Testes unitários são efetivos contra Bugs?

TRABALHO INTERDISCIPLINAR DE SOFTWARE VI

Contexto do trabalho

Projeto de pesquisa e desenvolvimento elaborado para a disciplina de Trabalho Interdisciplinar de Software VI, lecionada pelos professores Humberto Neto e José Laerte Júnior, referente ao 6° período do curso de Engenharia de Software da PUC-MG.

Integrantes do grupo:

- Anderson Barbosa Coutinho
- Henrique Alberone Nunes Alves Ramos
- Mateus Santos Fonseca
- Yan Max Rodrigues Sette Pinheiro

Introdução

- Testes em software é um conceito comum nos dias de hoje. Seja feito por um time especializado ou pelos próprios desenvolvedores, verificar que o sistema funciona da maneira esperada é importante para que este atinja as expectativas dos usuários finais.

- Na pirâmide de testes, os unitários estão na base, por serem os de menor granularidade e mais rápidos.

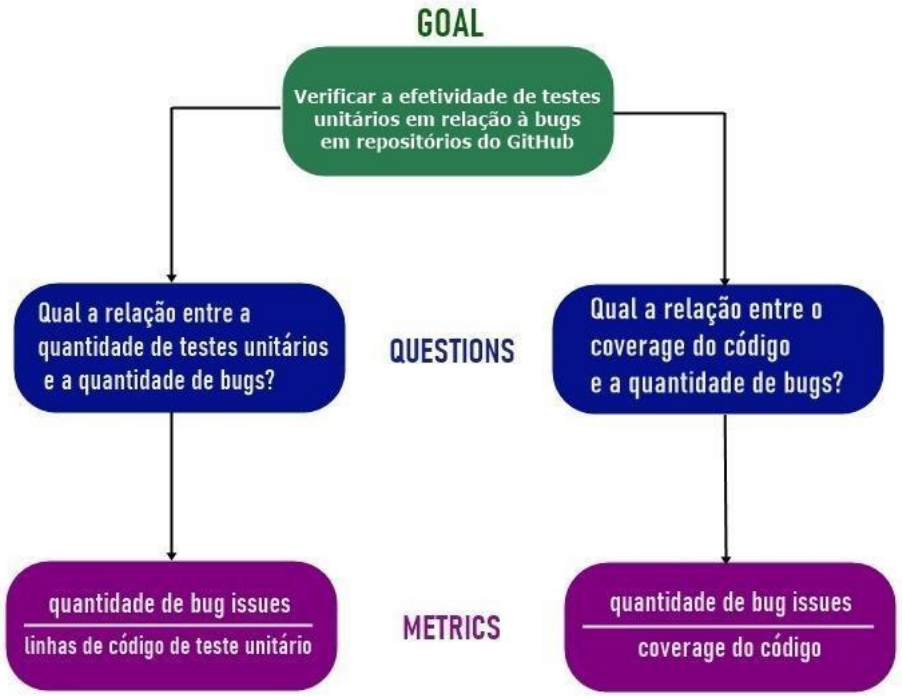
- Portanto, tendo conhecimento da importância deste tema, foi analisada a necessidade de se verificar a relação entre testes unitários e o surgimento de Bugs em repositórios populares de linguagens diversas, uma vez que, nas pesquisas em bases bibliográficas, não foram encontrados estudos profundos neste específico assunto.

GQM

- Portanto, tendo conhecido o contexto, identificamos o objetivo (goal) dessa pesquisa: **verificar a efetividade de testes unitários em relação à bugs nos repositórios do GitHub**.

- Com este Goal conhecido, definimos as questions de nossa pesquisa como sendo:
  1. Qual a relação entre a quantidade de testes unitários e a quantidade de bugs?
  2. Qual a relacão entre o coverage do código e a quantidade de bugs?

- As métricas de nosso projeto que auxiliarão a responder as questions são:
  1. Quantidade de BUG Issues/Linhas de código de teste unitário.
  2. Quantidade de BUG Issues/Coverage do repositório.



GQM

Representação gráfica do GQM de nosso projeto de pesquisa

Hipóteses

- Uma vez que buscamos atestar se existe impacto de testes unitários nos BUG Issues dos repositórios analisados, definimos nossas hipóteses como:

  - **Hipótese nula (Ho):** Testes unitários impactam nos BUG Issues dos repositórios.

  - **Hipótese alternativa (Ha):** Testes unitários não impactam nos BUG Issues dos repositórios.

Trabalhos Relacionados

- Tendo em vista que nosso trabalho é baseado no conceito de testes, mais especificamente em testes unitários, a maioria de nossas referências bibliográficas estão contidas neste contexto.

- Dentre os artigos analisados e relacionados, a maioria fala sobre Test-driven development. Alguns deles fazem relações entre os testes unitários em BUGs de produção, porém não aprofundam nesta questão e ficam mais focados em como produzir ou implementar testes unitários nas diversas tecnologias atuais de desenvolvimento de software.

