Estudo das Características de Qualidade em Sistemas Java

João Vitor Romero Sales, Lucas Randazzo

17 de Setembro de 2025

1 Introdução

O desenvolvimento de software *open-source* envolve colaboração distribuída, o que amplia o alcance e a velocidade de evolução dos projetos, mas também impõe desafios à manutenção da qualidade interna do código. Práticas como análise estática são adotadas para monitorar atributos de qualidade como modularidade e manutenibilidade.

Este trabalho propõe analisar repositórios Java populares do GitHub para investigar relações entre características do processo de desenvolvimento e métricas de qualidade interna obtidas por análise estática. O foco está em correlacionar métricas de processo (popularidade, maturidade, atividade e tamanho) com métricas de produto (CBO, DIT e LCOM) da ferramenta CK, utilizando uma amostra dos 1.000 repositórios Java mais populares.

1.1 Objetivo Geral

Analisar e caracterizar os 1.000 repositórios Java com maior número de estrelas no GitHub, investigando a relação entre atributos do processo de desenvolvimento e métricas de qualidade interna obtidas por análise estática com a ferramenta CK.

1.2 Objetivos Específicos

- **RQ 01:** Investigar a relação entre popularidade (estrelas) e métricas de qualidade (CBO, DIT, LCOM)
- **RQ 02:** Investigar a relação entre maturidade (idade) e métricas de qualidade

- **RQ 03:** Investigar a relação entre atividade (releases) e métricas de qualidade
- **RQ 04:** Investigar a relação entre tamanho (LOC e comentários) e métricas de qualidade

1.3 Hipóteses

Para cada questão de pesquisa, foram definidas hipóteses nulas (H0) e alternativas (H1) para testes de correlação.

RQ01 — Qual a relação entre a popularidade dos repositórios e as suas características de qualidade?

- H0: Não existe correlação significativa entre a popularidade dos repositórios Java (medida pelo número de estrelas) e suas métricas de qualidade (CBO, DIT, LCOM).
- H1: Existe correlação significativa entre a popularidade dos repositórios Java e pelo menos uma de suas métricas de qualidade.

RQ02 — Qual a relação entre a maturidade do repositórios e as suas características de qualidade?

- H0: A maturidade dos repositórios (idade em anos) não apresenta correlação significativa com as métricas de qualidade (CBO, DIT, LCOM).
- H1: A maturidade dos repositórios apresenta correlação significativa com pelo menos uma das métricas de qualidade.

RQ03 — Qual a relação entre a atividade dos repositórios e as suas características de qualidade?

- H0: A atividade de desenvolvimento (quantidade de releases) não apresenta correlação significativa com as métricas de qualidade dos repositórios Java.
- H1: A atividade de desenvolvimento apresenta correlação significativa com pelo menos uma das métricas de qualidade dos repositórios Java.

RQ04 — Qual a relação entre o tamanho dos repositórios e as suas características de qualidade?

- H0: O tamanho dos repositórios (linhas de código e linhas de comentários) não apresenta correlação significativa com as métricas de qualidade interna (CBO, DIT, LCOM).

- H1: O tamanho dos repositórios apresenta correlação significativa com pelo menos uma das métricas de qualidade interna.

1.4 Estrutura do Relatório

A Seção 2 descreve a metodologia adotada, incluindo o delineamento do estudo, os critérios de amostragem, os procedimentos de coleta de dados via API do GitHub e a definição das variáveis e métricas analisadas. A Seção 3 apresenta os resultados, organizados por questão de pesquisa, com estatísticas descritivas e visualizações para cada métrica. A Seção 4 discute os achados à luz das hipóteses iniciais, aponta as limitações do estudo, sugere direções para trabalhos futuros e apresenta as conclusões.

2 Metodologia

Esta seção descreve, de forma estruturada, o percurso metodológico adotado para responder às questões propostas. São detalhados o delineamento do estudo, a definição da amostra, os procedimentos de coleta de dados via API REST do GitHub, a extração das métricas de qualidade com a ferramenta CK, a mensuração das métricas de tamanho (LOC e comentários) e as etapas de sumarização e análise estatística dos dados.

