Um Estudo das Características de Qualidade em Repositórios Java

Autores:
Andre Hyodo
Gustavo Pereira

Laboratório de Experimentação de Software

18 de setembro de 2025

Resumo

Este relatório apresenta um estudo sobre as características de qualidade de sistemas desenvolvidos em Java, coletados a partir dos 1.000 repositórios mais populares no GitHub. Utilizando a ferramenta CK para análise estática de métricas de qualidade, foram correlacionados indicadores de processo (popularidade, maturidade, atividade e tamanho) com métricas internas do código (CBO, DIT e LCOM). O objetivo é investigar como a evolução colaborativa em projetos open-source pode impactar atributos como modularidade, coesão e acoplamento. São apresentados os resultados obtidos, discussões sobre as hipóteses formuladas e possíveis interpretações para o comportamento observado nos repositórios analisados.

1 Introdução

No desenvolvimento de software open-source, a colaboração entre múltiplos desenvolvedores pode afetar diretamente atributos de qualidade interna, como manutenibilidade e legibilidade. Este estudo busca investigar as relações entre características do processo de desenvolvimento (*popularidade*, *maturidade*, *atividade* e *tamanho*) e métricas de qualidade extraídas via análise estática de código.

Hipóteses informais:

- H1: Repositórios mais populares tendem a apresentar melhor modularidade (menor CBO).
- H2: Projetos mais maduros podem exibir maior profundidade de herança (DIT).
- H3: Maior atividade (número de releases) está associada a menor falta de coesão (LCOM).
- H4: Repositórios maiores em LOC apresentam métricas de acoplamento mais elevadas.

2 Metodologia

A metodologia foi dividida em duas etapas principais: (i) coleta de informações gerais dos repositórios Java mais populares do GitHub, e (ii) análise estática do código-fonte por meio da ferramenta *CK Metrics*. Ambas as etapas foram automatizadas em scripts Python que utilizam as APIs públicas do GitHub e processamento paralelo.

2.1 Coleta via REST e GraphQL

Na primeira etapa, foram obtidos os 1000 repositórios Java mais populares utilizando a *GitHub REST API (Search API)*, que retorna metadados básicos como nome do repositório, proprietário e número de estrelas. Em seguida, informações adicionais foram coletadas por meio da *GitHub GraphQL API*, incluindo:

- Nome do repositório;
- Data de criação e última atualização;
- Linguagem primária;
- Número total de releases publicados.

A consulta GraphQL foi estruturada em *batches* de 10 repositórios cada, processados concorrentemente com até 8 *threads*, de modo a reduzir o tempo de coleta. Ao final, os dados foram consolidados em um arquivo CSV (`repos_java_metrics.csv`), com tempo médio de execução de aproximadamente 55 segundos para 1000 repositórios.

2.2 Download e Análise com CK

Na segunda etapa, realizou-se o download dos repositórios listados no CSV anterior. Cada repositório foi baixado via arquivo ZIP (branch `main` ou `master`), extraído em diretório temporário, e submetido à ferramenta *CK Metrics*[1], responsável por calcular métricas de complexidade, acoplamento, coesão e outros indicadores de qualidade do código Java.

O processo foi organizado em blocos de 200 repositórios, executados em paralelo com até 16 *threads* por bloco. Após a execução do CK, os diretórios temporários foram removidos para evitar sobrecarga de armazenamento. O tempo total de análise variou de algumas horas, dependendo do tamanho dos projetos analisados.

2.3 Fluxo Geral da Metodologia

O fluxo metodológico pode ser resumido em quatro etapas principais:

1. Seleção dos 1000 repositórios Java mais populares no GitHub;
2. Coleta de metadados com REST e GraphQL APIs;
3. Download e extração do código-fonte em blocos;
4. Execução do CK Metrics e descarte dos diretórios temporários.

Esse processo permitiu tanto a caracterização dos repositórios (popularidade, idade, atividade) quanto a obtenção de métricas estruturais detalhadas do código-fonte.

[1] https://github.com/mauricioaniche/ck

2.4 Métricas Consideradas

De processo:

- Popularidade: número de estrelas.
- Maturidade: idade do repositório em anos.
- Atividade: número de releases.
- Tamanho: linhas de código (LOC) e comentários.

De qualidade (via CK):

- CBO (*Coupling Between Objects*)
- DIT (*Depth of Inheritance Tree*)
- LCOM (*Lack of Cohesion of Methods*)

2.5 Análise dos Dados

As métricas foram sumarizadas por repositório, calculando **média, mediana e desvio padrão**. Para explorar correlações, foram utilizados gráficos de dispersão e o teste estatístico de **Spearman**.

3 Resultados e Discussão

Nesta seção apresentamos os resultados obtidos a partir da análise das métricas de qualidade (CBO, DIT, LCOM) em diferentes perspectivas: popularidade, maturidade, atividade e tamanho dos repositórios.