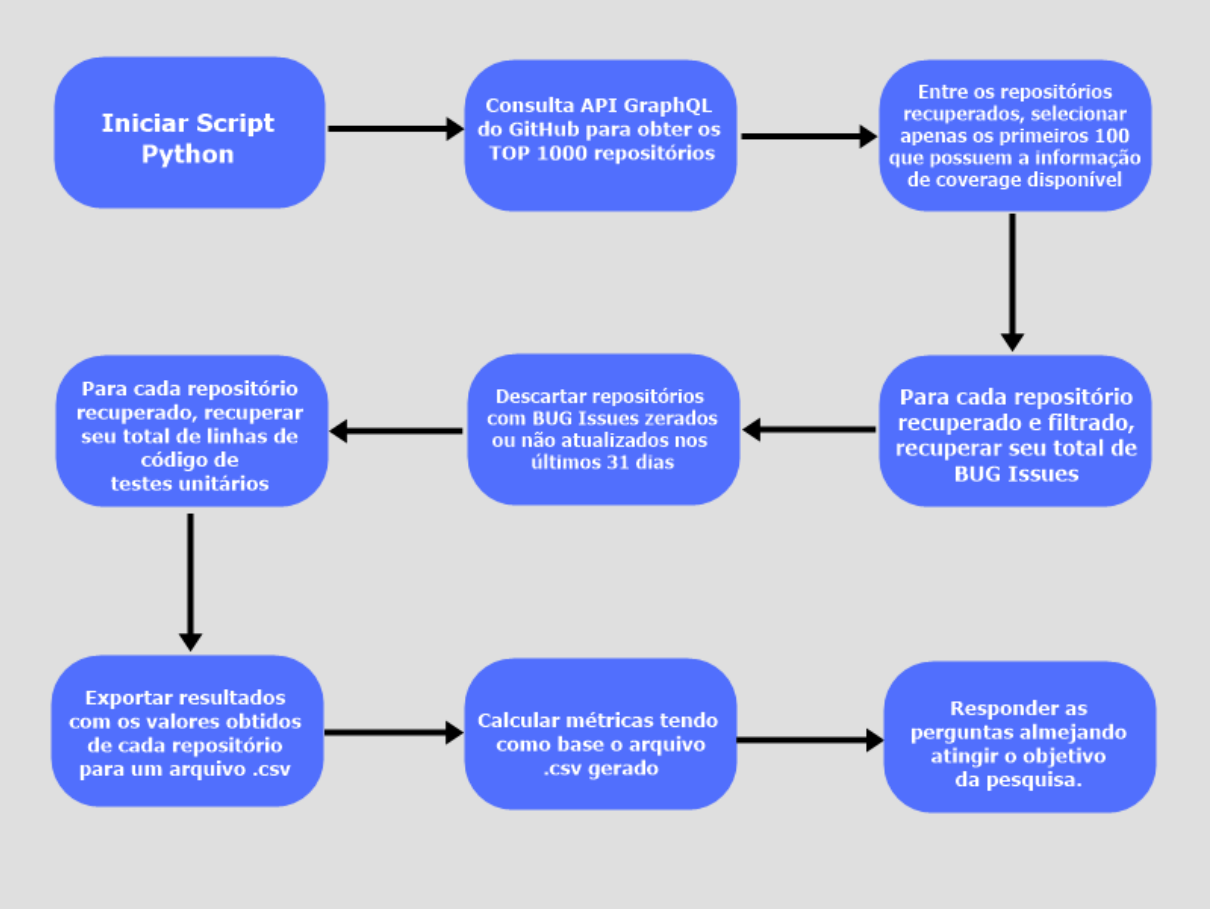
- Justamente devido a esta escassez de comparações entres questões de testes unitários e bugs, foi percebido uma necessidade de desenvolver nosso trabalho.

Metodologia

- A metodologia deste trabalho foi dividida em duas etapas chave:

1. Desenvolver e executar Script Python que obteria todos os dados necessários sobre os repositórios que serão analisados.