2.1 Delineamento da Pesquisa

O estudo caracterizou-se como **observacional, exploratório e quantitativo**, seguindo as práticas de **Mineração de Repositórios de Software (MSR)**. Adotou-se um recorte transversal (*cross-sectional*) com *snapshot* na data de 07 de Setembro de 2025.

2.2 Definição da Amostra

A amostra contemplou os 1.000 repositórios Java com maior número de estrelas no GitHub, filtrados pela linguagem primária classificada como Java pela plataforma.

2.3 Coleta de Dados

A coleta foi realizada através da API REST do GitHub utilizando paginação. Para cada repositório, extraíram-se atributos de processo (estrelas, data de criação, *releases*) e metadados para rastreabilidade, com persistência em arquivos CSV. Em seguida, automatizou-se a clonagem dos repositórios e a execução da ferramenta CK para obtenção das métricas de qualidade.

As métricas de tamanho (LOC e linhas de comentários) foram calculadas a partir da ***branch* principal**. A contagem de linhas de comentários foi realizada por meio de uma função personalizada, `contar_linhas_comentarios`, desenvolvida pela dupla. Esta função percorre cada arquivo `.java` linha a linha, mantém um estado para detectar de forma robusta o início e o fim de blocos de comentário, e contabiliza tanto comentários de linha única (`//`) quanto de bloco (`/* ... */`) e trata eventuais exceções de I/O de forma silenciosa para não interromper a execução do script.

2.4 Desafios e Limitações da Coleta

O processo de coleta de dados apresentou alguns desafios e limitações inerentes à natureza da pesquisa:

- **Limitações da API do GitHub:** A API REST impõe limites rígidos de taxa de requisições (*rate limiting*) por minuto, o que demandou a implementação de mecanismo de *sleep* no script de coleta, aumentando significativamente o tempo total necessário para obter os dados dos 1.000 repositórios.
- **Disponibilidade dos Repositórios:** Durante a janela de coleta, alguns repositórios podem ter sido renomeados, tornados privados ou excluídos ("404 Not Found"), resultando em uma amostra final potencialmente inferior ao número-alvo inicial.
- **Complexidade da Análise de Comentários:** A heurística implementada na função `contar_linhas_comentarios`, embora robusta, pode não ser infalível. Casos extremos, como strings contendo sequências de caracteres que se assemelham a marcadores de comentário (p.ex., `"http://"` ou `"/* não é comentário */"`) ou comentários malformados, podem levar a contagens imprecisas (falsos-positivos ou falsos-negativos).
- **Características dos Projetos:** Projetos multimódulo ou que utilizam ferramentas de *build* complexas podem apresentar estruturas de diretórios não convencionais, dificultando a identificação automática de todos os arquivos fonte `.java` relevantes. Além disso, a presença de código gerado automaticamente ou de bibliotecas (*vendoring*) pode inflar as métricas de LOC e afetar as métricas CK, caso não seja devidamente filtrada.
- **Definições de Métricas:** A ferramenta CK segue definições específicas para as métricas (p.ex., a fórmula para LCOM), que podem diferir de outras ferramentas ou da interpretação teórica original. Esta é uma limitação metodológica a ser considerada na interpretação dos resultados.

2.5 Variáveis e Métricas

Métricas de Processo

- **Popularidade:** Número de estrelas (*stars*) por repositório.
- **Maturidade:** Idade do repositório (em anos), calculada a partir da diferença entre a data de criação e a data do *snapshot*.
- **Atividade:** Número total de *releases* publicadas.
- **Tamanho:** Linhas de código (LOC) e quantidade de linhas de comentários, ambas calculadas a partir da **branch principal**. A contagem de comentários foi realizada pela função `contar_linhas_comentarios`.

