Reprodução do estudo “Evolution of the Practice of Software Testing in Java Projects”

Ana Julia Teixeira Candido
anajuliateixeiracandio@gmail.com
PUC Minas
Belo Horizonte, Minas Gerais, Brasil

Marcella Ferreira Chaves Costa
marcellafccosta@gmail.com
PUC Minas
Belo Horizonte, Minas Gerais, Brasil

Sophia Mendes Rabelo
sophiamendesrabelo@gmail.com
PUC Minas
Belo Horizonte, Minas Gerais, Brasil

RESUMO

A reprodutibilidade de artigos consiste na repetição da pesquisa realizada em busca de resultados semelhantes ao obtidos a fim de validar o que foi estudado. Nesse sentido, o objetivo deste trabalho ao reproduzir o artigo “*Evolution of the Practice of Software Testing in Java Projects*”, é confirmar que as práticas de teste em projetos de código aberto Java têm se mantido consistentes ao longo dos anos. O estudo original investigou questões relacionadas ao teste de software, e os autores do artigo reproduzido replicaram o estudo de 2013 extraindo os dados do GitHub através da infraestrutura World of Code (WoC). A análise dos dados para determinar se as conclusões do estudo permaneceram consistentes ao longo dos anos, foi feita utilizando medidas estatísticas (teste de Mann-Whitney-Wilcoxon e o coeficiente de correlação de Spearman).
Em síntese, a reprodução do experimento confirmou a eficácia da metodologia proposta pelo artigo original, evidenciando práticas de teste consistentes e alinhadas em projetos Java, o que reforça a confiabilidade e aplicabilidade da abordagem utilizada.

1 INTRODUÇÃO

É essencial avaliar os contextos dos estudos envolvidos neste trabalho. Kochhar et al. analisaram 20.817 projetos de software em 2013 para determinar a popularidade dos testes de software em projetos de código aberto. Com o avanço das práticas e ferramentas de teste, é necessário verificar se as conclusões ainda são válidas. Em 2023, uma replicação desse estudo analisou projetos de código aberto em Java no GitHub de 2012 a 2021 para validar as descobertas originais. Este trabalho universitário reproduz o estudo "Evolution of the Practice of Software Testing in Java Projects"para investigar se as conclusões de Kochhar et al. permanecem válidas, seguindo os mesmos passos da replicação de 2023.

A relevância dessa reprodução na Engenharia de Software vai além da validação das conclusões originais, oferecendo uma visão atualizada das práticas de teste. Ela aumenta a confiança na pesquisa original, fortalece a base para novas pesquisas e práticas, e permite identificar mudanças e tendências nas práticas de teste de software ao longo dos anos. Isso beneficia tanto pesquisadores quanto profissionais da área, proporcionando insights atualizados e relevantes para melhorias contínuas no desenvolvimento de software.

, ,

2 MATERIAIS E MÉTODOS

O artigo original analisa a evolução das práticas de teste de software em projetos Java. O método principal envolve a coleta de repositórios Java do GitHub, seguida de uma análise detalhada das seguintes métricas:

(1) Número de Linhas de Código (LOC)
(2) Número de Autores
(3) Presença de Arquivos de Teste

2.1 Equações e Algoritmos Utilizados

(1) Contagem de Linha de Código (LOC)
- Algoritmo: Percorrer todos os arquivos Java no repositório e contar o número de linhas.
- Equação:

$$\text{LOC} = \sum_{f \in \text{ArquivosJava}} \text{linhas}(f) \tag{1}$$

(2) Contagem de Autores
- Algoritmo: Extrair o histórico de commits do repositório e contar o número de autores únicos.
- Equação:

$$\text{Autores} = \text{count}\left(\text{distinct}\left(\text{autor}(c) \mid c \in \text{Commits}\right)\right) \tag{2}$$

(3) Detecção de Arquivos de Teste
- Algoritmo: Verificar a presença de arquivos cujo nome ou caminho contenha "test"e que tenham a extensão ".java".
- Equação:

$$\text{Possui Testes} = \exists f \in \text{ArquivosJava},\ \text{'test'} \in \text{nome}(f) \tag{3}$$

2.2 Materiais e Métodos Utilizados pelos Alunos

Para replicar a análise descrita no artigo, os seguintes materiais e métodos foram utilizados:

(1) Ferramentas e Ambiente de Desenvolvimento:
- Computador
- IDE: Visual Studio Code (VSCode) para desenvolvimento e execução de scripts Java
- Java Development Kit (JDK)
- Git: Ferramenta de controle de versão Git instalada no sistema

(2) Seleção de Repositórios:
- Repositórios Java foram selecionados do GitHub utilizando a busca avançada para encontrar projetos relevantes e populares.

(3) Clonagem de Repositórios:
- Os repositórios selecionados foram clonados utilizando o comando Git:

```
git clone <URL_do_Repositorio>
```

(4) Análise de Métricas de Código:
- Contagem de LOC: Um script foi desenvolvido para percorrer os arquivos Java e contar as linhas de código.
- Contagem de Autores: O comando git log foi utilizado para extrair autores dos commits:

```
git log --pretty=format:'%ae'
```

- Detecção de Arquivos de Teste: Os arquivos foram verificados para a presença de "test"no nome ou caminho.

