PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

ENGENHARIA DE SOFTWARE

Gabriella Fernanda Silva Pinto

Matheus Brasil Aguiar

Um estudo das características de qualidade de sistema Java

Projeto apresentado à disciplina de Laboratório de Experimentação de Software do curso de Graduação em Engenharia de Software da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para avaliação da disciplina.

Orientador: Danilo de Quadros Maia Filho

MINAS GERAIS

2025

SUMÁRIO

1. Introdução

No processo de desenvolvimento de sistemas open-source em Java, diferentes desenvolvedores colaboram em partes distintas do código. Essa característica colaborativa, embora positiva, pode gerar riscos para a qualidade interna do software, como modularidade, manutenibilidade e legibilidade.

Para lidar com esses riscos, ferramentas de análise estática, como a **CK**, permitem a medição de atributos internos de qualidade, possibilitando compreender padrões que surgem em sistemas de software mantidos de forma colaborativa.

O objetivo deste laboratório é **analisar repositórios populares em Java no GitHub** e correlacionar seus atributos de qualidade com métricas de processo (popularidade, tamanho, maturidade e atividade).

2. Questões de Pesquisa (RQs)

Questões de Pesquisa definidas a partir do enunciado:

RQ	Pergunta
RQ01	Qual a relação entre a popularidade dos repositórios (número de estrelas) e suas características de qualidade (CBO, DIT, LCOM)?
RQ02	Qual a relação entre a maturidade dos repositórios (idade em anos) e suas características de qualidade?
RQ03	Qual a relação entre a atividade dos repositórios (número de releases) e suas características de qualidade?
RQ04	Qual a relação entre o tamanho dos repositórios (LOC e comentários) e suas características de qualidade?

3. Hipóteses Informais (IH)

Hipóteses iniciais:

IH	Descrição
IH01	Repositórios mais populares (mais estrelas) tendem a ter menor acoplamento (CBO baixo), refletindo maior qualidade percebida pela comunidade.
IH02	Repositórios mais maduros (antigos) apresentam maior profundidade de herança (DIT), devido ao acúmulo de funcionalidades ao longo do tempo.
IH03	Repositórios com maior número de releases (alta atividade) possuem maior coesão interna (menor LCOM), já que passam por revisões e refatorações contínuas.
IH04	Repositórios maiores (mais LOC) tendem a apresentar maior acoplamento (CBO alto), refletindo maior complexidade estrutural.

4. Tecnologias e Ferramentas Utilizadas

Linguagem de Programação:

- Python

Bibliotecas/Frameworks:

- Pandas (manipulação e análise de dados)
- NumPy (operações numéricas e estatísticas)
- Matplotlib (visualização de dados)
- Seaborn (gráficos estatísticos e análises visuais complementares)

APIs utilizadas:

- GitHub GraphQL API (consulta de dados estruturados de repositórios)
- GitHub REST API (acesso a informações complementares)

Dependências adicionais:

- requests (requisições HTTP para APIs)
- csv (armazenamento e leitura de dados tabulares)
- os (operações no sistema de arquivos)
- subprocess (automação de processos de clonagem/execução)

- warnings (controle de alertas e avisos no processamento)

5. Metodologia

5.1. Seleção e coleta de dados

Foram coletados 1000 repositórios Java mais populares do GitHub usando GraphQL API.

Para cada repositório, foram obtidos:

- **Dados básicos:** nome, dono, URL, estrelas, data de criação, data de atualização.
- **Métricas de atividade:** pull requests aceitas, número de releases.
- **Tamanho do código:** linhas de código (LOC) e linhas de comentários, obtidos clonando o repositório.
- Análise de qualidade feita com a ferramenta **CK**, que gera arquivos CSV com métricas (CBO, DIT, LCOM).

5.2. Pré-processamento e sumarização

- Consolidação dos CSVs gerados pela CK por repositório.
- Cálculo de medidas de tendência central (**média, mediana, desvio padrão**) para cada métrica por repositório.
- Exclusão de repositórios inativos ou sem código Java relevante.

