Evaluating Test Quality in GitHub Repositories: A Comparative Analysis of CI/CD Practices Using GitHub Actions

Edson Campolina Silva[1], **Rodolfo Rodrigues**[1] **Johnatan Oliveira**[2], **Danilo Boechat**[1], **Cleiton Tavares**[1]

[1]Institute of Exact Sciences and Informatics – Pontifical Catholic University of Minas Gerais (PUC-MG)
Brazil - Belo Horizonte- MG

[2]Federal University of Lavras (UFLA/ICTIN)
Brazil - São Sebastião do Paraíso-MG

{rodolforrodrigues14,ejcsengsoft}@gmail.com, johnatan.oliveira@ufla.br

{cleitontavares,daniloboechat}@pucminas.br

Abstract. *Tests are present in different stages of software development and perform an important role throughout the lifecycle of an application. Although, test smells are undesirable, as they characterize poorly designed tests that negatively impact their quality. Therefore, the goal of this study is to investigate the quality of tests in GitHub repositories that implement or do not implement Continuous Integration and Continuous Delivery (CI/CD). To achieve this goal, we conduct a comparative analysis between repositories that use GitHub Actions as CI/CD environments and repositories that do not use CI/CD. To evaluate the quality of the test suite, test smells were detected, and bug issues and the time taken to fix these issues were analyzed. In total, 651 repositories that use GitHub Actions and 289 that do not use CI/CD were analyzed. As a result, 1,648,254 test smells were identified in repositories that use GitHub Actions and 709,680 in repositories that do not. It was found that 86.18% of bug-type issues were closed in repositories that use GitHub Actions, compared to 89.20% in those that do not. Furthermore, the median time for resolving bug-type issues was 156 hours in repositories using GitHub Actions, compared to 178 hours in those without CI/CD. Finally, after statistical tests, it was not possible to state that the use of GitHub Actions in the repositories improves the quality of the tests implemented.*

Keywords: *GitHub Actions, Test smells, Continuous Integration.*

1. Introduction

Test smells are poorly designed tests that negatively affect the quality of test suites and code [3]. They provide an essential role in the software development lifecycle, ensuring that expected outcomes are achieved with structured and cohesive code [1]. Continuous Integration and Continuous Delivery (CI/CD) are practices that enable the continuous integration and delivery of code changes, which may include the test suite validation, playing a significant role in modern software development [5]. An example of a service

that facilitates the automation, customization, and execution of workflows, while also including CI/CD activities, is GitHub Actions[1].

Some studies focus on the use of automated testing tools together with CI/CD [2, 6, 7], while others discuss the relationship between the quality of software tests and the quality of the final product [8, 11, 12]. However, there are few studies that link the quality of software tests to projects using CI/CD tools. Therefore, it is necessary to measure the quality of tests to validate their scope and contribution to CI/CD, as redundant tests may fail to find bugs, creating the illusion that everything is functioning correctly. For this reason, this paper investigates whether repositories using GitHub Actions as a CI/CD tool have better quality in test cases compared to repositories that do not use CI/CD tools.

CI/CD is a set of practices that automate the building, testing, and deployment of applications, helping developers deliver code changes more quickly and reliably [5]. Software tests employed with CI/CD ensure the quality of the final product and discover bugs in the code [4]. Therefore, it is crucial to verify whether repositories using CI/CD pay attention to these principles, specifically whether their automated tests have extensive coverage to prevent the deployment of bugs in the production environment. This research aims to analyze the quality of tests created in GitHub repositories that use GitHub Actions compared to repositories that do not use CI/CD tools

As a result, 1,648,254 test smells were identified in repositories that use GitHub Actions, compared to 709,680 in repositories without CI/CD. It was found that 86.18% of bug-type issues were closed in repositories using GitHub Actions, while 89.20% were closed in those not using CI/CD. Furthermore, the median time for resolving bug-type issues was 156 hours in repositories with GitHub Actions, compared to 178 hours in those without CI/CD, with a *p-value* of 0.611. When comparing the number of test smells between repositories with GitHub Actions and those without CI/CD, the *p-value* was 0.3095. Additionally, evaluating the impact of test smells on the resolution time of bug-type issues showed a *p-value* of 0.8324 for repositories using GitHub Actions. In contrast, repositories without CI/CD had a *p-value* of 0.00005, indicating a moderate and statistically significant correlation in the latter scenario.

