What Is the Relationship between Code Review Direct Approval Rate and Release Deadline?

Zhimao Lin
Department of Computing Science
University of Alberta
Edmonton, Canada
zhimao@ualberta.ca

Shuonan Pei
Department of Computing Science
University of Alberta
Edmonton, Canada
spei@ualberta.ca

***Abstract*—Code review is a software quality assurance process that helps software companies improve the quality of the code written by their software developers. It becomes a standard procedure of software development in the industry these days. However, code reviewers may not have time to review the code line by line when it is closed to the release deadline. Due to the deadline pressure, software developers may just simply click the approval button without a careful review. In this paper, we find out if code reviewers tend to approve the code reviews without actually reviewing them in detail. Then, we speculate on the potential reasons for the phenomenon that we find.**

Keywords— Code review, code review approval rate, release deadline, relationship

I. Introduction

Code review is a software quality assurance process that helps software companies improve the quality of the code written by their software developers. It becomes a standard procedure of software development in the industry these days [1]. There is no doubt that code review can improve the code quality; However, we suspect that code reviewers may approve the code reviews without an actual careful review when they are under the release deadline pressure. Most companies do not have a pure code review team who work on nothing but review code. Thus, code review requests are usually sent to peer software developers within the same team who also need to develop new features [2]. When the release deadline is approaching, those developers do not only need to complete the new features assigned to them, but also have to review the code written by other developers. Under the pressure of the release deadline, these developers may tend to simply approve the incoming code reviews without any actual detailed review in order to complete the new features and meet the deadline. The main goal of this paper is to help software companies to identify if this phenomenon actually exists in reality. If this phenomenon exists, managers can take some actions and take full advantage of the code review process to actually ensure the quality of their software.

This paper aims to answer the following research questions:

RQ1: Is there any relationship between code review direct approval rate and release deadline?

RQ2: What are the potential reasons for the occurrence of the phenomenon that we find?

In order to answer RQ1, we analyzed the code review data of 10 repositories from CROP website [3]. We extracted and calculated the code review direct approval rate of each day for each repository. Then, we mined the release dates for each repository using PyGithub in order to identify which dates are closed to release dates [4]. After that, for each repository, we first separated code review direct approval rate according to whether its date is closed to the nearest release date. Secondly, we drew and compared the boxplot of those two groups. We also conducted a Wilcoxon signed-rank test [5] and Logistics Regression [6] between these two groups in order to compare the difference between these two distributions numerically.

In addition, we drew a line chart of the code review direct approval rate of each year for each repository. We want to find if there is a pattern such that the code review direct approval rate increases when it is closed to release dates and decreases immediately after the release dates.

In the end, we calculated the mean of the code review direct approval rate when it is closed to release dates and when it is not closed to release dates for each repository. Then, we conducted paired Wilcoxon signed-rank test to find out the difference between the mean of those two groups across all 10 repositories.

Finally, we found there is no relationship between code review direct approval rate and release date, but there are few repositories indicate a significant difference between the code review direct approval rate of the code reviews closed to release dates and those not closed to release dates. Also, only a few line charts show some kind of trend that the code review direct approval rate increases while approaching release dates and decreases after release dates.

In order to answer RQ2, we speculate the reasons based on their GitHub repositories. We manually inspected their repositories and found that the code review direct approval rate may have some relationship with team size and project size.

This paper has the following contributions:

- It researches the relationship between code review directly approval rate and release deadline. To our knowledge, there have not been any research studies the relationship between code review and release deadline.

- It conducts statistic tests and provides numerical results of the relationship between code review directly approval rate and release deadline.

The structure of this paper is as follows: Section 2 provides the necessary background and terminologies. Section 3 outlines our study design, including the hypothesis, data parsing, output generation, and plotting. Section 4 presents the results and Section 5 discusses lessons learned. In Section 6 we describe the limitations and threats to validity. We finalize the paper with the discussion of related work in Section 7 and conclusions in Section 8.