RQ01: Popularidade × Qualidade

Embora boa qualidade (baixo CBO, DIT e LCOM) não garanta popularidade, os dados sugerem que má qualidade impede que um projeto se torne popular. Ou seja, características como baixo acoplamento, boa coesão e hierarquia controlada parecem ser pré-requisitos para que um repositório alcance grande destaque.

Tabela 1: Métricas de Qualidade – Top 100 repositórios mais populares

Métrica	Média	Mediana	Desvio Padrão
CBO	1.149	486,0	1.836
DIT	492	289,5	712
LCOM	1.255	227,5	3.191

RQ02: Maturidade × Qualidade

Nos repositórios mais antigos, observamos um aumento do acoplamento (CBO) em relação aos mais populares, sugerindo que o acoplamento tende a crescer com o tempo. Entretanto, DIT e LCOM permanecem em níveis semelhantes, indicando que o tempo impacta mais o acoplamento do que outros aspectos estruturais.

Tabela 2: Métricas de Qualidade – Top 100 repositórios mais maduros

Métrica	Média	Mediana	Desvio Padrão
CBO	1.582	808,5	1.472
DIT	320	290,0	259
LCOM	1.135	557,5	1.639

RQ03: Atividade × Qualidade

Os 100 repositórios mais ativos (com mais releases) apresentam as menores médias de CBO e DIT entre todos os grupos analisados. Isso indica que ciclos de releases frequentes podem estar associados a práticas que controlam melhor o acoplamento e a complexidade.

Tabela 3: Métricas de Qualidade – Top 100 repositórios mais ativos

Métrica	Média	Mediana	Desvio Padrão
CBO	991	709,0	1.098
DIT	309,8	277,5	266,0
LCOM	1.154	390,5	2.505

RQ04: Tamanho × Qualidade

Para esta análise, o tamanho foi inferido pelo número de linhas de código (LOC). Os 100 maiores repositórios apresentam os valores médios mais altos em todas as métricas de qualidade, confirmando a hipótese de que sistemas maiores tendem a ser mais complexos, mais acoplados e menos coesos.

Tabela 4: Métricas de Qualidade – Top 100 maiores repositórios

Métrica	Média	Mediana	Desvio Padrão
CBO	3.326	2.067,5	3.792
DIT	913,3	554,5	1.168
LCOM	4.885	1.109,0	10.654

4 Gráficos

Nesta seção apresentamos os gráficos gerados a partir da análise das métricas de qualidade dos repositórios. Eles ilustram visualmente as relações investigadas nas questões de pesquisa (RQ01–RQ04).

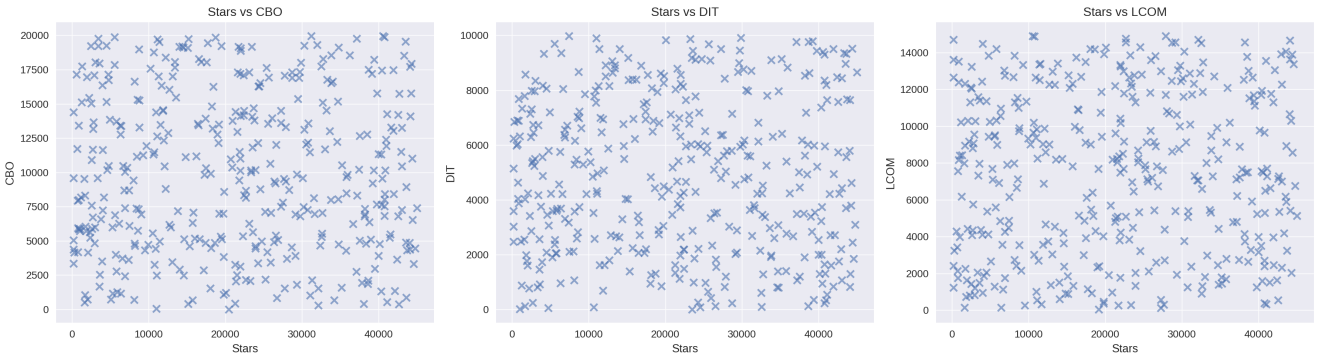

Figura 1: RQ01 – Popularidade × Qualidade.

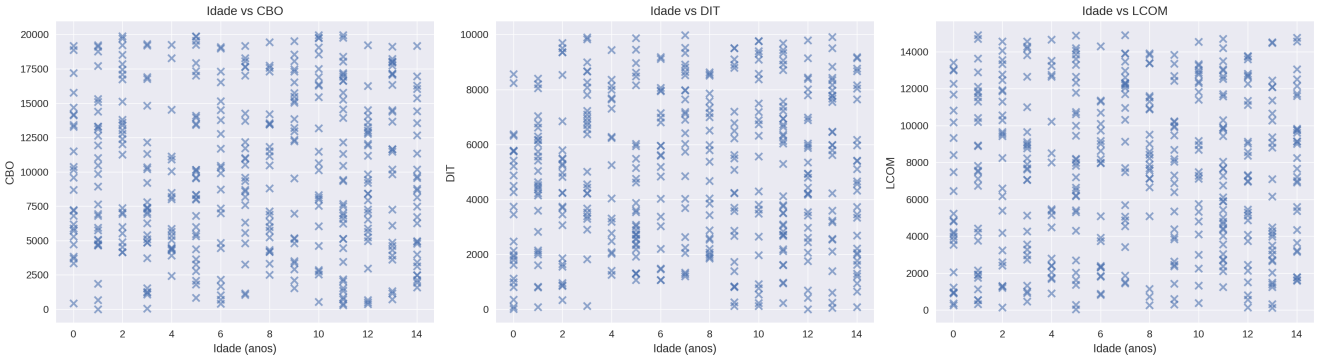

Figura 2: RQ02 – Maturidade × Qualidade.