The remainder of this paper is structured as follows. Section 2 discusses the study setup. Section 3 presents the results. Section 4, an analysis of the results is conducted. Section 5 shows related works in the area. Section 6 shows the threats to validity. Finally, Section 7 presents the main conclusions and future directions.

2. Study Setup

This quantitative study aims to measure the quality of tests in GitHub repositories that use GitHub Actions as a CI/CD tool compared to those that do not use CI/CD. We use test smells to evaluate the quality of tests. Additionally, the analysis involves data on bug-type issues and the time required to fix these issues. Figure 1 presents the steps performed in the methodology, which are detailed below.

1-Collect Repository. In this step, we selected repositories that use GitHub Actions and have automated tests, as well as repositories that have tests but do not use CI/CD. Table 1 presents the four criteria (C1 to C4) used to refine the selection of repositories.

[1] `https://docs.github.com/en/actions`

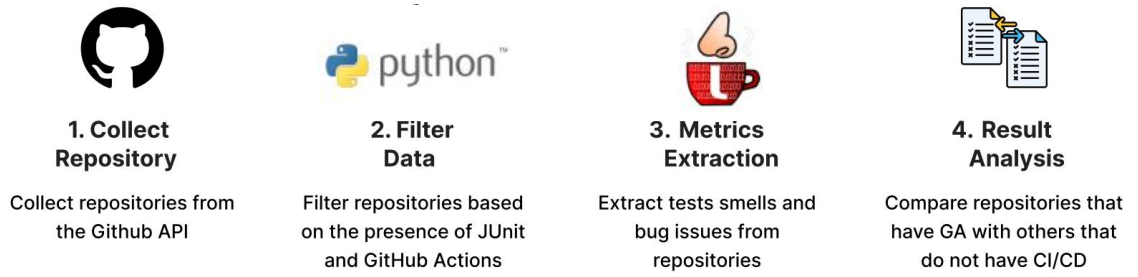

Figure 1. Study Steps

The first criterion (C1) selects repositories that use the Java programming language, chosen for its established presence in the software industry, exceeding 15 million repositories. The next criterion (C2) selects repositories with a minimum of 500 stars, aiming to include projects with a certain level of relevance in the community, resulting in approximately 6.5 thousand repositories. The third criterion (C3) includes only repositories that have been updated in the last 3 months to ensure active and maintained projects, resulting in 2,290 repositories. Finally, the criterion (C4) restricts the selection to repositories with at least 100 Pull Requests, indicating significant developer engagement and collaboration. After applying these filters, the dataset consisted of 1,387 repositories.

Table 1. Total number of repositories by selection criteria.

Identifier	Selection criteria	Quantity
C1	Java language	15.520.958
C2	C1 + Minimum 500 stars	6.506
C3	C2 + Last update date 3 months ago	2.290
C4	C3 + Minimum 100 Pull request	1.387

2-Filter Data. This step was executed in two stages. In the first stage, repositories were filtered to exclude those without JUnit tests, resulting in 197 repositories being removed. Those using JUnit were analyzed in the next stage, which consisted of identifying repositories using Github Actions (GA). The criteria used to select these repositories were: (i) having the .yml configuration file in the .GitHub/workflows directory, which is used to define automation tasks and workflows for the GitHub repository; and (ii) detecting the presence of automated tests through a task in the .yml configuration file named test. This stage resulted in 651 repositories that had GitHub Actions. Repositories that did not meet these requirements formed the group without any CI/CD, representing 289 repositories.

Furthermore, we identified 78 repositories that used a CI/CD different from GitHub Actions, which were discarded because they are not the focus of this study. Additionally, 172 repositories could not be analyzed due to the presence of non-UTF-8 characters in the code, leading to their exclusion. For the next steps, only repositories with GA (651) and without CI/CD (289) were considered, representing a total of 940 repositories.

3-Metrics Extraction. After collecting the repositories, closed bug-type issues, the time taken to fix each issue, and the test smells from each repository were obtained.