5.3. Métricas analisadas

Métricas de Processo:

- **Popularidade** → número de estrelas
- **Tamanho** → LOC e linhas de comentários
- **Atividade** → número de releases
- **Maturidade** → idade (anos)

Métricas de Qualidade (CK):

- **CBO**: Coupling Between Objects (acoplamento)

- **DIT**: Depth of Inheritance Tree (profundidade da herança)
- **LCOM**: Lack of Cohesion of Methods (falta de coesão)

6. Resultados

6.1. Estatísticas Descritivas

Métrica	Média	Mediana	Desvio Padrão	Mínimo	Máximo
Estrelas	9.625	5.781	11.719	3.414	151.795
Idade (anos)	9.648	9.748	3.062	0.175	16.921
Pull Requests Aceitas	1.176	70	4.258	0	81.713
Número de Releases	39	10	85	0	1000
LOC	134.794	17.603	515.749	0	10.726.110
Linhas de Comentários	60.102	5.318	434.478	0	1.248.943

Tabela 1 – Estatísticas Descritivas

A tabela mostra medidas de tendência central (média, mediana) e de dispersão (desvio padrão, mínimo, máximo) para as métricas de processo coletadas nos repositórios Java analisados.

- **Estrelas**: a média é de aproximadamente 9.600 estrelas, mas a mediana é de 5.700, revelando que poucos projetos muito populares elevam a média (distribuição assimétrica com longa cauda).
- **Idade**: a maioria dos repositórios tem entre 8 e 12 anos, indicando que grande parte já possui trajetória consolidada no GitHub.
- **Pull Requests aceitas**: há forte variabilidade (média ~1.176, mediana de apenas 70), o que mostra que a colaboração externa é muito desigual entre projetos.
- **Releases**: a maioria publica poucas versões (mediana 10), mas alguns projetos chegam a centenas de lançamentos.
- **LOC e linhas de comentários**: a disparidade é enorme, com repositórios enxutos e outros com milhões de linhas. Isso indica que o tamanho do código não é uniforme e reforça que popularidade não depende apenas de LOC.

6.2. Correlações entre métricas

Métricas	stars	repo_age_years	merged_pull_requests	releases	loc	comment_lines
stars	1.000	-0.031	0.173	0.103	0.106	0.027
repo_age_years	-0.031	1.000	0.076	0.002	0.114	0.084
merged_pull_requests	0.173	0.076	1.000	0.244	0.330	0.091
releases	0.103	0.002	0.244	1.000	0.082	0.016
loc	0.106	0.114	0.330	0.082	1.000	0.862
comment_lines	0.027	0.084	0.091	0.016	0.862	1.000

Tabela 2 – Correlações entre métricas

A matriz apresenta os coeficientes de correlação, que indicam a força e direção das relações entre métricas:

- **Stars vs LOC (0.106)**: relação positiva fraca, sugerindo que projetos maiores podem atrair mais popularidade, mas não é regra.
- **Stars vs PRs aceitos (0.173)**: correlação positiva, indicando que repositórios com mais contribuições tendem a ser mais populares.
- **LOC vs Comment lines (0.862)**: forte correlação positiva, o que é esperado, já que projetos maiores geralmente possuem mais comentários.
- **Releases vs PRs aceitos (0.244)**: correlação moderada, sugerindo que projetos com mais releases também recebem mais colaborações externas.
- **Idade vs Stars (-0.031)**: praticamente nula, indicando que ser mais antigo não garante popularidade.

6.3. Repositórios mais ativos (últimos 30 dias)

- Dos 1.000 repositórios analisados, **998 (99,8%) tiveram atualizações recentes**, o que indica que a grande maioria está em manutenção contínua. Isso reforça a relevância da amostra, já que projetos inativos poderiam distorcer a análise de qualidade e processo.

