the bot adoption. Although `TODO Bot` seems to be encouraging developers to use more TODO comments, it does not make the process of addressing them faster.

This work provides the basis for future research on `TODO Bot`. To find out *effects* of `TODO Bot` one can perform a deeper analysis on the lifespan of TODO comments and issues considering the complexity of their objectives. The reasons why the owners of major repositories are not interested in such bots as `TODO Bot` can also be investigated.

REFERENCES

[1] Ahmad Abdellatif, Khaled Badran, and Emad Shihab. [n. d.]. A Repository of Research Articles on Software Bots. http://papers.botse.org.

[2] David F. Bauer. 1972. Constructing Confidence Sets Using Rank Statistics. *J. Amer. Statist. Assoc.* 67, 339 (1972), 687–690. http://www.jstor.org/stable/2284469

[3] Ward Cunningham. 1992. The WyCash Portfolio Management System. In *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum)* (Vancouver, British Columbia, Canada) *(OOPSLA '92)*. Association for Computing Machinery, New York, NY, USA, 29–30. https://doi.org/10.1145/157709.157715

[4] Jason Etcovitch. [n. d.]. *todo*. https://github.com/apps/todo

[5] Martina Iammarino, Fiorella Zampetti, Lerina Aversano, and Massimiliano Di Penta. 2021. An empirical study on the co-occurrence between refactoring actions and Self-Admitted Technical Debt removal. *JSS* 178 (2021), 110976.

[6] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *MSR*. ACM, 92–101.

[7] David G. Kleinbaum and Mitchel Klein. 2005. *Survival analysis: A self-learning text*. Springer New York.

[8] Everton Da S. Maldonado, Rabe Abdalkareem, Emad Shihab, and Alexander Serebrenik. 2017. An Empirical Study on the Removal of Self-Admitted Technical Debt. In *ICSME*. 238–248.

[9] Pengyu Nie, Rishabh Rai, Junyi Jessy Li, Sarfraz Khurshid, Raymond J. Mooney, and Milos Gligoric. 2019. A Framework for Writing Trigger-Action Todo Comments in Executable Format. In *ESEC/FSE*. ACM, 385–396.

[10] Aniket Potdar and Emad Shihab. 2014. An Exploratory Study on Self-Admitted Technical Debt. In *ICSME*. 91–100.

[11] J. P. Royston. 1982. An Extension of Shapiro and Wilk's W Test for Normality to Large Samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31, 2 (1982), 115–124.

[12] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. PyDriller: Python framework for mining software repositories. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. ACM Press, New York, New York, USA, 908–911. https://doi.org/10.1145/3236024.3264598

[13] Giriprasad Sridhara. 2016. Automatically Detecting the Up-To-Date Status of ToDo Comments in Java Programs. In *India Software Engineering Conference*. ACM, 16–25.

[14] Margaret-Anne Storey, Jody Ryall, R. Ian Bull, Del Myers, and Janice Singer. 2008. TODO or to Bug: Exploring How Task Annotations Play a Role in the Work Practices of Software Developers. In *ICSE*. ACM, 251–260.

[15] Student. 1908. The probable error of a mean. *Biometrika* (1908), 1–25.