Initially, each repository was analyzed using JNose [9], a tool that automatically detects 21 test smells in code testing and collects coverage metrics with their default configuration. A Python crawler was developed to clone the repository, subject it to a test smells analysis, extract the data, remove the repository, and proceed to the next repository in these steps. Once all repositories were analyzed, a JavaScript script was developed to aggregate all the metrics and test smells extracted by JNose into a Comma Separated Value (CSV) file. This file indicated the number of test smells per test class and the number of lines of code (LOC) in each class. The selected test smells were configured in the JNose settings, allowing for the selection of which test smells would be identified during the repository analysis.

Following that, a Python script was developed to collect closed and open issues from the repositories. This script queried the GitHub REST API[2] to extract issues labeled as *bug* and stored the results in a CSV file, containing issue identifiers and the total number of closed and open issues for each repository. For repositories using a different label naming convention to denote bugs in the code, regular expressions were used for handling. Finally, for each collected issue, its creation and closing times were retrieved. These data allowed for calculating the issue resolution time.

4-Result Analysis. The next step involves comparing the collected metrics between the repositories in the group with GitHub Actions and the repositories without CI/CD. In this context, an analysis is conducted on bug-type issues, the time taken to fix these issues, and the test smells present in each repository. To mitigate the impact of the discrepancy in the number of repositories between the two groups, central tendency analyses were performed, using the median as the central measure.

3. Results

To analyze the results, we initially summarized the data from the 940 repositories. Table 2 presents the analyzed groups (column 1), the total number of test classes (column 2), the total value of test smells identified (column 3), and the total number of test smells per test class (column 4). Note that repositories with GitHub Actions have a lower quantity of test smells per class compared to the group without any CI/CD. This result may indicate higher test quality in repositories that implement GitHub Actions.

Table 2. Relationship of test smells in repositories

Groups	Number of classes	Quantity of tests smells	Test smells/ Class
With GitHub Actions	59,039	1,648,254	27.91
Without CI/CD	22,794	709,680	31.13

Table 3 presents the bug-type issues collected from the repositories. The first column presents the analyzed groups. The second and third columns present the number of open and closed issues, respectively. The fourth column presents the total issues. Finally, the last column presents the percentage of closed issues. We can observe that the group with GitHub Actions showed a lower percentage of closed issues. This suggests that the use of a CI/CD tool did not directly impact resolving more bugs.

[2]https://docs.github.com/en/rest?apiVersion=2022-11-28

Table 3. Bug issues in repositories

Groups	Open issues	Issues Closed	Total Issues	Issues Closed (%)
With GitHub Actions	10,436	65,107	75,543	86.18
Without CI/CD	2,910	24,050	26,960	89.20

Furthermore, we analyze the correction times for closed issues in the repositories. Figure 2 shows that repositories with GitHub Actions (GA) had a median of 156.59 hours to complete bug-type issues. In contrast, the group without CI/CD had a median of 178.96 hours for resolution. The Mann-Whitney U statistical test validates the existence of a statistically significant difference between the two samples. The result obtained was p-value = 0.6111, which is above the pre-defined significance level (0.05). This suggests that there is no statistically significant difference in fix times for bug-type issues between the groups with GA and without CI/CD.

