Evolution of the Practice of Software Testing in Java Projects

Anisha Islam*, Nipuni Tharushika Hewage†, Abdul Ali Bangash‡, and Abram Hindle§
Department of Computing Science, University of Alberta
Edmonton, AB, Canada
Email: *aislam4@ualberta.ca, †hewage@ualberta.ca, ‡bangash@ualberta.ca, §hindle1@ualberta.ca

***Abstract*—Software testing helps developers minimize bugs and errors in their code, improving the overall software quality. In 2013, Kochhar *et al.* analyzed 20,817 software projects in order to study how prevalent the practice of software testing is in open-source projects. They found that projects with more lines of code (LOC) and projects with more developers tend to have more test cases. Additionally, they found a weak positive correlation between the number of test cases and the number of bugs. Since the conclusions of a study might become irrelevant over time because of the latest practices in the relevant fields, in this paper, we investigate if these conclusions remain valid if we re-evaluate Kochhar *et al.*'s findings on the Java projects that were developed from 2012 to 2021. For evaluation, we use a random sample of 20,000 open-source Java projects each year. Our results show that Kochhar *et al.*'s conclusions regarding the projects with test cases having more LOC, the weak positive correlation between the number of test cases and authors, and the weak positive correlation between the number of test cases and bugs remain stable until 2021. Our study corroborates Kochhar *et al.*'s conclusions and helps developers refocus in light of the latest findings regarding the practice of software testing.**

***Index Terms*—Software Testing, Replication Study, WoC, Java Projects**

I. Introduction

Software testing is essential because it helps ensure the functionality [1], reliability [2], and security of a software system [3], leading to higher quality and user satisfaction. Lack of software testing can leave unhandled bugs in the system that could lead to severe financial issues and wasted resources [4]. Since software testing is a resource-consuming task [5], [6], researchers are continuously looking for ways to help developers test their code effectively. Besides, developers also need the latest information on software testing trends to recognize potential sources of vulnerability in the system and concentrate on relevant details while developing tools.

To understand the practice of software testing in open-source projects, in one of the previous studies, Kochhar *et al.* [4] investigated how software testing is adopted on GitHub projects using software project data from 2012 [4]. The goal of their study was to determine the popularity of software testing in open-source projects and to examine various aspects of software development related to testing [4]. Among the findings, Kochhar *et al.* showed that the project size, the count of authors, and the bug count positively correlate with the number of test cases [4].

However, over the years, the software community has undergone changes, for which the results of a study might become irrelevant as per the new practices. To ensure that the findings of different studies remain consistent, they should be re-evaluated over time to uncover any changes in conclusions due to the changing environment since the original studies were conducted [7], [8]. By validating the stability of the conclusions of a study, developers can determine which areas require further improvement and whether their focus should shift in accordance with the updated results.

In this study, we aim to reproduce some of the research questions of Kochhar *et al.* by using open-source Java projects on GitHub from 2012 to 2021. To observe the trend in software testing practices and how they have evolved in the past decade, we validate the stability of Kochhar *et al.*'s findings over time. We investigate the recent relationship between different project entities and test cases. Our study can help developers re-adjust their understanding of how Java projects on open-source platforms adopt software testing. We will refer to Kochhar *et al.* [4] as KochharStudy [4] from here onwards.

We re-investigate three research questions in this paper:

RQ1: "How many projects have test cases?"
RQ2: "Does the number of developers affect the number of test cases present in a project?"
RQ3: "Does the presence of test cases correlate with the number of bugs?"

— *Quoted from* KochharStudy [4].

To find out if the conclusions of the first three research questions of KochharStudy [4] have been stable over the years, we used the World of Code (WoC) [9] infrastructure provided by the MSR 2023 [10] organizers to extract the names of the Java projects on GitHub from 2012 to 2021. Then we used Git [11] commands to clone the repositories and to extract relevant project information required for the research questions mentioned above. We took a random sample of 20,000 Java projects per year, discarded projects with no lines of code, and only considered projects hosted on GitHub.

Although KochharStudy [4] considered projects from multiple programming languages, our study is limited to the Java projects on GitHub only as Java is one of the most popular and frequently used programming languages [12]. Furthermore, Java offers a wide range of popular testing libraries such as JUnit, TestNG, PowerMock, Mockito, and Hamcrest. Therefore, we decided to observe the latest trends in Java's software testing practices.