Os métodos descritos acima permitiram a replicação do experimento original, analisando a evolução das práticas de teste em projetos Java. A utilização de ferramentas como Git e scripts personalizados facilitou a coleta e análise das métricas necessárias para o estudo.

O repositório contendo o código fonte e scripts utilizados na reprodução do experimento pode ser encontrado em *nosso repositório no GitHub*.

3 RESULTADOS

Nessa sessão vamos discutir e analisar os resultados que o artigo "Evolução da Prática de Teste de Software em Projetos Java" chegou para responder as questões: A- Prevalência de Casos de Teste, B- Correlação entre Número de Desenvolvedores e Casos de Teste, C- Correlação de contagem de bugs com casos de teste. A partir disso vamos compará-los com o resultado que chegamos a partir da reprodução do estudo original.

3.1 Prevalência de Casos de Teste Artigo Original

O artigo original validou a afirmativa de KOCHHARSTUDY, no qual "Projetos com casos de teste são maiores em tamanho do que projetos sem teste", a partir da inspeção da relação entre números de linhas de código e casos de teste. A figura 1 mostra essa relação de LOC *-Line of code-* e casos de teste. Ao final, perceberam que o valor de LOC para projetos sem casos de teste é inferior a média de LOC de projetos que possuem caso de teste. O que os levaram a reconfirmação da afirmativa de KOCHHARSTUDY.

3.2 Prevalência de Casos de Teste Artigo de Reprodução

Para reproduzirmos essa reconfirmação do artigo orignal, utilizamos um algoritmo para calcular quantas linhas de código o projeto tem e se tem ou não presença de casos de teste. A figura 2 apresenta a relação entre LOC e casos de teste. A conclusão que chegamos é que mais uma vez foi reafirmado que o LOC para projetos sem casos de teste é inferior a média de LOC que apresentam caso de teste.

3.3 Correlação entre Número de Desenvolvedores e Casos de Teste Artigo Original

O artigo original reproduziu a questão que KOCHHARSTUDY investigou "Número de desenvolvedores afeta o número de casos de teste?". Para avaliar a correlação entre o número de autores e o número de casos de teste, foi utilizado o coeficiente de Spearman. O estudo KOCHHARSTUDY reportou uma correlação positiva fraca, e os resultados da reprodução feita pelo artigo reafirmou essa correlação fraca entre o número de autores e números de casos de teste, apresentados na figura 3. Ou seja, isso implica que, embora o número de desenvolvedores possa ter algum impacto no número de casos de teste, ele não é o único fator determinante, e essa influência não é muito forte.

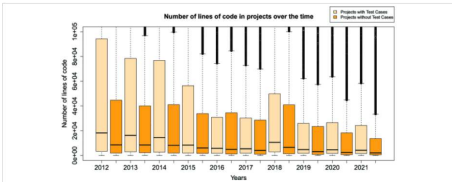

Figura 1: Comparação de linhas de código entre projetos Java com e sem casos de teste de 2012-2021

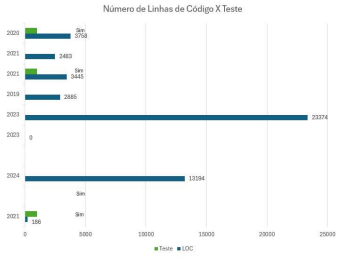

Figura 2: Reprodução da comparação de linhas de código entre projetos Java com e sem casos de teste de 2019-2024

3.4 Correlação entre Número de Desenvolvedores e Casos de Teste Artigo Reprodução

Para reproduzirmos o resultado chegado no artigo original, fizemos um algoritmo para calcular a correlação de número de autores do projeto e o número de casos de teste. O resultado que chegamos, apresentado na figura 4 mostra que a conclusão que o artigo original chegou e o a conclusão que chegamos é igual. O número de desenvolvedores no projeto não é algo tão relevante para pensarmos em implementação de caso de teste ou não. Dito que também foi observado que projetos com mais autores tendem a ter menos casos de teste do que os projetos com menos autores.

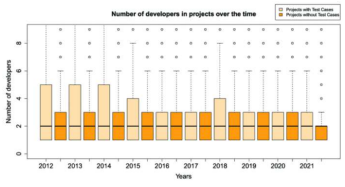

Figura 3: Comparação de Número de Autores entre Projetos Java, com e sem Casos de Teste, de 2012 a 2021

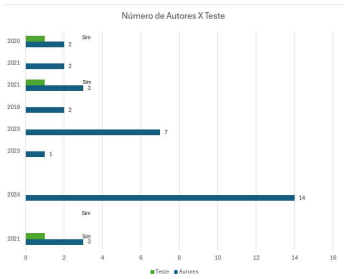

Figura 4: Comparação de Número de Autores entre Projetos Java, com e sem Casos de Teste, de 2019 a 2024

3.5 Correlação de contagem de bugs com casos de teste Artigo Original

O estudo investigou a correlação entre a presença de casos de teste e o número de bugs em projetos. Para reproduzir esta questão de pesquisa, o artigo calculou o número de bugs em um projeto a partir do rho de Spearman, que resultou em uma correlação positiva fraca, apresentado na fig 5. Diante disso, o estudo reafirmou que “Projetos com maior número de casos de teste observam um aumento no número de bugs, embora a correlação entre eles seja fraca” permanece estável entre os anos de 2012-2021.