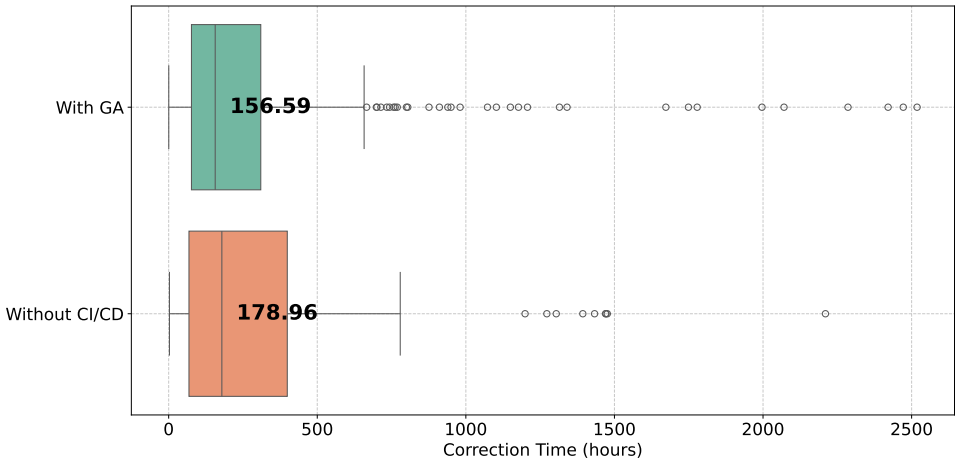

Figure 2. Time to fix bug issues

3.1. Comparison of Test Smells between Repositories with GitHub Actions and without CI/CD Tools

Evaluating the quality of tests in repositories with GA and without CI/CD requires an analysis based on the number of test smells that each group of repositories has. Figure 3 shows that the median value was 0.1495 for repositories that adopt the GitHub Actions tool and 0.1515 for repositories that do not use CI/CD tools. It is also observed that repositories without CI/CD have slightly higher maximum values. Regarding minimum values, there was no significant difference between the two groups.

To perform this comparison equitably, a calculation was made by dividing the number of test smells by the number of lines of code (LOC) of each repository. This approach allowed obtaining a measure of test smells per LOC used in the analysis, mitigating the bias that the increase in the number of lines of code could artificially increase the detection of test smells. Furthermore, outliers were kept to ensure the integrity of the results.

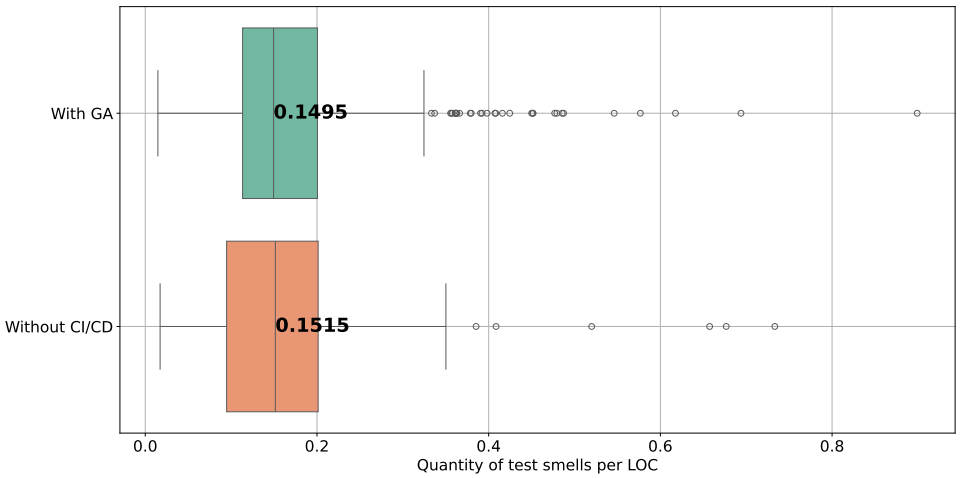

Figure 3. Test smells in repositories

The Mann-Whitney U test was also applied to evaluate the statistical significance of the results between the groups. A significance level of 0.05 was established with the null hypothesis that there is no significant difference in the number of test smells between the groups with GA and without CI/CD. The obtained value was $p - value = 0.3095$, which exceeds the defined significance level. This suggests that there is not enough evidence to reject the null hypothesis, so we cannot assert that there is a statistically significant difference in the number of test smells per LOC between the two analyzed groups.

Finally, the presence or absence of GitHub Actions did not show a significant impact on the occurrence of test smells in the analyzed repositories, indicating that this CI/CD tool does not measurably influence the quality of software tests in this aspect.

3.2. Correspondence between Bug Type Issues and Number of Test Smells

To investigate the relationship between the number of bug-type issues and the number of test smells in the repositories, a scatter plot was created for the two metrics used. Figure 4 shows this relationship, where each point represents a repository, with the position on the X-axis indicating the number of test smells and the position on the Y-axis indicating the number of bug-type issues. Additionally, the graph includes a trend line for each group. To statistically analyze the results, Pearson's correlation was applied, which measures the strength and direction of the linear relationship between two variables.

In this analysis, it is observed that the relationship between the number of test smells per LOC and the number of bug type issues is very weak in both groups. In repositories with GA, the Pearson correlation coefficient was -0.065, and in repositories without CI/CD, it was 0.073, indicating that there is no significant linear correlation, as coefficients close to 0 indicate a very weak correlation between the variables.