Our results indicate that all of the conclusions of the study conducted by KOCHHARSTUDY [4] regarding the research questions under consideration remain stable and unchanged from 2012 to 2021.

II. METHODOLOGY

WoC allows users to explore relationships between various software project entities such as author, blob, file, commit, and project through mappings. Using these mappings, we can get information about one software repository entity by using the other. For example, the `c2P` mapping of WoC provides users with the project names (after fork normalization) associated with a particular commit id [13].

For evaluation, we collected the names of the Java projects with at least one commit in a year for all the years starting from 2012 to 2021 using the `c2P` mapping of WoC. For example, for collecting the project names in 2015, we selected projects with at least one commit in the year 2015. If a project had commits in multiple years, then the project was considered separately for all those years. In this way, we created different samples of projects from 2012 to 2021. Finally, we randomly selected 20,000 projects per year from the set of projects. Then, we used the following approach to extract relevant information to address our research questions.

a) Calculating Lines of Code: To address the first research question of KOCHHARSTUDY [4], we extracted the lines of code (LOC) information for the projects using a different approach than KOCHHARSTUDY [4]. We used git commands to clone the repositories from GitHub, checked out to the last commit of a project in a particular year, and extracted the LOC of that project by using `git ls-files | xargs wc -l`. This command gives us the LOC of all files without spaces in their names. Since KOCHHARSTUDY [4] considered the LOC of the entire project, we also considered the LOC of the entire Java project rather than counting only the LOC of Java files. Then, we discarded projects with no lines of code and only considered projects hosted on GitHub.

Table I shows the number of projects each year in our sample after discarding unnecessary projects. Although KOCHHARSTUDY [4] considered projects with more than 500 LOC, our study did not consider this lower bound because projects of smaller sizes might contain test cases as well.

b) Identifying Test Files and Test Cases: In accordance with KOCHHARSTUDY [4], we identified the test files by searching for the word "**test**" in their names, disregarding the case [4]. Although KOCHHARSTUDY [4] did not differentiate test files and test cases, in our study, we consider them different. We define the term "test cases" as different methods in a Java test file with testing logic. Most of the Java testing frameworks, such as JUnit and TestNG, use the "**@Test**" keyword to identify the methods in test files as test cases [14], [15]. We followed this convention and extracted the test cases using the "@Test" annotation.

c) Calculating the Number of Authors: To reproduce the second research question of KOCHHARSTUDY [4], we extracted the author information for projects using the `git log --pretty="%an %ae"` command.

d) Calculating the Number of Bugs: KOCHHARSTUDY [4] used the GitHub issue tracker to get information about the issues and bugs [4]. However, the WoC does not have an issue tracker, so we decided to use a proxy: the count of those commit messages of a project which refer to possible bugs as the bug count like Borle *et al.* [16]. We modified the regular expression used in Borle *et al.* [16] to find the commit messages that reference bugs. We used the following regular expression case-insensitively: $(error|bug|issue|solv|fix)$. This expression covers a vast arena of words related to issues and bugs in a software system, such as **error**, **error**s, **bug**, **bug**s, de**bug**, de**bug**s, de**bug**ged, de**bug**ging, **issue**, **issue**s, re**solv**e, re**solv**es, re**solv**ed, re**solv**ing, **solv**e, **solv**es, **solv**ing, **solv**ed, **fix**, **fix**ing, **fix**ed, and **fix**es. We followed Borle *et al.* [16]'s assumption that any commit with these base words of the regex in the commit message could be equivalent to a bug report [16].

While we largely followed KOCHHARSTUDY [4]'s methodology, we made some changes in ours, which are listed in Table II.

III. RESULTS

To reproduce results of our research questions, we followed the same null hypotheses that KOCHHARSTUDY [4] assumed to address their research questions and used the same significance level, $\alpha = 0.05$. We used two different statistical measurements: the Mann-Whitney-Wilcoxon (MWW) test [18] and Spearman's rho (ρ) [19] that KOCHHARSTUDY [4] used for statistical analysis to determine the difference and correlation between datasets. For the calculation of the rho-values, we have only considered projects with test cases in our study. All the scripts and datasets used to generate these results are publicly available in our replication package [20].