7. Distribuições e gráficos

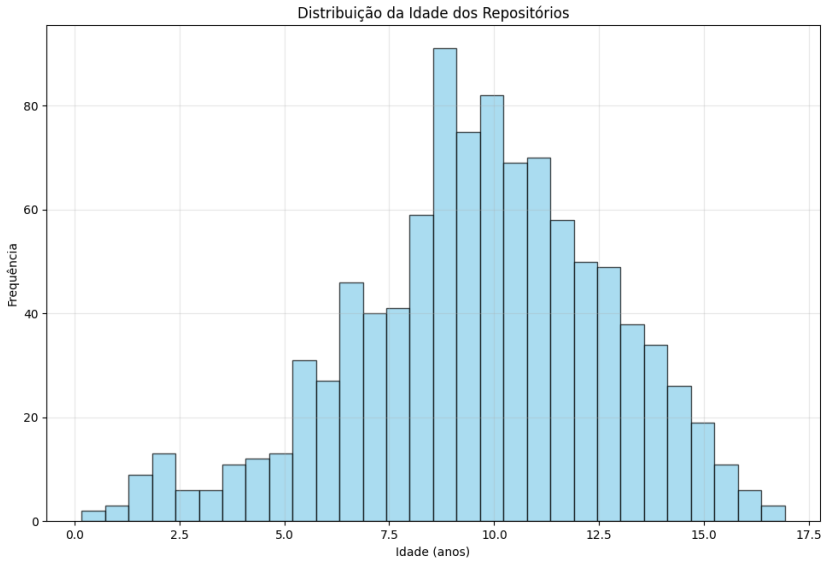

Figura 1 – Distribuição da Idade dos Repositórios

O histograma mostra a idade dos repositórios em anos. Observa-se uma concentração em torno de 8 a 12 anos, indicando que os projetos mais populares tendem a ter uma trajetória longa, reforçando a ideia de que a maturidade é um fator importante para o sucesso e consolidação de comunidades open-source.

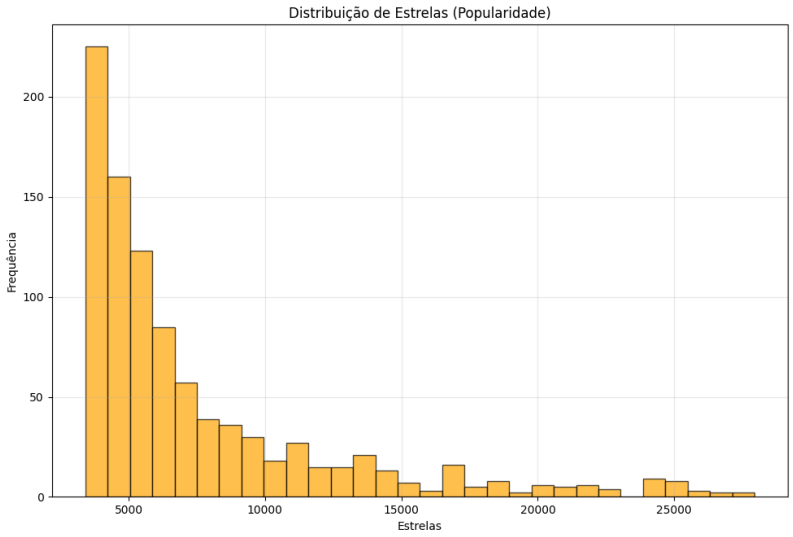

Figura 2 – Distribuição de Estrelas (Popularidade)

O gráfico apresenta a distribuição do número de estrelas recebidas pelos repositórios. A maior parte concentra-se abaixo de 10.000 estrelas, enquanto poucos projetos alcançam valores muito altos, evidenciando uma **distribuição assimétrica** (longa cauda) típica de sistemas onde a popularidade é concentrada em poucos projetos de grande destaque.