Finally, for a correlation to be considered statistically significant, the p value must be less than 0.05. In the results obtained, the GA group had a p value of 0.243 and the group without CI/CD had a p value of 0.465. This indicates that the observed correlation in both groups is not statistically significant, i.e., the number of test smells per LOC does not have an evident influence on the number of bug type issues in the repositories. Therefore, other factors may be contributing to the number of bugs.

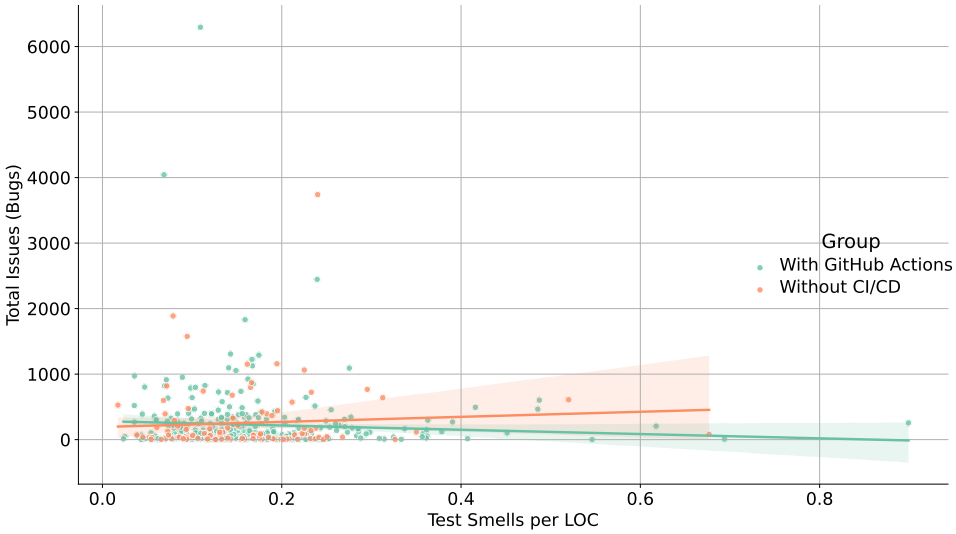

Figure 4. Number of bug type issues vs. test smells in repositories

3.3. Impact of Test Smells on Bug Resolution Time

To evaluate the influence of test smells on the correction time of bug-type issues for repositories that use GitHub Actions and those that do not use CI/CD, a scatter plot was created. Figure 5 shows this chart, with the X-axis representing the number of test smells and the Y-axis representing the correction time of closed bug-type issues. The chart includes two trend lines reflecting the general direction of the relationship between the two analyzed metrics. Additionally, to improve the comparison between the groups, outliers related to closed issues were treated. Only issues with completion times between 0.08 hours (about 5 minutes) and 4320 hours (approximately 6 months) were considered.

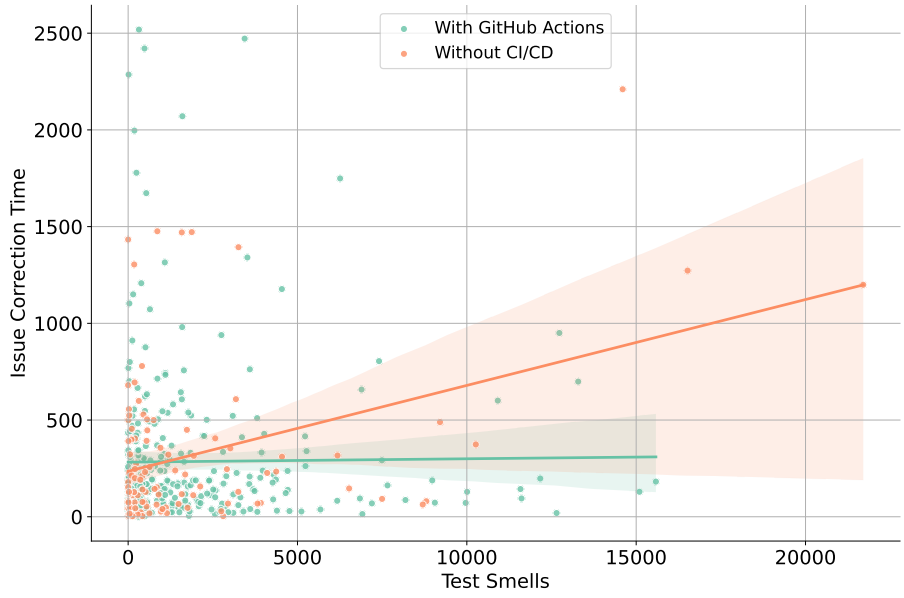

Figure 5. Bug correction time vs. test smells in repositories

The results of Pearson's correlation revealed that, in repositories with GitHub Actions, the correlation coefficient was 0.0117 ($p - value = 0.8324$), indicating that there