A. Prevalence of Test Cases (RQ1)

To reproduce the research question: "How many projects have test cases?" [4], we calculated the percentage of projects having test files and test cases. We present the values in Table III and find that the percentage of test files and test cases generally showed an upward trend throughout the years, which implies an increase in the adoption of testing in Java projects on GitHub from 2012 to 2021.

Moreover, KOCHHARSTUDY [4] showed that "Projects with test cases are bigger in size than projects without test cases." [4]. Therefore, to validate this finding, we inspected the relationship between a project's number of test cases and size, which we represented by LOC following KOCHHARSTUDY [4]. Figure 1 illustrates the size of projects with and without test cases in our study. For all the years, we observe that the median LOC value for projects without test cases is lower than the median LOC value of projects that have test cases. This difference in LOC values, for the latest dataset until 2021, re-confirms that "projects with test cases are bigger in size than projects without test cases" [4].

TABLE I: Number of projects in our sample after applying the exclusion criteria

Year	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
Projects	14,665	14,353	14,922	15,317	16,208	15,785	14,964	15,476	16,924	17,862

TABLE II: Differences in methodology between KOCHHARSTUDY [4] and our study

	Our study	**KOCHHARSTUDY [4]**
Time range	2012 to 2021	2012
Number of projects	20,000 per year	20,817
Project collection	Used WoC	Used GitHub API
Calculating LOC	Git commands	SLOCCount [17] utility
Counting bugs	Used bug referencing commits	Used GitHub issue tracking system
Definition of test case	Calculated test cases using @Test annotation	Considered test files same as test cases
Programming language	Java	Multiple programming languages

TABLE III: The trend (in %) of projects having test cases and test files over the past decade (2012-2021)

Year	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
% of projects containing test files	48.37	45.99	44.65	55.38	57.87	59.97	62.87	61.17	60.43	58.64
% of projects containing test cases	29.06	27.58	24.81	28.24	42.65	47.60	50.92	50.70	50.31	48.96

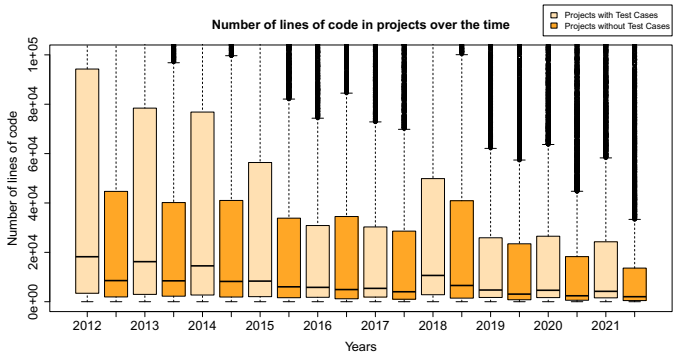

Fig. 1: Comparison of Lines of Code between Java Projects with and without Test Cases from 2012-2021

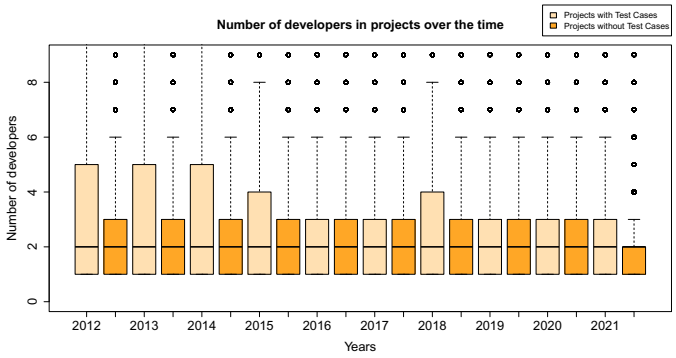

Fig. 2: Comparison of Number of Authors between Java Projects, with and without Test Cases, from 2012-2021

Additionally, KOCHHARSTUDY [4] used the MWW test to validate their conclusion and showed p-value < 2.2 e^{-16} [4]. In our study, for the MWW test, we observed p-values = 2.2 e^{-16} for all the years except 2016, when the p-value was 5.587 e^{-14}. This observation demonstrates a statistically significant difference between the LOC in projects with test cases and those without, which confirms the stability of KOCHHARSTUDY [4]'s conclusion from 2012 to 2021.

