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

{Hugo Poletto Alacoque Gomes, Matheus Nolasco Miranda Soares, Maria Aryene Costa dos Santos,Lucas Santos Rosa}@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

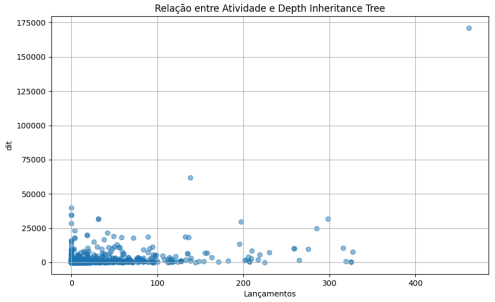
Para análise das métricas de popularidade, atividade e maturidade, foram coletadas informações dos repositórios mais populares em Java, utilizando a API GraphQL do GitHub e scripts em Python.

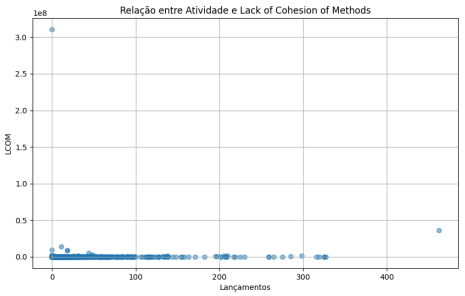
O trabalho foi desenvolvido utilizando a metodologia Scrum, sendo dividido em duas sprints. Na primeira sprint, para extrair as **métricas de processo**, foram extraídos os 1.000 repositórios mais populares em Java contendo os parâmetros name, name With Owner, url, stargazers (popularidade), releases (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.

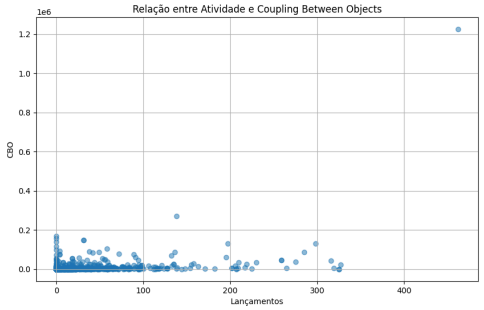
Para as **métricas de qualidade**, a extração foi salva em arquivos .csv e foram utilizadas as métricas CK para cálculo dos valores. Nesse primeiro momento, as métricas foram calculadas somente para um projeto.

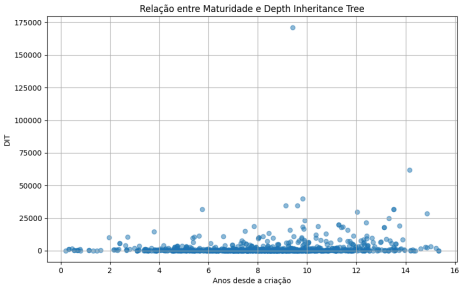
Na segunda sprint, as métricas foram calculadas para todos os repositórios extraídos na primeira fase, nesta última fase, também foi elaborado o presente relatório.