II. Background and Terminology

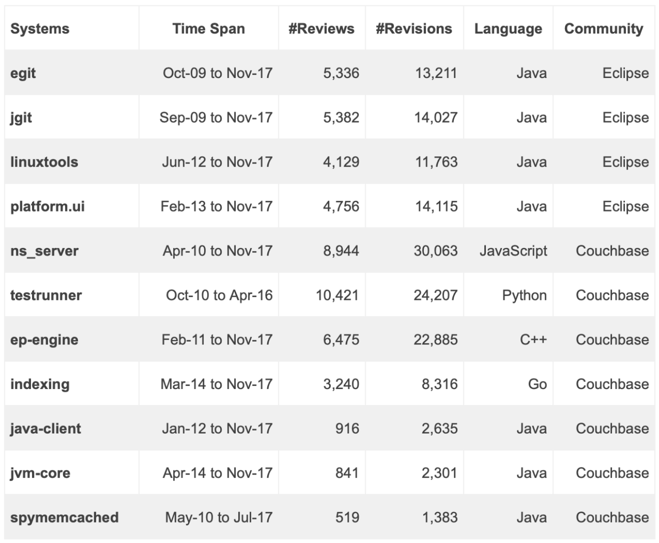
In this section, we introduce the code review process and some terminologies that are used in this paper.

A. *Code Review*

Code review is a software quality assurance activity in which developers ask their peers to identify defects mainly by manually reading their code change. As Figure 1 shown below [7], an author sends his initial code to some reviewers. Then, reviewers can decide if they want to accept the code changes or not. They can approve the code changes if there is no problem with the code changes. If there are some minor problems, they can approve the code changes with comments. However, if the code changes have major problems, they have to reject the code changes. The author can only commit his code changes when the code review is fully approved. Otherwise, he has to rework on the code changes which are returned by reviewers with constructive comments until they are satisfied.

B. *CROP Dataset*

The dataset that we analyzed is provided by CROP. It contains the code review records of 11 repositories written by Eclipse and Couchbase. Fig. 1 lists some general information of these 11 repositories [3].



Systems	Time Span	#Reviews	#Revisions	Language	Community
egit	Oct-09 to Nov-17	5,336	13,211	Java	Eclipse
jgit	Sep-09 to Nov-17	5,382	14,027	Java	Eclipse
linuxtools	Jun-12 to Nov-17	4,129	11,763	Java	Eclipse
platform.ui	Feb-13 to Nov-17	4,756	14,115	Java	Eclipse
ns_server	Apr-10 to Nov-17	8,944	30,063	JavaScript	Couchbase
testrunner	Oct-10 to Apr-16	10,421	24,207	Python	Couchbase
ep-engine	Feb-11 to Nov-17	6,475	22,885	C++	Couchbase
indexing	Mar-14 to Nov-17	3,240	8,316	Go	Couchbase
java-client	Jan-12 to Nov-17	916	2,635	Java	Couchbase
jvm-core	Apr-14 to Nov-17	841	2,301	Java	Couchbase
spymemcached	May-10 to Jul-17	519	1,383	Java	Couchbase

Fig. 1. Information of those 11 Repositories on CROP

We only used 10 of them except for indexing repository because it does not have any releases on its GitHub page.

C. *Terminologies*

- **Code review is directly approved (DA):** The code review request has been approved and merged right away without any revision required by reviewers.

- **Code review is approved with revision (AWR):** The code review request has been approved and merged with some revision required by reviewers.

- **Code review is rejected (R):** The code review request has been rejected by reviewers, and the author cannot merge his changeset into the source code repository.

- **Release date:** In this paper, we use the commit date of the last commit of each release as the release date. The reason for using the last commit date instead of the actual release date is that many releases actually happen a few days after the last commit according to our dataset. However, the last commit date is the actual deadline for software developers.