Furthermore, to find out how the project size and the number of test cases correlate, KOCHHARSTUDY [4] used Spearman's rho and reported $\rho = 0.427$ in their study [4]. In our case, the ρ values were between 0.26 and 0.37 (p-value = 2.2 e^{-16}) for all the years, which is similar to their reported value. Therefore, it is clear that their statement "there is a positive correlation between the number of test cases and the number of LOC" [4] is stable for all the years in our study.

KOCHHARSTUDY [4] further investigated the relationship between project size and the number of test cases per line of code. They observed a negative correlation in this scenario and reported Spearman's $\rho = -0.451$. We also observed negative rho values in the range of -0.60 to -0.68 with p-value = 2.2 e^{-16} (< α=0.05) in our study, which validates their finding that "the number of test cases per LOC decreases with increasing LOC" [4] for all years from 2012 to 2021.

> For the Java projects on GitHub from 2012 to 2021, the conclusions from KOCHHARSTUDY [4] that "Projects with test cases are bigger in size than projects without test cases." [4] and "the number of test cases per LOC decreases with increasing LOC" [4], remain stable through 2012-2021.

B. Correlation between Number of Developers and Test Cases (RQ2)

In their second research question, KOCHHARSTUDY [4] investigated, "Does the number of developers affect the number of test cases present in a project?" [4]. To re-investigate this, we counted the number of authors for each project in our sample for all the years. Figure 2 illustrates the number of authors in the Java projects.

We also manually investigated our data and identified some projects with zero developers. This could have been a technical issue, so we discarded such data points to ensure a valid number of authors for every project in our sample.

To confirm the relationship between the number of authors in projects that contain test cases and projects that do not, KOCHHARSTUDY [4] conducted the MWW test and observed p-value < $2.2\ e^{-16}$ [4]. In our study, the p-values were between $2.2\ e^{-16}$ and 0.01 for all the years except for 2017, when the p-value was 0.065 ($> \alpha$). As a result, we can say that the conclusion of KOCHHARSTUDY [4] for this particular scenario, "The results signify that the difference between these two sets is statistically significant" [4], is stable for all years from 2012 to 2021 with an exception in 2017.

KOCHHARSTUDY [4] also used Spearman's rho to measure the correlation between the number of authors and the number of test cases, giving $\rho = 0.207$. In our study, we observed the values between 0.216 and 0.338 (p-value = $2.2\ e^{-16}$) for all years, including 2017. Our result validates the finding that "there is a weak positive correlation between the number of developers and test cases" [4].

Furthermore, KOCHHARSTUDY [4] examined the correlation between the number of test cases per developer and the number of developers and observed a negative correlation ($\rho = -0.444$) [4]. Similarly, we observed Spearman's rho values between -0.11 and -0.269 with p-values = $2.2\ e^{-16}$ for all years except 2015, when the p-value = $9.473\ e^{-14}$, denoting a negative correlation. So, we can say that in our study, the observation of KOCHHARSTUDY [4] that "the number of test cases per developer decreases for the projects with more developers" [4], remains stable from 2012 to 2021.

The MWW test value is not valid for the Java projects in the year 2017. Possible reasons for this difference in results could be the differences in our experimental settings, methodology, and data sample. KOCHHARSTUDY [4] considered projects of various programming languages, whereas we only consider Java projects for our study. Also, our definition of test cases differs from KOCHHARSTUDY [4]. These reasons could have contributed to the differences in our observed results.

> For the Java projects on GitHub from 2012 to 2021, the conclusions from KOCHHARSTUDY [4] that "there is a weak positive correlation between the number of developers and test cases" [4] and "the number of test cases per developer decreases for the projects with more developers" [4], remain stable through 2012-2021.

C. Bug Count Correlation with Test Cases (RQ3)

In their third research question, KOCHHARSTUDY [4] investigated, "Does the presence of test cases correlate with the number of bugs?" [4]. To reproduce this research question, we calculated the number of bugs in a project using the procedure mentioned in Section II. Figure 3 shows the bug count in projects that contain test cases and those that do not.

KOCHHARSTUDY [4] initially assumed "with increase in the number of test cases, bug count increases" [4]. To confirm this assumption, they calculated Spearman's rho, which signified a weak positive correlation with a value of 0.181, [4]. In our study, We found the ρ values for all years to be between 0.25 and 0.417 (p-values < $2.2\ e^{-16}$). For this reason, we can say that the finding of KOCHHARSTUDY [4], "Projects having higher numbers of test cases observe an increase in the number of bugs, although the correlation is weak between them." [4], remains stable for all the years from 2012-2021.