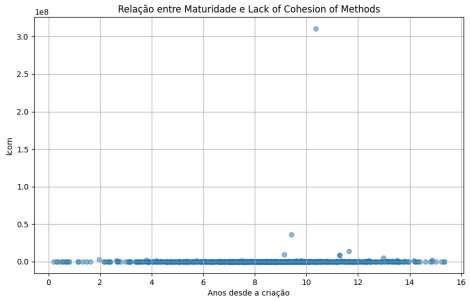
3. Resultados

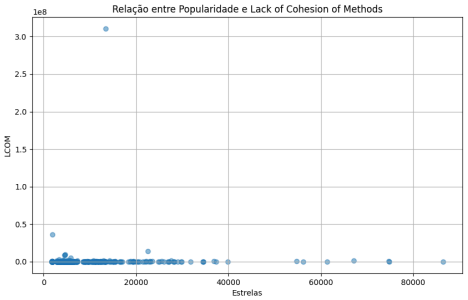


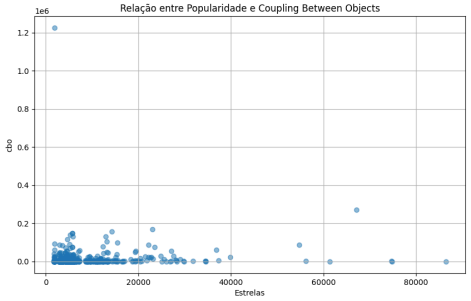


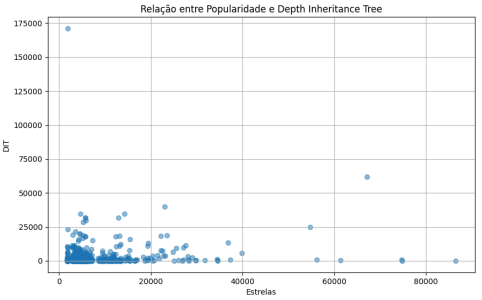


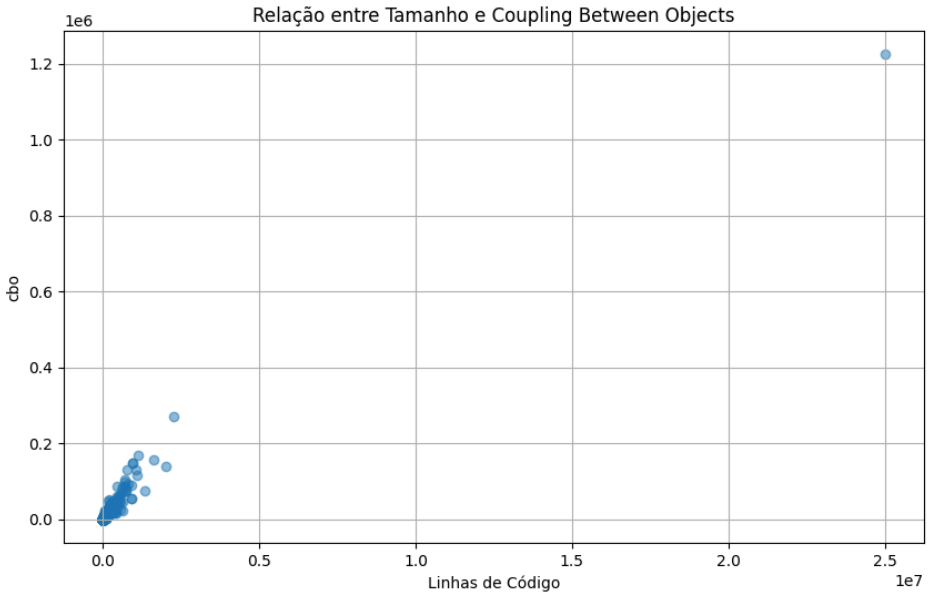


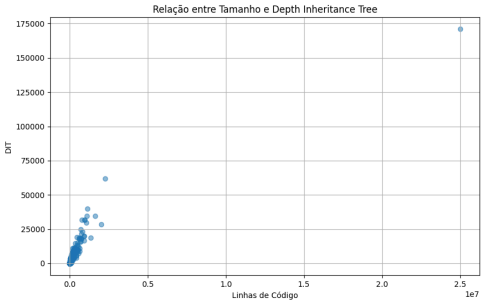

Relação entre Maturidade e Lack of Cohesion of Methods
1e8
lcom
Anos desde a criação

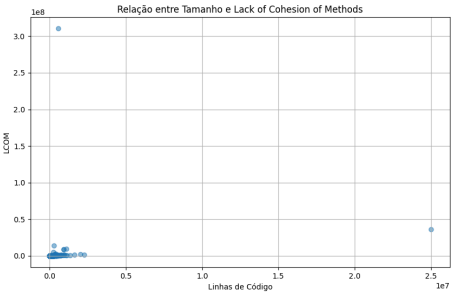

Relação entre Popularidade e Lack of Cohesion of Methods
1e8
LCOM
Estrelas


Relação entre Popularidade e Coupling Between Objects
1e6
cbo
Estrelas


Relação entre Popularidade e Depth Inheritance Tree
DIT
Estrelas


Relação entre Tamanho e Coupling Between Objects
1e6
cbo
Linhas de Código
1e7


Relação entre Tamanho e Depth Inheritance Tree
DIT
Linhas de Código
1e7


Relação entre Tamanho e Lack of Cohesion of Methods
1e8
LCOM
Linhas de Código
1e7