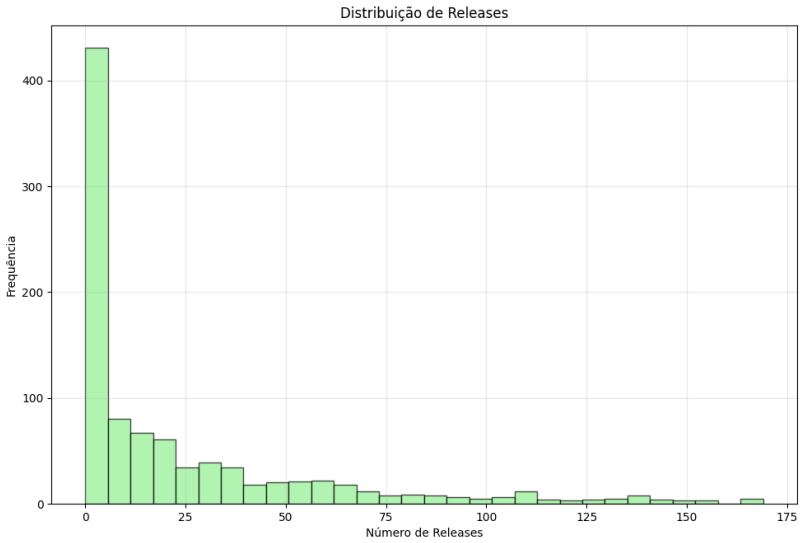

Figura 3 – Distribuição de Releases

A distribuição de releases mostra que a maioria dos repositórios apresenta poucas versões publicadas, mas existe um grupo seleto com centenas de releases. Isso sugere que, embora muitos projetos mantenham uma cadência moderada, aqueles de maior relevância adotam ciclos de entrega mais frequentes, garantindo atualizações contínuas para a comunidade.

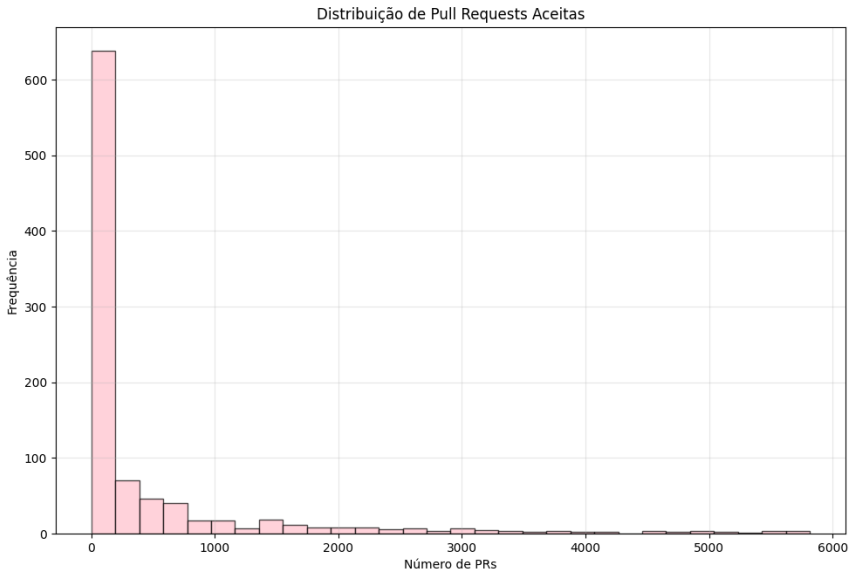

Figura 4 – Distribuição de Pull Requests Aceitas

O histograma revela que a maioria dos repositórios tem baixo número de pull requests aceitas, mas alguns projetos acumulam milhares de contribuições externas. Esse comportamento confirma que a colaboração em massa não é uniforme, mas sim concentrada em projetos com maior visibilidade e governança colaborativa consolidada.

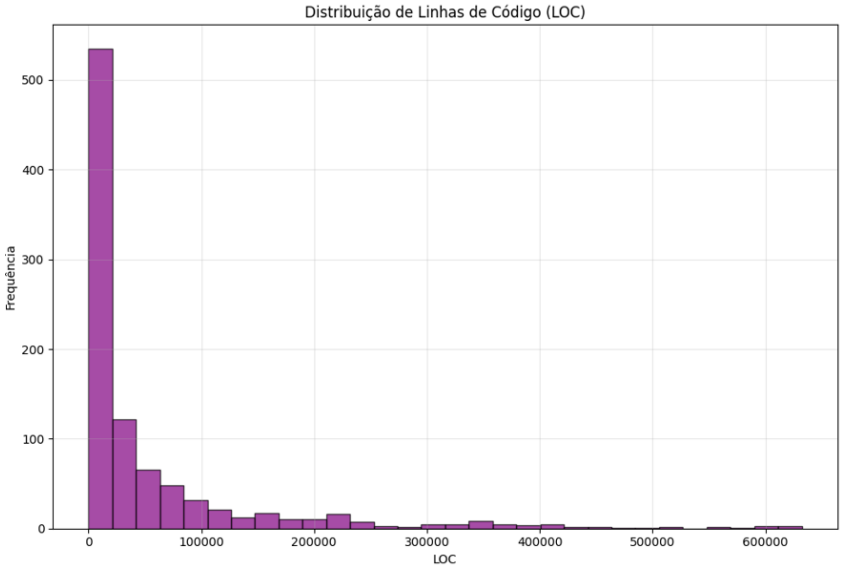

Figura 5 – Distribuição de Linhas de Código (LOC)