2. Análise dos resultados obtidos e sua documentação.



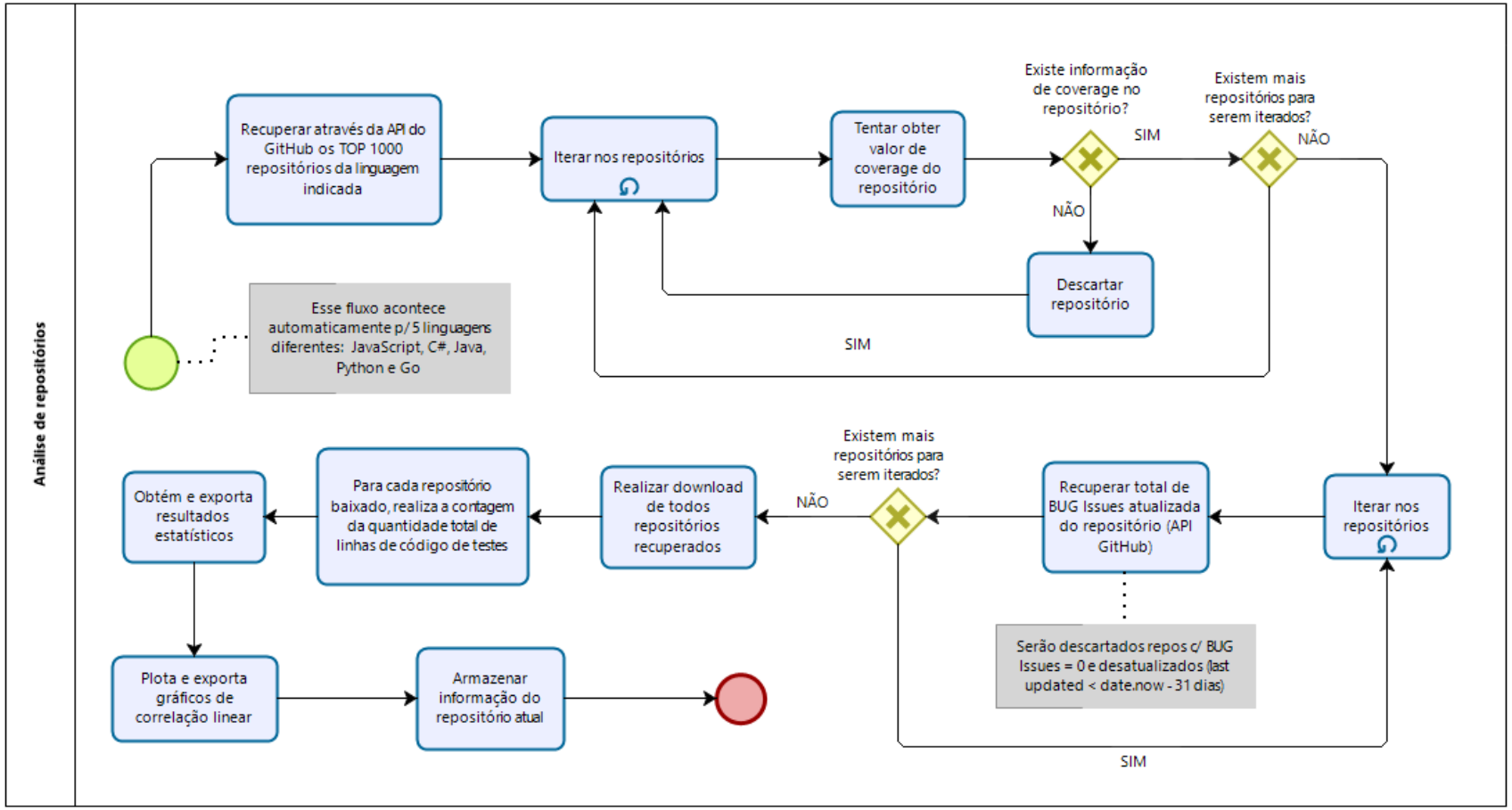
Metodologia

Overview gráfico da metodologia de nosso projeto de pesquisa

Metodologia – Script Python

- Para atingir o objetivo, desenvolvemos um script Python que, em uma só execução, para cada uma das seguintes linguagens: JavaScript, Go, C#, Java e Python:

- Recupera seus top 1000 repositórios com a API GraphQL do GitHub;
- Dentre os top 1000 repositórios, filtra apenas aqueles que possuem a informação de coverage disponível;
- Recupera os BUG Issues destes repositórios filtrados;
- Descarta repositórios com BUG Issues zerados ou não atualizados nos últimos 31 dias;
- Realiza e retorna as análises estatísticas automaticamente (e.g. R valor da relação entre LOC de testes e BUG Issues e Coverage e BUG Issues);
- Interpreta R Valor obtido;
- Plota e exporta gráfico de regressão linear da relação entre LOC de testes e BUG Issues e Coverage e BUG Issues.



Metodologia – Script Python

Fluxograma que demonstra as etapas dos scripts Python de nosso projeto de pesquisa