- **Close date:** We define the close date as the date on which the code review is finally approved or rejected.

- **Closed to release dates:** If the close date of a code review falls in a time period, which includes the release day, before its nearest release date, we say the code review is closed to release date. In this paper, we set the time period before its nearest release date to 2 days or 3 days. For example, we set the time period to 2 days, and the release date is April 10th, 2019. The code reviews closed on April 9th or 10th, 2019 will be considered they are closed to release date. The code review closed on April 8th, 2019 will be considered it is not closed to release date. We denote "CTRD" as closed to release dates, and "NCTRD" as not closed to release dates.

- **Code review Direct approval rate per day (DAR):** We group all kind of code review by date, and then we use the following formula to calculate the Code review Direct approval rate per day.

$$DAR = \frac{The\ number\ of\ DA\ per\ day}{Total\ number\ of\ code\ reviews\ per\ day}$$

- **DAR(CTRD):** It represents a collection of code review direct approval rate of the days that are closed to release dates.

- **DAR(NCTRD):** It represents a collection of code review direct approval rate of the days that are not closed to release dates.

III. Study Design

In this section, we first introduce our hypothesis. Then, we describe how we extract useful data from CROP dataset and calculate the result that is needed to answer RQ1. We also mention the method that we use for RQ2 to speculate the reason for the phenomenon found in RQ1.

A. Hypothesis

If we can find the code review direct approval rate is very high when it is closed to release date, or there is a pattern in the code review direct approval rate line chart such that the rate increases before release date and decreases afterward, then we can say there is a relationship between code review direct approval rate and release deadline.

B. Data Cleaning and Categorize Code Reviews

CROP dataset contains a metadata folder and a discussion folder. There is a comma-separated value (CSV) file [8] for each repository in the metadata folder. The CSV file lists code review number, revision number, author, status, URL of a code review, etc. of all code reviews occurred in a period of time mentioned on the CROP website. We sampled some code reviews from different repositories and manually inspected the code reviews by going to their webpage using the URL provided. We found that a code review is directly approved if its maximum revision number is less or equal 2, and its status is "MERGED." A code review is approved with revision if its maximum revision number is greater than 2, and its status is "MERGED." Furthermore, if the status of a code review is "Rejected," then the code review is definitely rejected. Thus, we grouped the data by code review number and found the maximum revision number and status for each code review. TABLE I illustrates it in detail. Based on this information we could identify if the code review is directly approved, approved with revision, or rejected.

TABLE I. CATEGORIZE CODE REVIEWS

Code Review Category	Maximum Revision Number	Status
DA	Less or equal to 2	MERGED
AWR	Greater than 2	MERGED
R	-	Rejected

In the discussion folder, there is a folder for each repository. In that folder, there is a folder for each code review of that repository, which is named after the code review number. Inside that folder, there is a text file for each revision of the code review, which records the discussion of the revision. Based on the result of the previous step, we can get the code review number of each code review. With those numbers, we can direct to its corresponding folder, which contains the discussion text file of that code review. Thus, we can get the close date of the code review by looking at the date in its last revision discussion text file.

In this way, we extracted the code review number, maximum revision number, status, author, URL, close date, and close time for each repository. Meanwhile, we deleted all the rows containing NA or invalid data since we had noticed some rows are missing values, and some rows do not have valid authors or URLs during our inspection. Finally, we saved the extracted data of each repository into a CSV file.

C. Acquire Release Date

Next, we acquired the release dates, which is defined in section 2, of each repository using PyGithub. PyGithub is like a wrapper of GitHub API [9]. It gets the release dates by calling GitHub API. However, the maximum number of API calls per hour is allowed by GitHub API is 5000 for an authenticated user, which makes this step the most time-consuming step. Thus, we had to first acquire the release dates for the 9 repositories except for eclipse.platform.ui since it has over 6000 releases. Secondly, we retrieved the first 4500 release dates of eclipse.platform.ui after an hour. Thirdly, we retrieved the rest release dates after another one-hour cooling down time. In the end, we saved the release dates of each repository as a CSV file for the next step.