Os resultados mostram que não há correlação entre o número de releases, maturidade do repositório e popularidade com as métricas de qualidade. Porém, o tamanho do software está fortemente correlacionado com as métricas DIT e CBO, mostrando que quanto mais linhas de código o software tiver, maior será o acoplamento entre as classes e profundidade de dependência.

3. Discussão

A partir da análise dos resultados e comparação deles com as hipóteses formuladas na Introdução, é possível concluir que, ao contrário do que as hipóteses indicavam, a única característica dos repositórios minerados que tem correlação com as métricas de qualidade é o tamanho. Todas as outras características analisadas, como maturidade, popularidade e atividade não representaram influenciar as métricas analisadas.

Tendo isso em vista, ao revistar as hipóteses formuladas, pode-se afirmar que:

- Repositórios populares possuem ótimos índices das métricas LCOM e CBO, ambas, em sua grande maioria, estando abaixo de 0.5 para o LCOM e 0.2 para o CBO. Isso indica que repositórios populares, mesmo possuindo um grande número de realeses e classes, mantêm seu nível de qualidade alto.
- Repositórios novos possuem um DIT maior do que os maduros. Com isso, pode-se concluir que repositórios maduros, mesmo possuindo mais tempo de existência e, consequentemente, um tempo maior para receberem contribuições, mantém a extensão da árvore de herança baixa. Uma hipótese do por que isso acontece seria a manutenção realizada com o tempo fazendo com que as heranças não atinjam grandes níveis.
- Repositórios ativos possuem um índice baixo de CBO. Isso demonstra que, mesmo os repositórios com muita atividade, mantém suas classes com alta possibilidade de reuso. Uma hipótese do por que isso acontece seria a mesma da hipótese anterior, ou seja, por possuir muita atividade, também possui um alto grau de manutenibilidade.
- Repositórios grandes não possuem um alto índice de LCOM. Isso mostra que por mais que eles sejam extensos, mantém sua qualidade de código.

Além disso, é possível observar que o ponto isolado no canto inferior direito nos gráficos que correlacionam tamanho com DIT e CBO se trata de um software com número de linhas muito superior aos demais. Esse software se trata de um repositório da Google, sendo ele o Google Cloud desenvolvido em Java, tendo aproximadamente 25 milhões de linhas de código, com isso podemos observar que até mesmo software desenvolvido e gerido por uma corporação de escala mundial tem o seu código-fonte altamente acoplado e dependente entre os objetos.

4. Conclusões e trabalhos futuros

Perante os dados analisados durante a confecção do estudo, pode-se concluir que a extração desses dados através da API do Github que foi possível responder às hipóteses do início do estudo, e que essa coleta pode ser muito valiosa para identificar padrões, outliers, e agrupar informações de tendências comuns em repositórios, durante

a coleta das informações, também foi possível analisar uma demora para execução dos scripts de clone no caso de repositórios que tinham mais arquivos.

Para trabalhos futuros, a continuidade pode se dar na análise mais aprofundada dos outliers do gráficos, tentando entender o padrão entre os repositórios que saíram do comportamento normal, já que na maioria dos casos é possível perceber uma quantidade pequena que difere do comum.

Em conclusão, como o java é uma linguagem de programação bastante utilizada, que possui diversas ferramentas famosas e de grande porte, o estudo se torna importantes para desenvolvedores da linguagem entenderem se existe correlação entre as medições e o que acontece na realidade dos projetos grandes, e identificar que as as mesmas variam muito de acordo com cada repositório.

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.