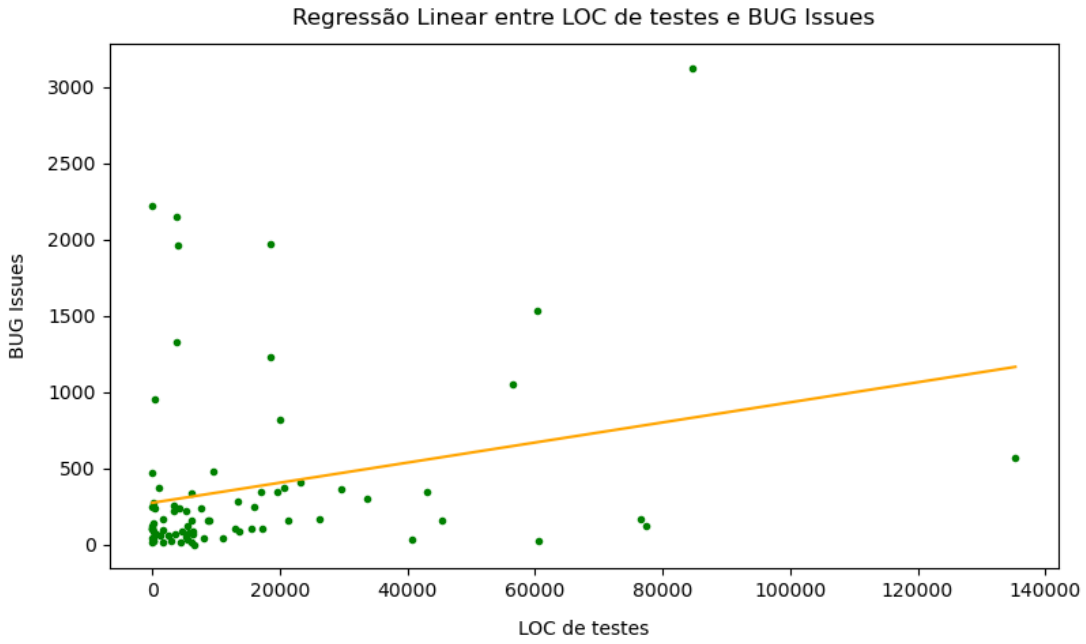
Resultados obtidos

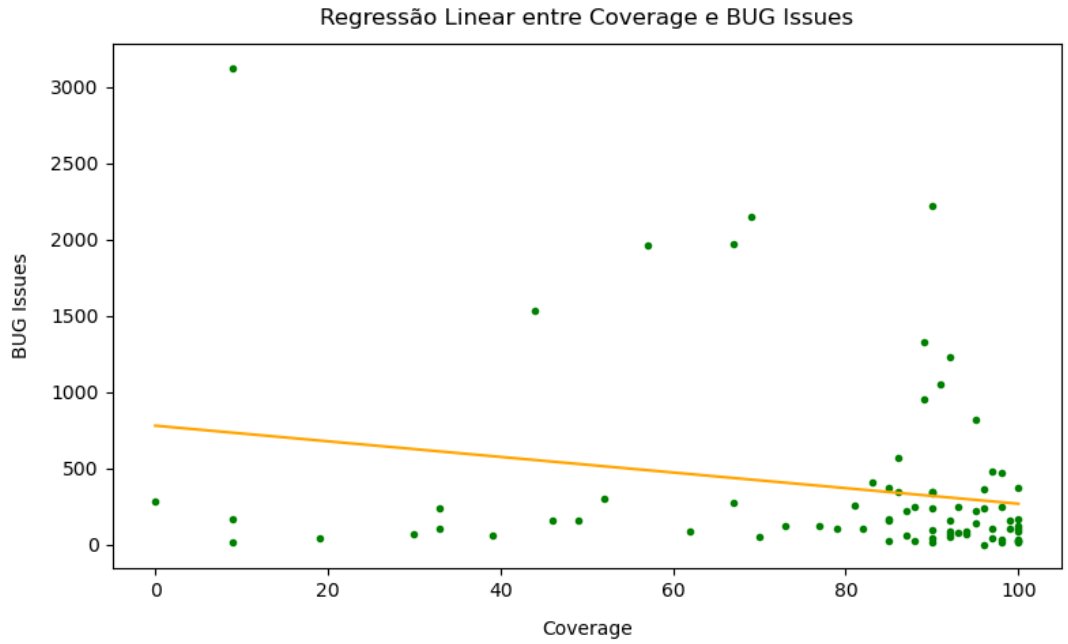
- A execução do Script Python foi realizada em 23 de Maio de 2020. É importante frisar que os dados desta pesquisa podem mudar à qualquer momento, portanto, deve-se compreender que os dados que serão apresentados foram obtidos na data de execução supracitada.

- A seguir temos o link com os repositórios analisados por linguagem (JavaScript, Python, Java, C# e Go) bem como os gráficos de regressão linear obtidos para as linguagens, seu valor R (Coeficiente de Correlação de Pearson) e sua interpretação.

.csv dos Repositórios analisados

- **JavaScript:** https://github.com/MateusSantosFonseca/TISVI-Analise-Efetividade-Testes-Unitarios/blob/master/Resultados/JavaScript/repositorios_JavaScript_analisados.csv
- **C#:** https://github.com/MateusSantosFonseca/TISVI-Analise-Efetividade-Testes-Unitarios/blob/master/Resultados/CSharp/repositorios_CSharp_analisados.csv
- **Java:** https://github.com/MateusSantosFonseca/TISVI-Analise-Efetividade-Testes-Unitarios/blob/master/Resultados/Java/repositorios_Java_analisados.csv
- **Python:** https://github.com/MateusSantosFonseca/TISVI-Analise-Efetividade-Testes-Unitarios/blob/master/Resultados/Python/repositorios_Python_analisados.csv
- **Go:** https://github.com/MateusSantosFonseca/TISVI-Analise-Efetividade-Testes-Unitarios/blob/master/Resultados/Go/repositorios_Go_analisados.csv





Valor R obtido: 0.2636300629844727

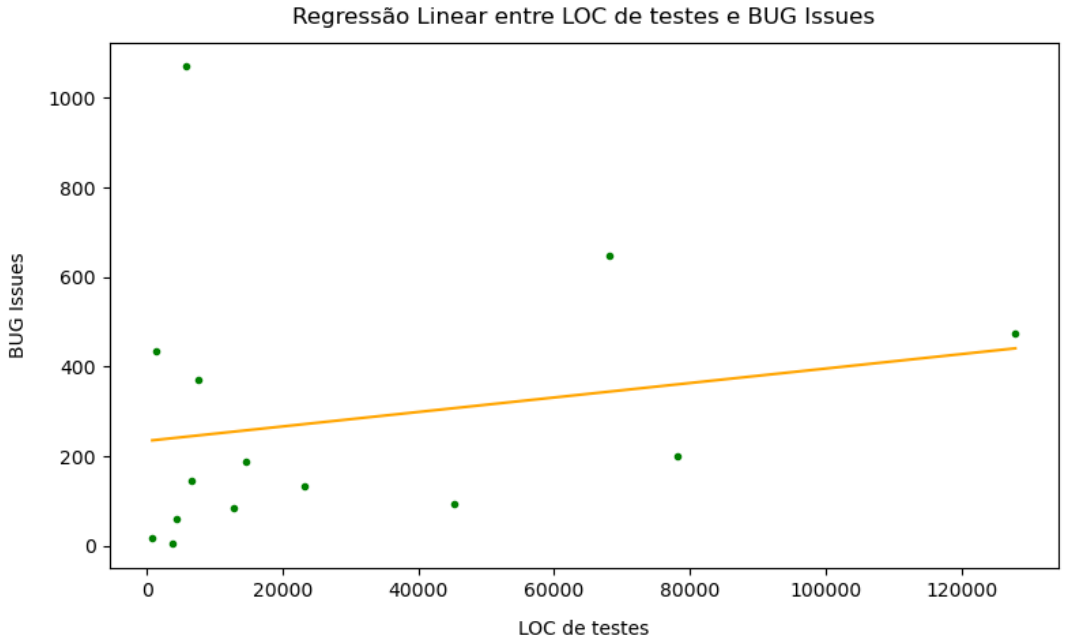
Interpretação: Correlação desprezível.