D. Group by and Statistic Test

Based on the result from previous steps, we grouped our code review data by date so that we could calculate the daily direct approval rate. We calculated the number of code review in each category (DA, AWR, and R) and the total number of code review on each day. Next, we switched row and column, putting the close date on the first column and different kind of code review number beside the close date. We calculated the direct approval rate base on the formula in section 2. We also created a new column which stores Booleans represent whether the close date is close to the release date or not. The Python program read the release date files from the previous step and compared the release date with the close date. If the close date is closed to release date we set it to "True" otherwise we set it to "False". Finally, we saved this table in a CSV file and outputted it to our result folder.

Based on the previous result, we finally can run Wilcoxon signed-rank test and logistics regression for each repository in order to show if there is a significant difference between the code review direct approval rate of code reviews closed to release dates and those not closed to release dates. We used Wilcoxon signed-rank test instead of Student's t-test [10] because Student's t-test requires the assumption that code review direct approval rate is normally distributed. After we ran those tests for each repository, we calculated the mean of the DAR(CTRD) and DAR(NCTRD) for each repository. Then, we conducted a paired Wilcoxon signed-rank test across all 10 repositories to find if there is any significant difference. In this paper, we use 0.01 as our significance level. In other words, if the p-value is less than 0.01, we can conclude that there is significant evidence shows that there is a significant difference between the distribution of DAR(CTRD) and DAR(NCTRD).

E. Draw Boxplot and Line Chart

After we got the result from the previous step, we drew a boxplot for each repository. The boxplot compares the distribution of DAR(CTRD) and DAR(NCTRD). We also drew line charts, which illustrate the trend of code review direct approval rate as time passes by for each repository, and we used a red vertical line to mark the release dates. Since the line chart of each repository is too long, we divided it by the year. So, for each repository, we have a line chart of code review direct approval rate for each year the repository spans. Just below each line chart of approval rate, we drew a sub-line-chart with the same x-axis to show the trend of the number of daily code reviews during that time period. The reason for adding the sub-line-charts is that we also want to take the number of daily code reviews into consideration. For example, if we get a 100% code review direct approval rate on a day, but there is only one code review on that day. The 100% does not necessarily mean that most code reviews are directly approved during that time period since the denominator is too small.

Once we have those boxplots and line charts, we can use our hypothesis in section 3(A) to answer RQ1. If the boxplots show a significant difference between the distribution of the code review direct approval rate between code reviews closed to release dates and those not closed to release dates, and the distribution of the code reviews closed to release dates mainly falls between 0.8 and 1, we can imply that there is a relationship between code review direct approval rate and release deadline. As for the line charts, if we can find a pattern as mentioned in section 3(A), we can also imply the relationship exists.

F. *Speculate the Reasons for the Occurrence of the Phenomenon*

We tried to explore the nature of this phenomenon by manually checking their GitHub repositories and code review sources. We have looked at the contributors' tag on GitHub to identify how many people have worked on this project and their contributions to the project. We also inspected the code review webpages to find out the participants of the code review and what kind of feedback they have provided.

G. *Expected Result*

We expected the result would show some relationship between the code review direct approval rate and the release deadline. We thought the boxplots of most repositories can show some difference between the distribution of DAR(CTRD) and DAR(NCTRD). We also expected that the pattern mentioned in section 3(A) exists in the most line charts. Hopefully, the p-values of the most statistic tests are less than 0.01.

IV. Result

In this section, we show the result of the research questions mentioned in section 1.

A. *RQ1: Is there a relationship between the code review direct approval rate and release deadline?*

After running the Python programs and R scripts, we found there is no relationship between code review direct approval rate and release deadline. TABLE II below shows the p-value of our Wilcoxon signed-rank test and logistics regression.