is no significant linear correlation between the number of test smells and the correction time of issues. On the other hand, in repositories without CI/CD, the correlation coefficient was 0.3832 ($p - value = 0.00005$), showing a moderate and statistically significant correlation. This suggests that, in repositories without CI/CD, an increase in the number of test smells is associated with an increase in the time required to correct the issues.

Therefore, while the presence of GA did not show a significant impact on the correction time of issues concerning test smells, the absence of CI/CD seems to imply additional challenges in managing and correcting code problems. This relationship may suggest that the increase in test smells increases the correction time in the group without CI/CD, possibly influenced by variables not considered, such as the skill and experience of development teams.

4. Discussion

This section discusses the results obtained after applying the described study setup. The analysis compared repositories that use GitHub Actions and those without CI/CD tools, focusing on the number of test smells, their relationship with bug-type issues, and the correction time of these issues. It was identified that there was no statistically significant difference in the number of test smells per LOC between repositories with GitHub Actions and those without CI/CD. The correlation between the number of test smells per LOC and bug-type issues, based on the Pearson test, did not show an evident influence between the variables.

The influence of test smells on the correction time of issues was also analyzed. In repositories that use GitHub Actions, it was not possible to establish a statistical relationship between the increase in test smells and the correction time of bugs. In contrast, repositories without CI/CD demonstrated a moderate and statistically significant correlation, where the correction time of issues tends to increase as the occurrence of test smells grows.

The results indicate that the use of CI/CD tools, such as GitHub Actions, does not significantly impact the bug correction time. However, in repositories without CI/CD, the increase in test smells is correlated with a longer correction time of issues, possibly influenced by variables not considered in this study. These findings suggest that continuous integration and delivery did not have a direct impact on the number of test smells, the incidence of bugs, or the correction time of issues, as evidenced in this research.

5. Related Work

This section discusses the works related to the topic of this paper. K. Gallaba et al. (2022) conducted an empirical study on the use of CircleCI from 2012 to 2020, analyzing 22.2 million builds from 7,795 open-source projects. It was found that build and test actions consume a large proportion of build time, suggesting that efficiency in CI/CD can reduce costs and provide quick feedback. This study is relevant to this work by analyzing CI/CD tools and providing applicable conclusions. Sk Golam et al. (2023) analyzed developers' perceptions of using GitHub Actions, using an empirical approach to understand CI/CD automation. The detailed analysis suggests that optimizing CI/CD processes can reduce costs and ensure agile returns on code changes, influencing test quality. This

study highlights the importance of optimizing CI/CD processes, aligning with the ongoing research goals.

Spadini et al. (2020) investigated the severity of four test smells and their impacts on test maintainability. They analyzed about 1,500 open-source projects and conducted a study with professional developers. The results showed that test smells negatively impact the maintainability of tests, directly relating to the present work. Aljedaani et al. (2021) conducted a systematic study comparing 70 test smell detection tools, based on a review of 125 papers. The tools were classified as manual, automated, and hybrid, concluding that they are useful for improving test quality. This paper provides a foundation of test smell detection tools applicable to the objectives of this work.

6. Threats to Validity

In this section, we discuss the potential threats to the validity of our study, following the guidelines proposed by Wohlin et al. [10]. We address threats to internal, external, conclusion, and construct validity.

Internal Validity – One of the main threats pertains to the terminological variation used to label issues as bugs. Some repositories used variations such as BUG, bugs, and Bugs. To mitigate this threat, automated preprocessing was conducted to identify these variations and ensure the inclusion of all pertinent issues. Additionally, the focus was exclusively on issues, excluding pull requests, to maintain the analysis centered on the identification and resolution of specific bugs.