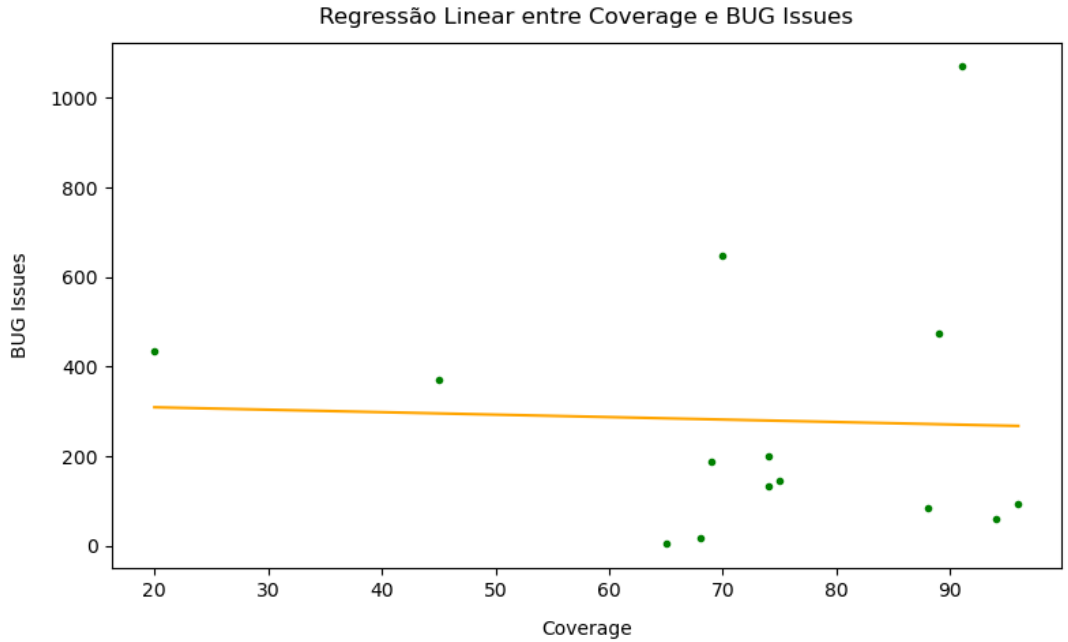
Valor R obtido: -0.21956414084848558

Interpretação: Correlação desprezível.

Gráficos de Regressão Linear e interpretação do valor R

Linguagem: JavaScript





Valor R obtido: - 0.2066207374367496

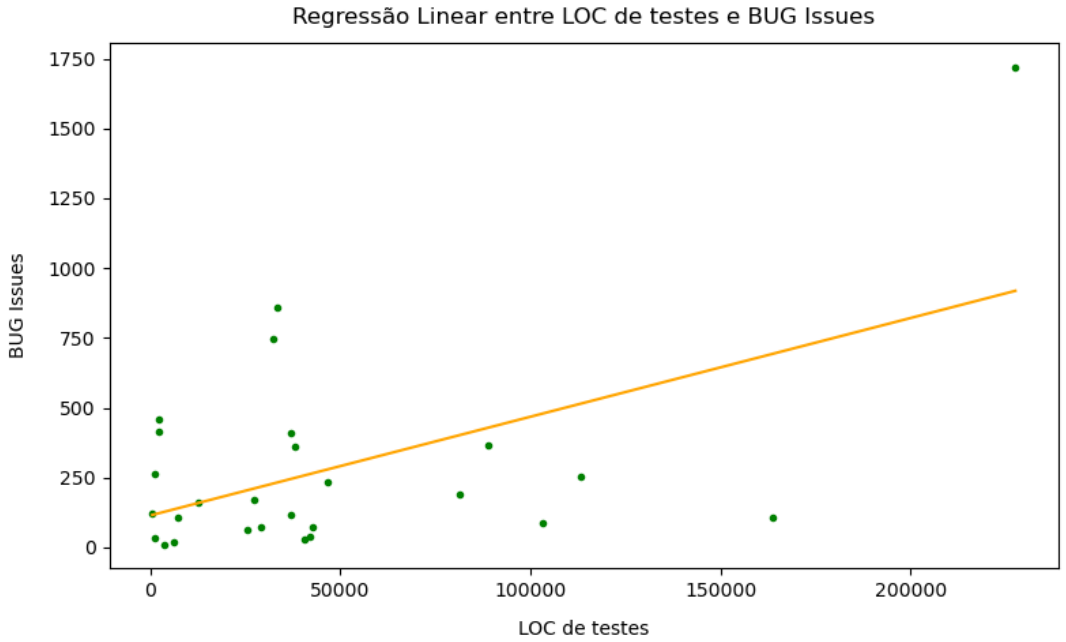
Interpretação: Correlação desprezível.