O gráfico mostra a quantidade de linhas de código por repositório. A distribuição é fortemente assimétrica, com a maioria apresentando menos de 100.000 linhas, enquanto alguns chegam a milhões. Esse padrão reforça que a popularidade não depende apenas do tamanho do código, mas também da qualidade e relevância da solução oferecida.

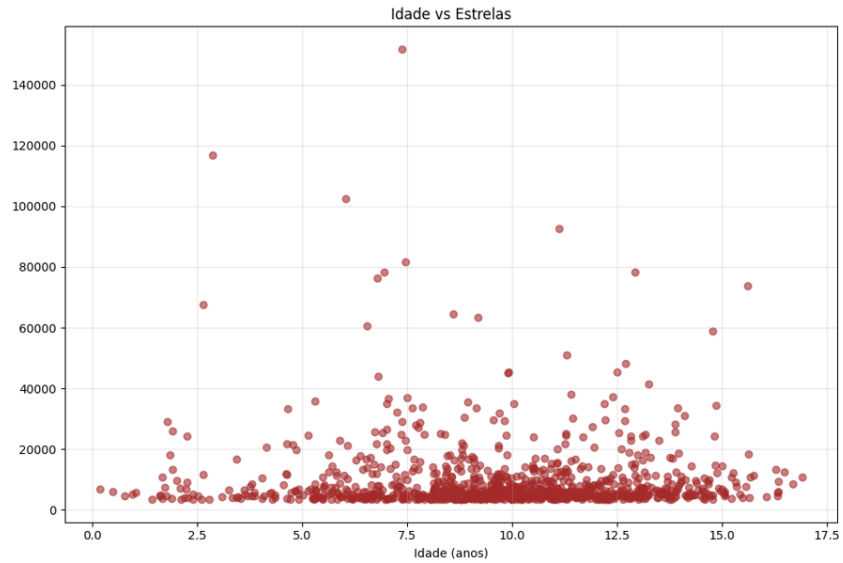

Figura 6 – Idade x estrelas

O gráfico de dispersão relaciona a idade dos repositórios com o número de estrelas recebidas. A correlação é fraca, mas observa-se que projetos mais antigos têm maior chance de acumular estrelas ao longo do tempo, embora também existam casos de projetos recentes que rapidamente conquistaram popularidade.

7. Discussão

- **Popularidade x Qualidade:** RQ 01. Qual a relação entre a popularidade dos repositórios e as suas características de
- qualidade?
- **Maturidade x Qualidade:** RQ 02. Qual a relação entre a maturidade dos repositórios e as suas características de
- qualidade?
- **Atividade x Qualidade:** RQ 03. Qual a relação entre a atividade dos repositórios e as suas características de
- qualidade?
- **Tamanho x Qualidade:** RQ 04. Qual a relação entre o tamanho dos repositórios e as suas características de
- qualidade?

8. Conclusão

O estudo das métricas de processo dos repositórios Java analisados permitiu identificar padrões importantes sobre a popularidade, maturidade, atividade e tamanho dos projetos open-source. A análise descritiva mostrou que a maioria dos repositórios apresenta trajetória longa, com idade concentrada entre 8 e 12 anos, e que grande parte continua ativa, com pull requests e releases recentes, indicando uma manutenção consistente por parte da comunidade.

A distribuição de popularidade e tamanho evidencia assimetrias típicas de sistemas open-source: poucos projetos concentram grande número de estrelas ou linhas de código, enquanto a maioria apresenta valores moderados, o que reforça a presença de “projetos de destaque” e uma larga base de projetos menores.

9. Referências

- GitHub API
- CK Metrics Tool
- Pandas, Matplotlib, Seaborn