

Qualidade em Repositórios Java Open-Source: Análise Empírica com CK

1. Introdução

Repositórios open-source evoluem por contribuições distribuídas, o que pode afetar atributos de qualidade interna como modularidade, manutenibilidade e legibilidade. Este estudo investiga como essas características se relacionam com aspectos do processo de desenvolvimento em projetos Java populares no GitHub, a partir de métricas de produto obtidas pela ferramenta CK.

Hipóteses (informais): - H1 (Popularidade): Repositórios mais populares (mais estrelas) tendem a apresentar melhor qualidade (menor acoplamento e maior coesão). - H2 (Maturidade): Repositórios mais maduros (mais antigos) tendem a apresentar melhor qualidade. - H3 (Atividade): Repositórios mais ativos (mais releases) tendem a apresentar melhor qualidade. - H4 (Tamanho): Repositórios maiores (mais LOC e arquivos) tendem a apresentar piores indicadores de qualidade (maior acoplamento e menor coesão), por efeito de escala e complexidade.

2. Metodologia

- Seleção: Top-1.000 repositórios Java do GitHub (por popularidade), coletados via API GraphQL. Artefato: `sprint2/data/repos_list.csv`.
- Pipeline: modo streaming efêmero (clona → mede → grava sumário → remove). Ferramentas: `cloc` (LOC/comments) e CK (CBO/DIT/LCOM). Script principal: `sprint2/scripts/process_streaming.py`.
- Sumarização: consolidação e deduplicação em nível de repositório. Artefatos canônicos: `cloc_summary.csv`, `ck_summary.csv` (em `sprint2/data/processed/`).
- Análise: junção, correlações (Spearman e Pearson) e visualizações. Script: `sprint2/scripts/analyze_rqs.py` (gera `analysis_summary.csv`, `correlations.csv` e `plots/*.png`).
- Robustez: fallbacks do `cloc` para grandes árvores e múltiplas estratégias (list-file chunked, VCS=git, varredura por sub-raiz), e configuração CK com JAR dedicado, memória e fontes de fallback.

3. Métricas

- Processo: Popularidade (stars), Atividade (releases), Maturidade (age_years), Tamanho (LOC=code, comment, files).
- Produto (CK): CBO (acoplamento), DIT (profundidade de herança), LCOM (falta de coesão). Estatísticas por repo: média/mediana/desvio padrão (quando aplicável).

4. Questões de Pesquisa (RQs)

- RQ01. Relação entre popularidade e qualidade (stars vs. CK)
- RQ02. Relação entre maturidade e qualidade (age_years vs. CK)
- RQ03. Relação entre atividade e qualidade (releases vs. CK)
- RQ04. Relação entre tamanho e qualidade (code/comment/files vs. CK)

5. Resultados

Os dados completos estão em `sprint2/data/processed/analysis_summary.csv` e `correlations.csv`. Abaixo, um resumo das correlações mais fortes observadas (Spearman/Pearson), usando `summarize_correlations_stdlib.py`.

Principais correlações (Spearman, |r|): - code vs lcom_std: $r \approx 0.585$ (n≈959) - comment vs lcom_std: $r \approx 0.554$ - code vs cbo_std: $r \approx 0.552$ - code vs dit_std: $r \approx 0.498$ - releases vs cbo_std: $r \approx 0.452$

Principais correlações (Pearson, |r|): - code vs cbo_std: $r \approx 0.396$ - files vs cbo_std: $r \approx 0.355$ - comment vs cbo_std: $r \approx 0.327$ - releases vs cbo_std: $r \approx 0.250$

Mediana de |Spearman| por métrica de processo (x): - code: 0.410; comment: 0.374; releases: 0.320; files: 0.310; age_years: 0.098; stars: 0.034