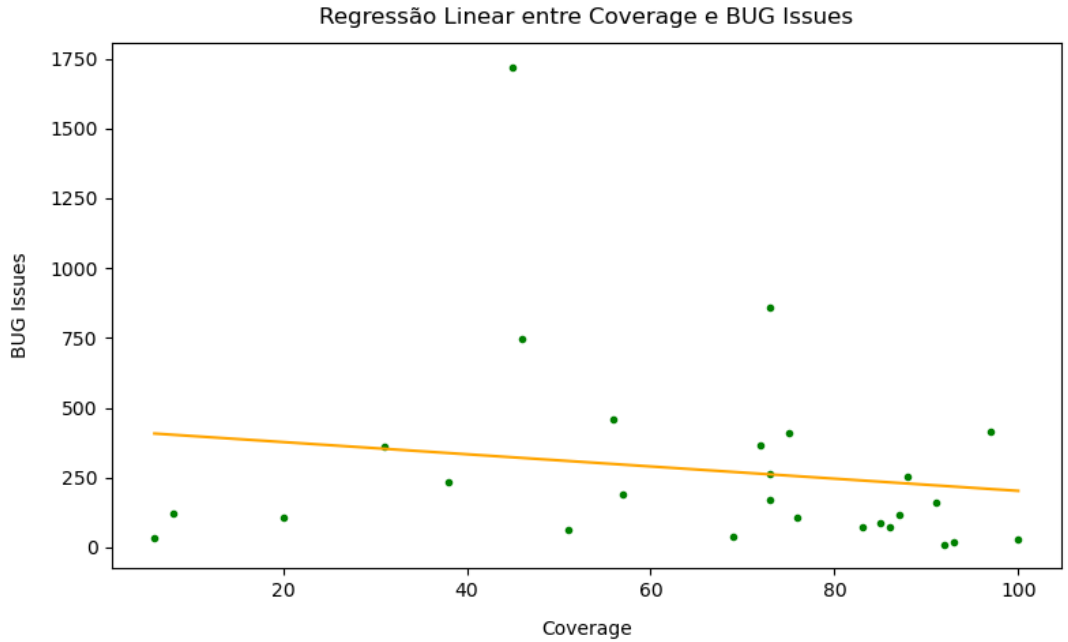
Valor R obtido: -0.03787934453847704

Interpretação: Correlação desprezível.

Gráficos de Regressão Linear e interpretação do valor R

Linguagem: C#





Valor R obtido: 0.5298728350321273

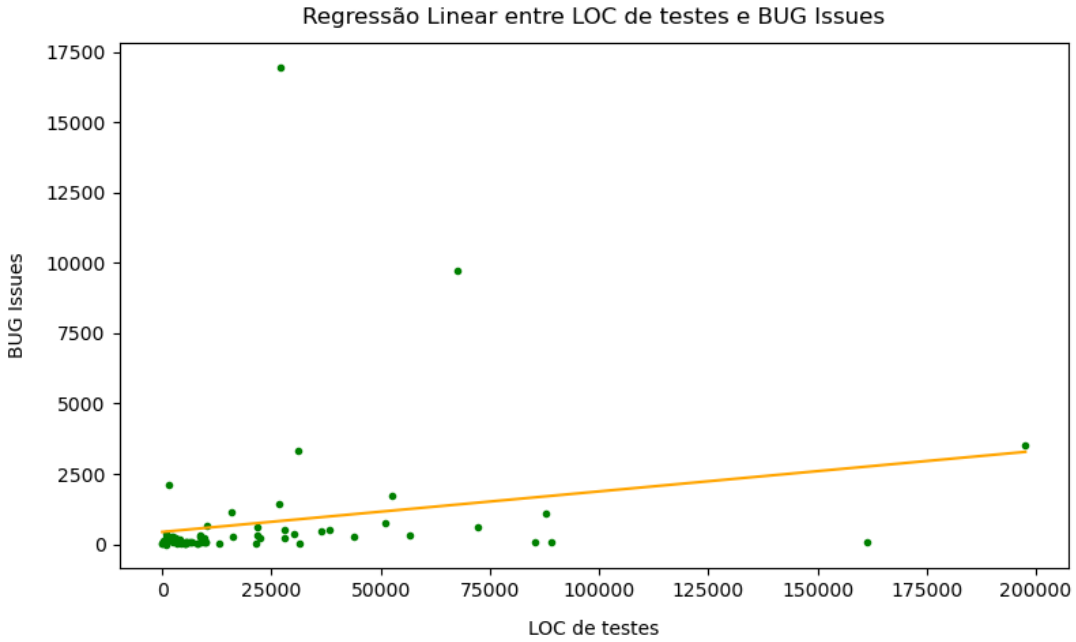
Interpretação: Correlação moderada.

Valor R obtido: -0.16366777588470346

Interpretação: Correlação desprezível.