TABLE II. Results of Statistic Tests

Repository Name	P-value of Wilcoxon signed-rank test	P-value of Logistics Regression test
couchbase-java-client	0.9319	0.908
couchbase-jvm-core	4.581e-07	2.65e-06
eclipse.platform.ui	0.001038	0.00149
egit	0.2511	0.2136
ep-engine	0.3888	0.568
jgit	0.735	0.919
linuxtools	0.01632	0.0131
ns_server	0.5736	0.606
spymemcached	0.3791	0.365
testrunner	0.005097	0.00164

Only the p-values of couchbase-jvm-core, eclipse.platform.ui, and testrunner are below 0.01, which means there is a significant difference between the distribution of DAR(CTRD) and DAR(NCTRD) of these three repositories. For other repositories, there is no strong evidence shows there is a significant difference between these distributions.

The boxplot results confirm the statistic result that we got. Only the boxplots of couchbase-jvm-core, eclipse.platform.ui, and testrunner (Fig. 2-4) shows a significant difference between the distribution of DAR(CTRD) and DAR(NCTRD) of these three repositories.

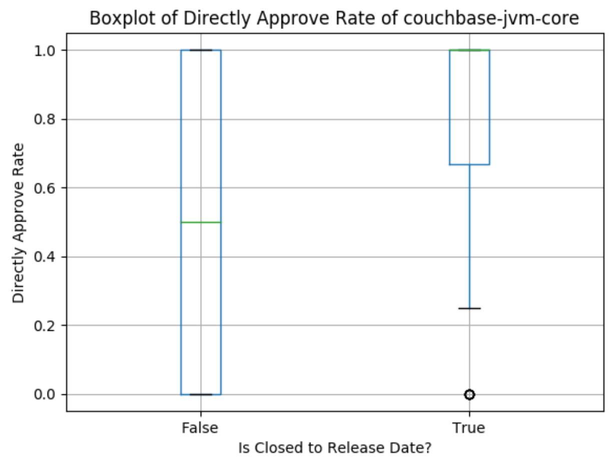


Fig. 2. Boxplot of couchbase-jvm-core

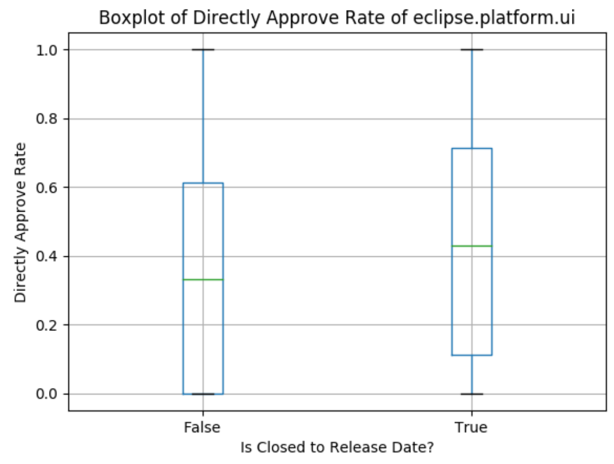


Fig. 3. Boxplot of eclipse.platform.ui

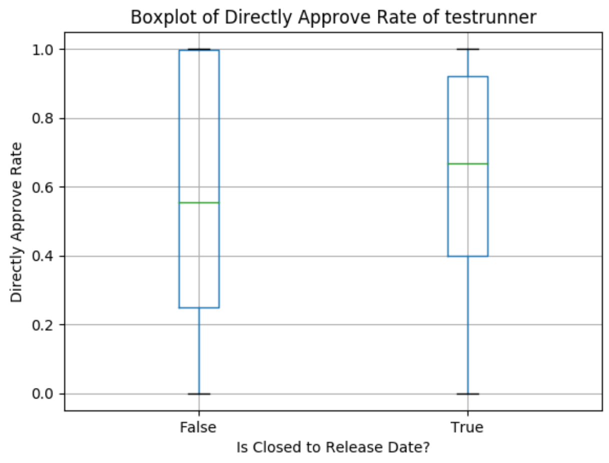


Fig. 4. Boxplot of testrunner

In addition to the boxplot, the result of the paired Wilcoxon signed-rank test also shows there is no difference between those distributions of each repository. The p-value that we got is 0.08281, which is greater than 0.01. Therefore, we conclude in general, there is no significant difference between the distributions of DAR(CTRD) and DAR(NCTRD).

As for the line charts, we only found one-line chart that clearly shows the pattern mentioned in sections 3(A), which is the line chart of couchbase-jvm-core in 2016. As Fig. 5 shown below, there is an increase of DAR before each release date and a decrease of DAR after each release date. Also, there is a reasonable number of code reviews around each release during the first half of the year.

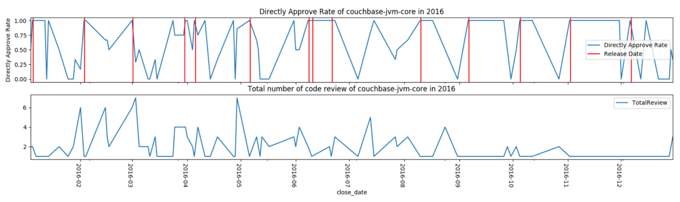


Fig. 5. Line chart of couchbase-jvm-core in 2016

Overall, only a few repositories show some difference in distribution. Therefore, we conclude that there is no relationship between the code review direct approval rate and release deadline.

B. *RQ2: What are the potential reasons for the occurrence of the phenomenon that we find?*

According to the previous conclusion, most teams do not change their direct approval rate when approaching the release dates. For example, the distributions of DAR(CTRD) and DAR(NCTRD) for ns_server are very similar. There are usually at least two reviewers reviewing code and sometimes there are six people reviewing one commit. According to our result table, it has 4135 direct approve code reviews out of 8943 total code reviews. Therefore, this project does not have a high direct approval rate when approaching the release dates.