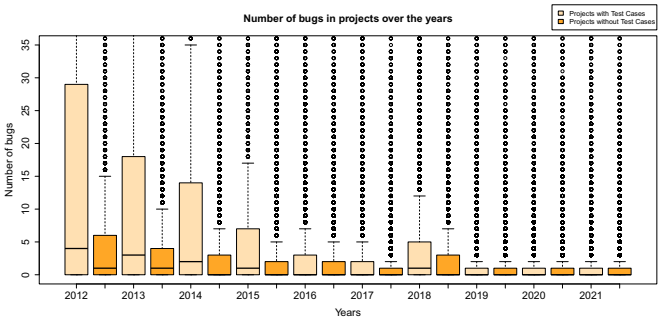

Fig. 3: Comparison of Number of Bugs between Java Projects, with and without Test Cases, from 2012-2021

We also manually investigated some cases where the bug count is high but the test cases count is low and vice versa. Three of the projects we explored had $6,040$, $9,841$, and $12,390$ bugs respectively, but our calculation did not show any test cases for these projects. We found that the tests of these projects were written in JavaScript, LiveCode, and Lua programming languages. As we only considered test cases of Java test files, our script ignored these test cases.

On the other hand, another project had $14,500$ test cases and one bug because most of the commit messages were in the format *apply <alphanumeric digits>*. The alphanumeric digits could mean a bug fix id or issue id internal to the project. As this pattern did not match our regex pattern for counting bugs from commit messages, our script could not identify these commits as references to bugs. This finding shows that the outlier data that defies the positive correlation between bugs and test cases could be caused by how we calculated bugs and test cases, which is different from KOCHHARSTUDY [4].

> For the Java projects on GitHub from 2012 to 2021, the conclusion from KOCHHARSTUDY [4] that "Projects having higher numbers of test cases observe an increase in the number of bugs, although the correlation is weak between them." [4] remains stable through 2012-2021.

IV. RELATED WORK

Previous studies have focused on the types of available software testing practices and tried to find a relationship between testing and various entities of a software project. In 2013, KOCHHARSTUDY [4] examined the correlation between the number of test cases and different project entities like size, number of authors, number of bugs, bug reporters, and programming languages in GitHub projects. Additionally, due to the significance of software testing in the software development lifecycle, Jangra *et al.* [5] explored different existing soft-

ware testing strategies and illustrated the connection between these strategies using a diagram. Bangash *et al.* [21] explored how stable the conclusions of defect prediction models are and found out that the defect prediction models performance remains inconclusive over time. Shepperd *et al.* [7] examined how likely it is for the replication studies to confirm with the original studies and found that replication studies with at least one common author from the original study are more likely to confirm the original findings. Our study is motivated by KOCHHARSTUDY [4]. We replicated a part of their work to re-investigate if their conclusions are still valid on the Java projects that were developed between 2012 and 2021.

V. THREATS TO VALIDITY

KOCHHARSTUDY [4] mentioned some limitations of their study, which are also valid in our case. For example, considering only a sample of the GitHub projects, which may not reflect the general behavior of all real-world projects, and using the "test" keyword to identify the test files, which may not identify all test files. Moreover, we used the *@Test* annotation to identify the test cases. However, there might be some testing frameworks that do not use the same annotation. In those cases, we might not have identified some test cases.

We also calculated the number of bugs from the commit messages using a regex pattern to map the commits to bugs. Nevertheless, not all commits use the regex pattern we used to reference bugs. For this reason, we might have missed some commits that mention bugs or issues differently. Finally, as collecting information about thousands of projects using git was a long-running query, some technical issues might have evaded our inspection.

