Um Estudo das Características de Qualidade de Sistemas Java

**Hugo Poletto Alacoque Gomes 1,Matheus Nolasco 2, Maria Aryene Costa 3, Lucas Santos Rosa 4**

**1** Instituto de Ciências Exatas e Informática

Pontifícia Universidade de Minas Gerais (PUC Minas)  
Belo Horizonte – MG – Brasil

{aluno1, aluno2, aluno3, lucas.rosa.1363147@sga.pucminas.br, aluno5}@sga.pucminas.br

***Resumo.*** *Neste estudo, será feita uma análise de dados de repositórios populares da plataforma Github, através da API do GraphQL, scripts Python e a ferramenta de análise CK buscando extrair medições dos 1000 repositórios Java mais populares.*

1. Introdução

A importância dos repositórios no GitHub é inegável no cenário tecnológico, pois esses repositórios desempenham um papel crucial na colaboração global e empresarial entre desenvolvedores. Por meio do GitHub, equipes podem compartilhar, revisar e aprimorar o código de projetos, controlar suas versões e auxiliar no gerenciamento.

Neste estudo, foi feita uma análise de dados de repositórios populares da plataforma Github, através da API do GraphQL disponibilizada pelos mesmos, buscando extrair medições dos 1000 repositórios Java mais populares baseado no número de estrelas. Para isso, foram utilizadas as métricas de qualidade (CBO, LCOM e DIT) e métricas de processo (Popularidade, Maturidade, Atividade e Tamanho), com o objetivo de responder às seguintes questões de pesquisa:

RQ 01. Qual a relação entre a popularidade dos repositórios e as suas características de qualidade?

RQ 02. Qual a relação entre a maturidade dos repositórios e as suas características de qualidade ?

RQ 03. Qual a relação entre a atividade dos repositórios e as suas características de qualidade?

RQ 04. Qual a relação entre o tamanho dos repositórios e as suas características de qualidade?

A partir de cada uma das perguntas, hipóteses iniciais foram formuladas com o

resultado esperado. Essas “hipóteses informais” são:

1. A maioria dos repositórios mais populares têm um nível baixo de qualidade considerando somente as métricas levantadas, porque lidam com um número grande de classes, releases e conexões entre essas classes. O LCOM e o CBO devem ser ruins para a maioria dos projetos mais populares.
2. Repositórios mais maduros possuem DIT considerável, porque possivelmente tem um número de releases alto.
3. Repositórios que têm alta atividade, devem possuir CBO considerável porque possivelmente realizam várias chamadas entre classes já existentes e as que são adicionadas.
4. Repositórios grandes devem possuir LCOM considerável, porque recebem muitas contribuições, possivelmente tem classes maiores e que tem mais de uma função.

No segundo capítulo é apresentado a metodologia implementada para o desenvolvimento do trabalho, no terceiro capítulo, são apresentados os resultados e discussões, e no capítulo final, será apresentada a conclusão do trabalho.

# 2. Metodologia

O trabalho apresentado foi conduzido utilizando a abordagem empírica quantitativa. A abordagem empírica aplicada no desenvolvimento deste trabalho, engloba observação, medição e experimentação, de acordo com Gomez & Reidl (2010). Em complemento, a pesquisa quantitativa, busca a validação das hipóteses mediante a utilização de dados estruturados, estatísticos, com análise de muitos casos representativos, segundo Mattar (2001).

Para cada questão de pesquisa, foi realizada uma comparação entre as características do processo de desenvolvimento dos repositórios e os valores obtidos para as métricas definidas nesta seção.

Para as métricas de processo, define-se:

**Popularidade:** número de estrelas

**Maturidade:** idade (em anos) de cada repositório coletado

**Atividade:** número de releases

**Tamanho:** linhas de código (LOC) e linhas de comentários

Para métricas de qualidade, entende-se:

**CBO**: Coupling between objects

**DIT**: Depth Inheritance Tree

**LCOM**: Lack of Cohesion of Methods

Para análise das métricas de popularidade, atividade e maturidade, foram coletadas informações dos repositórios mais populares em Java, utilizando a API GraphQL do GitHub e scripts em Python.

O trabalho foi desenvolvido utilizando a metodologia Scrum, sendo dividido em duas sprints. Na primeira sprint, para extrair as **métricas de processo**, foram extraídos os 1.000 repositórios mais populares em Java contendo os parâmetros name, name With Owner, url, stargazers (popularidade**)**, releases (atividade), data (maturidade**)**, utilizando a API do GraphQL e python para automação do processo de clone. Para medir o tamanho, foi desenvolvido um script em Python.

Para as **métricas de qualidade**, a extração foi salva em arquivos .csv e foram utilizadas as métricas CK para cálculo dos valores. Nesse primeiro momento, as métricas foram calculadas somente para um projeto.

Na segunda sprint, as métricas foram calculadas para todos os repositórios extraídos na primeira fase, nesta última fase, também foi elaborado o presente relatório.

# 3. Resultados

Resultados do trabalho devem ser apresentados. Consiste da descrição técnica da solução desenvolvida. Use figuras e tabelas sempre que necessário. Todas as etapas descritas na metodologia devem ter seus resultados apresentados aqui. Detalhar os resultados obtidos para cada uma das perguntas. Ilustre, os resultados utilizando visualizações de dados que ajudem a demonstrar.

# 3. Discussão

Discuta os resultados obtidos e compare com as hipóteses que foram definidas no início do trabalho.

# Sugestão: Avaliar se os projetos com menos atividade têm melhor desempenho nas métricas de qualidade do que os projetos com mais atividade.

# 4. Conclusões e trabalhos futuros

A conclusão deve iniciar resgatando o objetivo do trabalho e os principais resultados alcançados. Em seguida, devem ser apresentados os trabalhos futuros.

# Referências

C. E. Anchundia and E. R. Fonseca C., "Resources for Reproducibility of Experiments in Empirical Software Engineering: Topics Derived From a Secondary Study," in IEEE Access, vol. 8, pp. 8992-9004, 2020, doi: 10.1109/ACCESS.2020.2964587.

MATTAR, F. N. Pesquisa de marketing. 3.ed. São Paulo: Atlas, 2001. Samer Buna, GraphQL in Action , Manning, 2021.