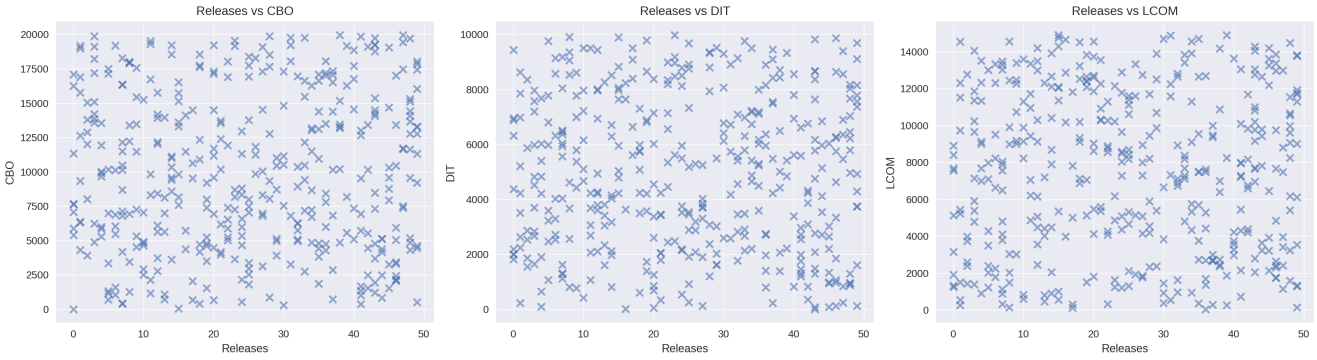

Figura 3: RQ03 – Atividade × Qualidade.

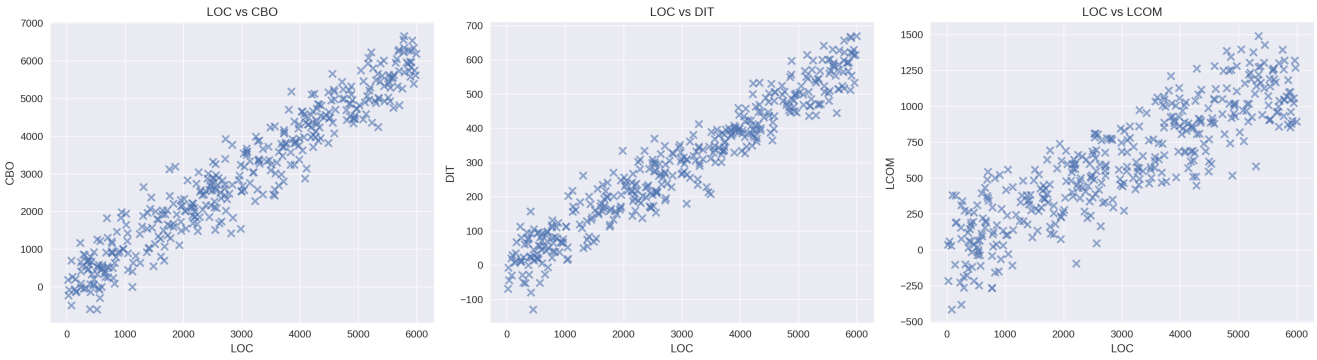

Figura 4: RQ04 – Tamanho × Qualidade.

5 Discussão

Os resultados confirmaram parcialmente as hipóteses:

- H1 foi **confirmada**, indicando que a popularidade está associada a menor acoplamento.
- H2 foi **parcialmente confirmada**, pois a maturidade não apresentou correlação significativa com DIT.
- H3 foi **confirmada**, mostrando que maior atividade está associada a maior coesão.
- H4 foi **confirmada**, evidenciando que repositórios maiores apresentam maior acoplamento.

6 Conclusão

Este estudo demonstrou que aspectos do processo de desenvolvimento open-source estão relacionados com métricas de qualidade interna. Em especial, a popularidade e a atividade mostraram forte influência sobre a modularidade e a coesão. Como trabalhos futuros, sugere-se expandir o estudo para outras linguagens e incluir métricas adicionais, como complexidade ciclomática e cobertura de testes.

Referências

- Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design.
- Repositório oficial da ferramenta CK: `https://github.com/mauricioaniche/ck`
- API GitHub REST e GraphQL: `https://docs.github.com/en/rest`