VI. CONCLUSION

Software testing is paramount to the success and maintainability of software systems. KOCHHARSTUDY [4] explored the correlation of software testing with various software development entities like project size, number of authors, and number of bugs in an effort to understand the importance of software testing from a broader perspective. However, results or conclusions drawn in a study can become obsolete as a system or development culture changes over time, requiring that previous studies be re-evaluated to ensure that their conclusions remain accurate. For this reason, in this paper, we have reproduced the results of KOCHHARSTUDY [4]'s three research questions to validate if their conclusions still hold on the latest Java projects that were developed from 2012 to 2021. Our study confirms the validity of KOCHHARSTUDY [4]'s conclusions that there are more lines of code in projects with test cases, and a weak positive correlation exists between i) the number of test cases and the author count and ii) the number of test cases and the bug count in Java projects on GitHub till 2021. Additionally, our study allows developers to identify the latest trends in software testing and to re-adjust their knowledge accordingly to address issues that require more attention. In the future, researchers may improve their understanding of software testing across different programming languages and version control systems by leveraging the vast amount of software repository information and the advanced mappings available in the World of Code.

VII. ACKNOWLEDGEMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through their Discovery Grant. We are grateful for NSERC's commitment to supporting scientific innovation in Canada.

REFERENCES

[1] V. R. Basili and R. W. Selby, "Comparing the effectiveness of software testing strategies," *IEEE transactions on software engineering*, no. 12, pp. 1278–1296, 1987.

[2] R. H. Rosero, O. S. Gómez, and G. Rodríguez, "15 years of software regression testing techniques—a survey," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 05, pp. 675–689, 2016.

[3] B. Potter and G. McGraw, "Software security testing," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 81–85, 2004.

[4] P. S. Kochhar, T. F. Bissyandé, D. Lo, and L. Jiang, "An empirical study of adoption of software testing in open source projects," in *2013 13th International Conference on Quality Software*, pp. 103–112, IEEE, 2013.

[5] A. Jangra, G. Singh, J. Singh, and R. Verma, "Exploring testing strategies," *International Journal of Information Technology and Knowledge Management*, vol. 4, pp. 297–299, 2011.

[6] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in *2016 6th international conference on information and communication technology for the Muslim world (ICT4M)*, pp. 177–182, IEEE, 2016.

[7] M. Shepperd, N. Ajienka, and S. Counsell, "The role and value of replication in empirical software engineering results," *Information and Software Technology*, vol. 99, pp. 120–132, 2018.

[8] M. Cruz, B. Bernárdez, A. Durán, J. A. Galindo, and A. Ruiz-Cortés, "Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018," *IEEE Access*, vol. 8, pp. 26773–26791, 2019.

[9] Y. Ma, T. Dey, C. Bogart, S. Amreen, M. Valiev, A. Tutko, D. Kennard, R. Zaretzki, and A. Mockus, "World of code: enabling a research workflow for mining and analyzing the universe of open source VCS data," *Empirical Software Engineering*, vol. 26, pp. 1–42, 2021.

[10] A. Mockus, A. Nolte, and J. Herbsleb, "MSR Mining Challenge: World of Code," 2023.

[11] S. Chacon, "Git SCM." https://git-scm.com. Accessed: 2023-01-26.

[12] D. Qiu, B. Li, E. T. Barr, and Z. Su, "Understanding the syntactic rule usage in java," *Journal of Systems and Software*, vol. 123, pp. 160–172, 2017.

[13] A. Mockus, "README.md." https://bitbucket.org/swsc/lookup/src/master/README.md. Accessed: 2023-03-12.

[14] JUnit, "Annotation Type Test." https://junit.org/junit4/javadoc/4.12/org/junit/Test.html. Accessed: 2023-01-26.

[15] C. Beust, "TestNG." https://testng.org/doc/documentation-main.html. Accessed: 2023-01-26.

[16] N. C. Borle, M. Feghhi, E. Stroulia, R. Greiner, and A. Hindle, "Analyzing the effects of test driven development in Github," in *Proceedings of the 40th International Conference on Software Engineering*, pp. 1062–1062, 2018.

[17] D. A. Wheeler, "SLOCCount." http://dwheeler.com/sloccount/. Accessed: 2023-03-11.

[18] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[19] J. H. Zar, "Spearman rank correlation," *Encyclopedia of biostatistics*, vol. 7, 2005.

[20] "Data for Evolution of the Practice of Software Testing in Java Projects." https://doi.org/10.6084/m9.figshare.22258444.v1. Accessed: 2023-03-11.

[21] A. A. Bangash, H. Sahar, A. Hindle, and K. Ali, "On the time-based conclusion stability of cross-project defect prediction models," *Empirical Software Engineering*, vol. 25, no. 6, pp. 5047–5083, 2020.