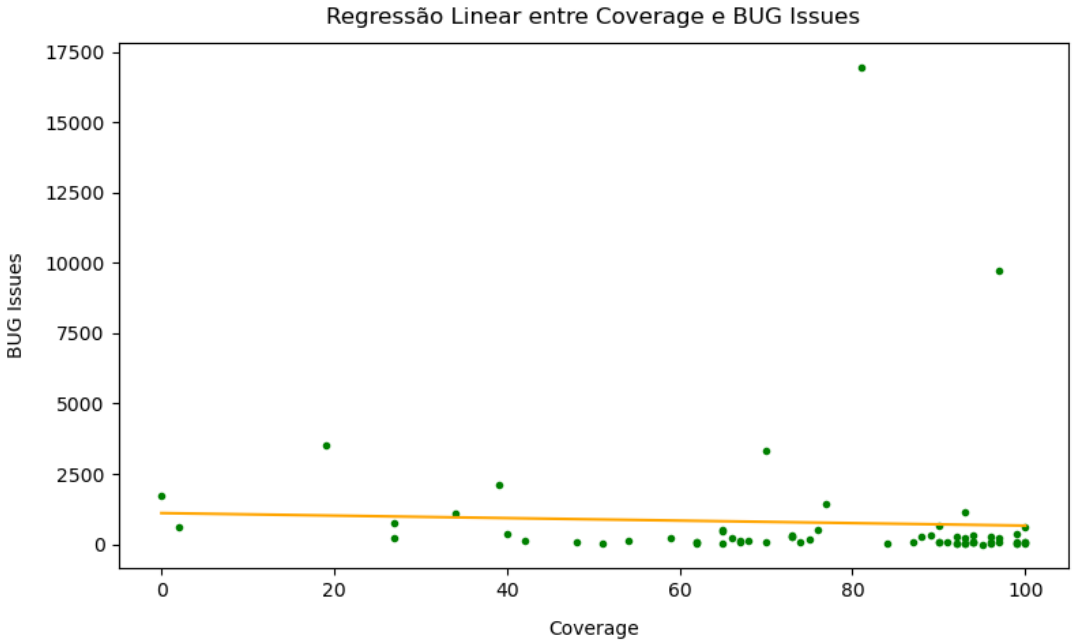
Gráficos de Regressão Linear e interpretação do valor R

Linguagem: Java



Valor R obtido: 0.21442192390533651

Interpretação: Correlação desprezível.