External Validity – The use of the programming language. The selection of Java repositories was due to its relevance in software development, utilizing the JNose tool to extract test smell metrics. The tool only analyzes repositories that use JUnit, resulting in the division of repositories into two groups: those that used JUnit and those that did not. It was ensured that the selection of repositories was representative, using clear inclusion criteria.

Construct Validity – It was identified that some repositories on GitHub used CI/CD tools other than GitHub Actions (GA). A third group was created for these repositories, in addition to the two initial groups, allowing the isolation of repositories that used other CI/CD tools and keeping the analysis focused on the impact of GA.

Conclusion Validity – There was a difficulty in segregating test smells into types for JNose analysis. This resulted in the inclusion of test smells such as “Ignored Test” with the same weight as more impactful test smells, such as Conditional Test Logic. To mitigate this limitation, a differentiated weighting was implemented, assigning variable weights according to the impact on test quality.

7. Conclusion

This study investigated the quality of tests in GitHub repositories using GitHub Actions as a CI/CD tool compared to those not using CI/CD. By analyzing 1,387 repositories, we assessed the presence of test smells, the number of bug issues, and the time taken to resolve these issues. The findings indicated no statistically significant difference in the number of test smells per LOC between repositories with GitHub Actions and those without CI/CD. Additionally, there was no significant correlation between the number of

test smells and bug issues or the correction time of these issues. This suggests that the use of GitHub Actions does not directly improve test quality or efficiency in resolving bugs.

Future work should consider extending the analysis to repositories using other programming languages and CI/CD tools, such as CircleCI and Travis CI. Moreover, exploring additional data sources beyond GitHub and employing various static code analysis tools could provide deeper insights into the impact of CI/CD practices on test quality.

Replication Package

We provide spreadsheets with GitHub projects, processed data (test smells, source code metrics), and analysis scripts to facilitate replication and validation. The replication package for this study is available at:
`https://doi.org/10.5281/zenodo.12774105`

References

[1] Wajdi Aljedaani, Anthony Peruma, Ahmed Aljohani, Mazen Alotaibi, Mohamed Wiem Mkaouer, Ali Ouni, Christian D. Newman, Abdullatif Ghallab, and Stephanie Ludi. Test smell detection tools: A systematic mapping study. EASE '21, page 170–180, New York, NY, USA, 2021. ACM.

[2] Keheliya Gallaba, Maxime Lamothe, and Shane McIntosh. Lessons from eight years of operational data from a continuous integration service: An exploratory case study of circleci. In *2022 IEEE/ACM ICSE*, pages 1330–1342, 2022.

[3] Vahid Garousi, Baris Kucuk, and Michael Felderer. What we know about smells in software test code. *IEEE Software*, 36(3):61–73, 2019.

[4] Shalini Joshi and Indra Kumari. Analyses of software testing approaches. In *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, pages 1276–1281, 2022.

[5] Pei Liu, Xiaoyu Sun, Yanjie Zhao, Yonghui Liu, John Grundy, and Li Li. A first look at ci/cd adoptions in open-source android apps. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA, 2023. ACM.

[6] Akshit Raj Patel and Sulabh Tyagi. The state of test automation in devops: A systematic literature review. In *Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing*, IC3-2022, page 689–695, New York, NY, USA, 2022. ACM.

[7] Sk Golam Saroar and Maleknaz Nayebi. Developers' perception of github actions: A survey analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, EASE '23, page 121–130, New York, NY, USA, 2023. ACM.

[8] Davide Spadini, Martin Schvarcbacher, Ana-Maria Oprescu, Magiel Bruntink, and Alberto Bacchelli. Investigating severity thresholds for test smells. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 311–321, New York, NY, USA, 2020. ACM.

[9] Tássio Virgínio, Luana Martins, Larissa Rocha, Railana Santana, Adriana Cruz, Heitor Costa, and Ivan Machado. Jnose: Java test smell detector. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pages 564–569, 2020.

[10] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Computer Science. Springer Berlin Heidelberg, 2012.

[11] Justyna Zander-Nowicka, Pieter J. Mosterman, and Ina Schieferdecker. Quality of test specification by application of patterns. In *Proceedings of the 15th Conference on Pattern Languages of Programs*, PLoP '08, New York, NY, USA, 2008. ACM.

[12] Yi Zhao, Yun Hu, and Jiayu Gong. Research on international standardization of software quality and software testing. In *2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall)*, pages 56–62, 2021.