3.6 Correlação de contagem de bugs com casos de teste Artigo Reprodução

Para reproduzirmos a correlação entre o números de bugs nos projetos e o número de casos de testes, utilizamos um algoritmo para calcular os bugs e a existência ou não de casos de teste no repositório. A conclusão que chegamos, apresentada na figura 6, reafirma o estudo de que “Projetos com maior número de casos de teste observam um aumento no número de bugs, embora a correlação entre eles seja fraca”, já que como observado no gráfico os projetos com erros no código e a presença de casos de teste é maior, apesar de ser uma correlação fraca.

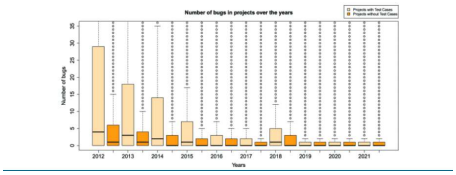

Figura 5: Comparação do Número de Bugs entre Projetos Java, com e sem Casos de Teste, de 2012-2021

4 CONCLUSÕES

A reprodução do experimento resultou em conclusões consistentes com os achados do artigo original, avaliando a eficácia da metodologia proposta. A análise das práticas de teste em projetos Java revelou resultados concretos e alinhados, reforçando a confiabilidade e aplicabilidade da abordagem utilizada.

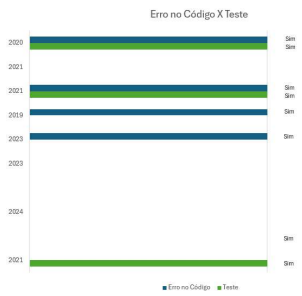

Figura 6: Comparação do Número de Bugs entre Projetos Java, com e sem Casos de Teste, de 2019-2024

Durante o processo de reprodução, diversos desafios foram enfrentados, como a familiaridade com o exercício proposto, a configuração do ambiente, clonagem de repositórios extensos, e principalmente a análise de grandes volumes de dados. Entretanto, foram superados por meio da colaboração entre a equipe.

Ademais, a clareza e detalhamento na documentação do artigo original foram essencial para a reprodução bem-sucedida. Esses fatores permitiram uma melhor e mais precisa compreensão da metodologia e do experimento.

5 AGRADECIMENTOS

Este artigo foi elaborado como parte da atividade de reprodução ou replicação da disciplina de Introdução à Pesquisa em Informática, lecionada pelo professor Lesandro Ponciano, no curso de Engenharia de Software da PUC Minas, no turno matutino, no primeiro semestre de 2024. O texto foi produzido usando a ferramenta Overleaf e o link público do texto está disponível *aqui*.

REFERÊNCIAS

[1] V.R. Basili and R.W. Selby. 1987. Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering* SE-13, 12 (1987), 1278–1296. https://doi.org/10.1109/TSE.1987.232881

[2] Neil Borle, Meysam Feghhi, Eleni Stroulia, Russell Grenier, and Abram Hindle. 2018. [Journal First] Analyzing the Effects of Test Driven Development in GitHub. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 1062–1062. https://doi.org/10.1145/3180155.3182535

[3] Margarita Cruz, Beatriz Bernárdez, Amador Durán, José A. Galindo, and Antonio Ruiz-Cortés. 2020. Replication of Studies in Empirical Software Engineering: A Systematic Mapping Study, From 2013 to 2018. *IEEE Access* 8 (2020), 26773–26791. https://doi.org/10.1109/ACCESS.2019.2952191

[4] Anisha Islam, Nipuni Tharushika Hewage, Abdul Ali Bangash, and Abram Hindle. 2023. Evolution of the Practice of Software Testing in Java Projects. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 367–371. https://doi.org/10.1109/MSR59073.2023.00057

[5] Muhammad Abid Jamil, Muhammad Arif, Normi Sham Awang Abubakar, and Akhlaq Ahmad. 2016. Software Testing Techniques: A Literature Review. In *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*. 177–182. https://doi.org/10.1109/ICT4M.2016.045

[6] Pavneet Singh Kochhar, Tegawendé F. Bissyandé, David Lo, and Lingxiao Jiang. 2013. An Empirical Study of Adoption of Software Testing in Open Source Projects. In *2013 13th International Conference on Quality Software*. 103–112. https://doi.org/10.1109/QSIC.2013.57

[7] B. Potter and G. McGraw. 2004. Software security testing. *IEEE Security Privacy* 2, 5 (2004), 81–85. https://doi.org/10.1109/MSP.2004.84

[4] [7] [1] [6] [5] [2] [3]