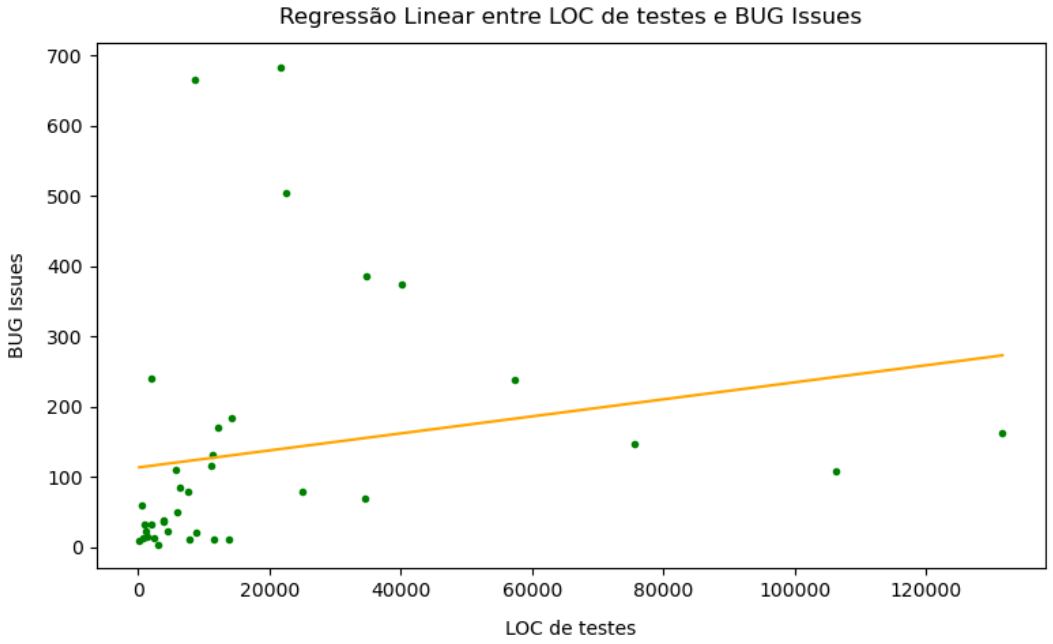


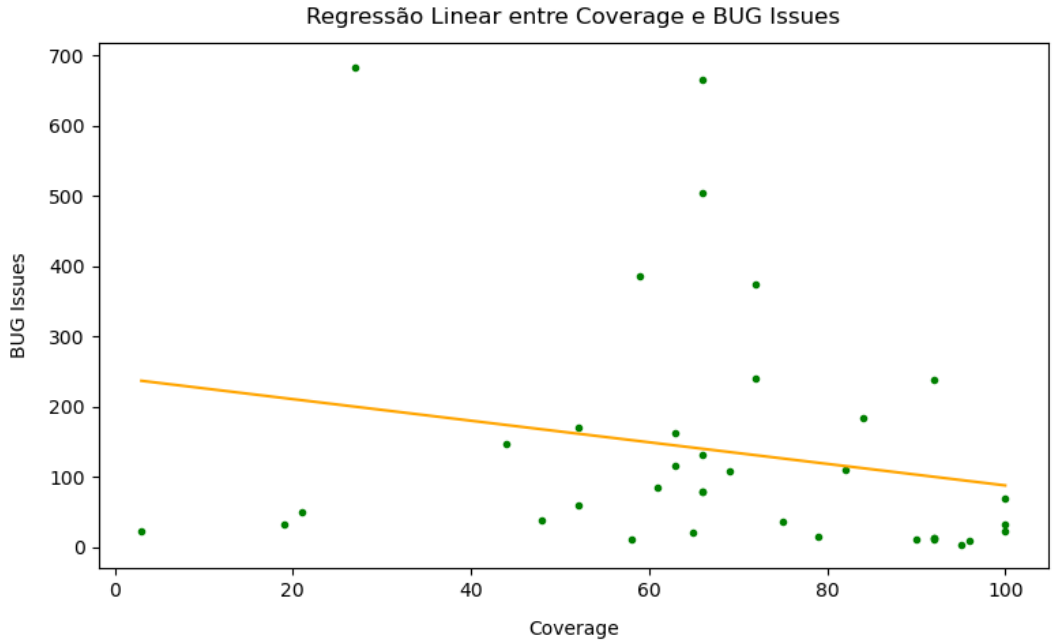
Valor R obtido: -0.04565925763262810 4

Interpretação: Correlação desprezível.

Gráficos de Regressão Linear e interpretação do valor R

Linguagem: Python





Valor R obtido: 0.2038104013911331 3

Interpretação: Correlação desprezível.

Valor R obtido: -0.2096221308092 56

Interpretação: Correlação desprezível.

Gráficos de Regressão Linear e interpretação do valor R

Linguagem: Go

Análise dos resultados

- Através da análise dos resultados, podemos, claramente, refutar a hipótese nula. Para chegar nessa conclusão, foram observadas as linguagens de forma isolada, utilizando o coeficiente de Pearson para identificar se há uma relação considerável entre a quantidade de BUGS e LOC ou cobertura de testes do projeto.

- Após esta verificação, chegou-se a conclusão que a correlação entre as variáveis de ambas as métricas anteriormente apresentadas é desprezível. Com ressalvas à linguagem Java, que obteve uma correlação moderada de aproximadamente 0,5 para a métrica LOC de testes/BUG Issues. Nas demais linguagens testadas, a interpretação do coeficiente de Pearson foi sempre desprezível.

- Desta forma, não é possível confirmar a hipótese nula para as linguagens abordadas neste estudo, confirmando, então, a hipótese alternativa de que testes unitários não impactam nos BUG Issues dos repositórios.

Ameaças à validade

- Ameaças a validade de construção:

  1. O filtro de bugs pode ser julgado, uma vez que ele pode não abranger todos os rótulos utilizados em Issues do GitHub relacionados a erros no software.

  2. Outra questão está na identificação dos arquivos de teste dos repositórios analisados, uma vez que as extensões procuradas para a contagem de linhas de código em arquivos podem não abranger todas as nomenclaturas utilizadas, portanto, há a possibilidade de contagem errada do LOC de testes por não considerar alguns arquivos de teste.

- Ameaça a validade de conclusão:

  1. Uma ameaça a conclusão está no fato de que a comunidade poderia não estar reportando Issues diretamente no repositório do GitHub e sim através de outras plataformas como o Bugzilla, por exemplo. Este fato pode influenciar diretamente na conclusão de nosso trabalho, uma vez que ele impacta diretamente na análise dos dados obtidos na execução da metodologia, influenciando a conclusão final da pesquisa.

Conclusão

- Este trabalho possibilitou a recuperação de informações em repositórios GitHub de múltiplas linguagens através de Web Crawling e uso de Scripts em Python. Os resultados explicitam que, embora haja forte correlação entre LOC de testes e número de Bugs reportados em Java, no geral, a correlação foi desprezível, contradizendo a hipótese nula de que testes unitários impactam nos BUG Issues dos repositórios.

- Portanto, com o cálculo das métricas supracitadas nos tópicos desta pesquisa, conseguimos responder todas as questões do trabalho. A primeira questão, onde há a indagação se existe relação entre a quantidade de testes unitários e quantidade de BUG Issues, temos a conclusão que esta relação inexiste. A mesma conclusão foi obtida quando calculamos a métrica que responde a segunda questão, que é descobrir a relação entre o coverage do código dos repositórios e a quantidade de bugs. Então, podemos dizer que a hipótese alternativa foi confirmada, ou seja, testes unitários não impactam nos BUG Issues dos repositórios.

Trabalhos futuros

- Demonstra-se fundamental uma análise mais aprofundada na razão pela qual esta hipótese inicial tenha sido refutada e, para trabalhos futuros, recomenda-se verificar também a participação e interação da comunidade destes repositórios juntamente com os dados obtidos neste trabalho, uma vez que este fator pode ter colaborado para a hipótese alternativa.

- Para isto, pode-se buscar em base de dados como do Bugzilla, informações necessárias a respeito de cada um dos repositórios analisados neste trabalho, visando complementar os dados já obtidos na presente pesquisa.

Link GitHub e Overleaf

- Link GitHub do nosso projeto: https://github.com/MateusSantosFonseca/TISVI-Analise-Efetividade-Testes-Unitarios

- Link Overleaf da nossa documentação em LaTeX: https://pt.overleaf.com/read/zvjsnjyccdtn

OBRIGADO!

ALGUMA PERGUNTA?