On the contrary, couchbase-jvm-core shows the strongest relationship between code review direct approval rate and release deadline. This repository is extremely small, which is about 9 MB. There are only two major programmers developing this project. The committer and reviewer were the same people at the very beginning. After a while, a second developer joined this team and they started to review each other's code changes. We believe these two developers know each other very well and confident with their coding competence. The data set also prove our suspect. There are 516 direct approve code reviews out of 841 total code reviews.

V. Discussion

Overall, we did not find any strong evidence shows any relationship between the code review direct approval rate and release deadline. We even tried a 3-day time period to identify whether a code review is closed to release date. However, we got similar results as the 2-day time period. Therefore, we think the code review process is still a reliable and effective way to ensure the quality of the software especially in a big team developing a large project. In this situation, code review is definitely a necessary process to prevent developers from committing defective code. Once the defects get into production, it will be very costly to fix them. So, the code review process can help software companies save a lot of money.

VI. Limitations and Threats to Validity

The main threat to the validity of our result is that we only analyzed 10 repositories, which are only written by 2 communities since CROP only provides code review records of 11 repositories from Eclipse and Couchbase. One of them even does not have any releases so that we can only use the other 10 repositories as our dataset. The limitation of our data source may impact the generality of our results. If we had a large enough dataset contains projects with various sizes and authors, then our sample data could represent the population much better, and our result could become more generic.

In addition to the limitation of the data source, manually reviewing each repository and speculating the answer to RQ2 cannot provide very strong evidence to support the result of RQ2. If we could contact the software developers and the team leaders of those projects and conduct some survey or interview, then we did not need to speculate the reason for the phenomenon.