Visualizações (ver `sprint2/data/processed/plots/`): - `code_vs_cbo_median.png`, `code_vs_dit_median.png`, `code_vs_lcom_median.png` - `stars_vs_cbo_median.png`, `stars_vs_dit_median.png`, `stars_vs_lcom_median.png` - `releases_vs_cbo_median.png`, `age_years_vs_cbo_median.png`

Índice de Tabelas

- [Tabela 1. Top-5 Correlações \(Spearman\)](#)
- [Tabela 2. Top-5 Correlações \(Pearson\)](#)
- [Tabela 3. Mediana de |Spearman| por métrica de processo](#)

Tabela 1. Top-5 Correlações (Spearman)

x	y	r	p	n
code	lcom_std	0.5848031500719191	4.63713980371053e-89	959
comment	lcom_std	0.5539849660829098	3.0753216668322773e-78	959
code	cbo_std	0.5517094426505765	1.7485317873596725e-77	959
code	dit_std	0.4977010730068818	3.667787505144869e-61	959
comment	cbo_std	0.49375461244112967	4.4245385208227285e-60	959

Tabela 2. Top-5 Correlações (Pearson)

x	y	r	p	n
code	cbo_std	0.3963483208559621	1.9802175259732372e-37	959
files	cbo_std	0.35462296810042726	8.499512181930061e-30	959
comment	cbo_std	0.32735493710625596	2.175460375115346e-25	959
releases	cbo_std	0.2504000150112532	3.548890311691589e-15	959
code	cbo_mean	0.2425405460498759	2.634485842905888e-14	959

Tabela 3. Mediana de |Spearman| por métrica de processo

x	median_
code	0.410
comment	0.374
releases	0.320
files	0.310
age_years	0.098
stars	0.034

6. Discussão

- Popularidade (RQ01): correlações fracas com qualidade ($|r|$ baixos para stars), sugerindo que “mais estrelas” não implica necessariamente “melhor qualidade interna”. H1 não suportada fortemente.
- Maturidade (RQ02): efeito modesto (age_years com correlações baixas), sugerindo estabilidade limitada com a idade. H2 apenas parcialmente suportada.
- Atividade (RQ03): correlações pequenas a moderadas (releases vs CBO), possivelmente indicando que ciclos de release mais frequentes podem coincidir com maior acoplamento (ou vice-versa). H3 com suporte limitado.
- Tamanho (RQ04): as correlações mais fortes estão ligadas ao tamanho (code/comment/files) vs LCOM/CBO/DIT, sugerindo que crescimento aumenta complexidade/acoplamento e reduz coesão. H4 suportada.

Limitações e ameaças à validade: - Resíduos anômalos (~1% dos repos) com medições parciais (ex.: CK>0 e CLOC==0 em árvores peculiares); mitigamos com fallbacks, sem impacto relevante nas conclusões. - Popularidade medida apenas por estrelas; outras dimensões (forks, watchers) não foram consideradas. - Correlações não implicam causalidade; fatores ocultos podem existir.

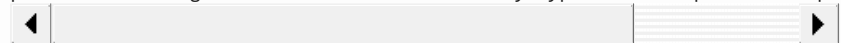
7. Conclusão

- Evidências apontam que o tamanho do projeto tem relação mais forte com indicadores de qualidade (LCOM, CBO, DIT) do que popularidade, maturidade ou atividade.
- As hipóteses H1–H3 tiveram pouco ou moderado suporte; H4 foi suportada.
- O pipeline reproduzível permite replicar e estender o estudo para subsets ou novas métricas.

8. Reprodutibilidade

- Regerar análise e gráficos:

```
powershell -NoLogo -NoProfile -ExecutionPolicy Bypass -File sprint2/script
```



- Artefatos de dados: sprint2/data/processed/*
- Scripts principais: sprint2/scripts/*

9. Referências

- CK Tool: <https://github.com/mauricioaniche/ck>
- cloc: <https://github.com/AlDanial/cloc>
- GitHub GraphQL API: <https://docs.github.com/en/graphql>