VII. Related Work

There are many empirical studies that propose a software template to improve software quality. For example, Patwardhan [2] provided a structured unit testable template, which is a coding method, to enforce code review standards. It provided a guided approach towards a reliable and efficient code review process. Base on his result, the templates prominently improved the code quality and code review process efficiency.

However, Bacchelli and Bird [11] found that the development of the code review tools does not necessarily lead to identifying more defects especially contextual issues. They mentioned that the understanding of context and code change is a huge challenge for developers during code review. It has a direct impact on the quality of the code review. They recommended that team members should improve their code reading skills, and authors should include code owners and some other developers who have a comprehensive understanding of the code.

VIII. Conclusion and Future Work

The code review process helps software companies to ensure the quality of their products. However, we suspect that code reviewers may simply approve the code reviews without an actual review when they are under the pressure of the release deadline. This paper helps software companies to identify if this phenomenon actually exists in reality. We tried to find whether the phenomenon exists by answering if there is a relationship between the code review direct approval rate and release deadline. After we conducted some data analysis and statistic tests, we found there is no relationship between them.

The main contribution of our work is to demonstrate the effectiveness of the code review process cannot be impacted by the pressure of the release deadline. This paper provides numerical results of the relationship between code review direct approval rate by conducting statistic tests. As for future work, researchers can research if the code review process can actually save some cost of software companies. With the consideration of the cost of software developers' salary and the cost of fixing bugs in the production, how much code review process can actually save for software companies. Furthermore, they can also research if the code review process is worthwhile in a research project with a small team and how much a research lab can benefit from this process.

Acknowledgment

Thanks to CROP website, we can have access to the dataset contains the code review information of the 10 repositories. We also thank L. Qiu, Y. Wang, and J. Rubin [12] for setting up a good example of research paper structure in "Analyzing the analyzers: FlowDroid/IccTA, AmanDroid, and DroidSafe" [12]. We learned how to write a research paper with a good structure from them.

References

[1] S. Mcintosh, Y. Kamei, B. Adams, and A. E. Hassan, "The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects," *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 2014.

[2] A. S. Patwardhan, "Structured unit testable templated code for efficient code review process," 2016.

[3] "Code Review Open Platform," *Code Review Open Platform*. [Online]. Available: https://crop-repo.github.io/. [Accessed: 16-Apr-2019].

[4] "PyGithub¶," *PyGithub*. [Online]. Available: https://pygithub.readthedocs.io/en/latest/index.html. [Accessed: 16-Apr-2019].

[5] "Wilcoxon signed-rank test," *Wikipedia*, 10-Apr-2019. [Online]. Available: https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test. [Accessed: 12-Apr-2019].

[6] "Logistic regression," *Wikipedia*, 06-Apr-2019. [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression. [Accessed: 12-Apr-2019].

[7] A. Manekovskiy, "Alexander Manekovskiy," *Why Your Team Should Do A Code Review On A Regular Basis - Alexander Manekovskiy*, 21-Oct-2015. [Online]. Available: http://amanek.com/why-your-team-should-do-a-peer-review-on-a-regular-basis/. [Accessed: 12-Apr-2019].

[8] "Comma-separated values," *Wikipedia*, 05-Apr-2019. [Online]. Available: https://en.wikipedia.org/wiki/Comma-separated_values. [Accessed: 12-Apr-2019].

[9] "GitHub API v3," *GitHub Developer*. [Online]. Available: https://developer.github.com/v3/. [Accessed: 16-Apr-2019].

[10] "Student's t-test," *Wikipedia*, 05-Apr-2019. [Online]. Available: https://en.wikipedia.org/wiki/Student's_t-test. [Accessed: 12-Apr-2019].

[11] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," *2013 35th International Conference on Software Engineering (ICSE)*, 2013.

[12] L. Qiu, Y. Wang, and J. Rubin, "Analyzing the analyzers: FlowDroid/IccTA, AmanDroid, and DroidSafe," *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2018*, 2018.