Métricas de Qualidade (CK)

- **CBO (Coupling Between Objects):** Mede o grau de acoplamento entre classes.
- **DIT (Depth of Inheritance Tree):** Mede a profundidade da hierarquia de herança de uma classe.
- **LCOM (Lack of Cohesion of Methods):** Mede a falta de coesão dos métodos de uma classe.

Sumarização por Repositório As métricas foram agregadas por repositório usando medidas de tendência central (mediana, média) e dispersão (desvio padrão), compondo um dataset final com uma linha por repositório.

2.6 Análise dos Dados

A análise compreendeu estatísticas descritivas e testes de correlação apropriados ao perfil dos dados, priorizando métodos robustos à não normalidade quando necessário. As relações foram avaliadas individualmente por métrica, com apoio de visualizações para inspeção da direção e força das associações.

3 Resultados

Esta seção apresenta os resultados organizados por questão de pesquisa, combinando estatísticas descritivas com testes de associação.

3.1 RQ 01: Popularidade vs Qualidade

Estatísticas descritivas (popularidade): Estrelas: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>.

Estatísticas descritivas (qualidade):

- CBO: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- DIT: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- LCOM: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>

Associação: Coeficiente entre estrelas e:

- CBO: <INSERIR COEF> (p-valor: <INSERIR>)
- DIT: <INSERIR COEF> (p-valor: <INSERIR>)
- LCOM: <INSERIR COEF> (p-valor: <INSERIR>)

Observações: <INSERIR NOTA SOBRE PADRÕES/OUTLIERS>

3.2 RQ 02: Maturidade vs Qualidade

Estatísticas descritivas (maturidade): Idade: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>.

Estatísticas descritivas (qualidade):

- CBO: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- DIT: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- LCOM: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>

Associação: Coeficiente entre idade e:

- CBO: <INSERIR COEF> (p-valor: <INSERIR>)
- DIT: <INSERIR COEF> (p-valor: <INSERIR>)
- LCOM: <INSERIR COEF> (p-valor: <INSERIR>)

Observações: <INSERIR COMENTÁRIO SOBRE NÃO LINEARIDADES>

3.3 RQ 03: Atividade vs Qualidade

Estatísticas descritivas (atividade): Releases: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>.

Estatísticas descritivas (qualidade):

- CBO: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- DIT: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- LCOM: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>

Associação: Coeficiente entre releases e:

- CBO: <INSERIR COEF> (p-valor: <INSERIR>)
- DIT: <INSERIR COEF> (p-valor: <INSERIR>)
- LCOM: <INSERIR COEF> (p-valor: <INSERIR>)

Observações: <INSERIR COMENTÁRIO SOBRE OUTLIERS>

3.4 RQ 04: Tamanho vs Qualidade

Estatísticas descritivas (tamanho):

- LOC: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- Comentários: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>

Estatísticas descritivas (qualidade):

- CBO: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- DIT: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>
- LCOM: mediana = <INSERIR>, média = <INSERIR>, DP = <INSERIR>

Associação:

- LOC vs CBO: <INSERIR COEF> (p-valor: <INSERIR>)
- LOC vs DIT: <INSERIR COEF> (p-valor: <INSERIR>)
- LOC vs LCOM: <INSERIR COEF> (p-valor: <INSERIR>)

- Comentários vs CBO: `<INSERIR COEF>` (p-valor: `<INSERIR>`)
- Comentários vs DIT: `<INSERIR COEF>` (p-valor: `<INSERIR>`)
- Comentários vs LCOM: `<INSERIR COEF>` (p-valor: `<INSERIR>`)

Observações: `<INSERIR COMENTÁRIO SOBRE COLINEARIDADE>`

4 Conclusão

Esta seção sintetiza as interpretações possíveis à luz das hipóteses formuladas para cada questão de pesquisa, considerando a amostra de 1.000 repositórios Java mais populares, a coleta via API REST do GitHub, a análise estática com CK e a sumarização por repositório das métricas de qualidade interna. Não são introduzidos valores não reportados nas seções anteriores ou não confirmados empiricamente no conjunto de dados final.

4.1 Discussão dos Resultados

Os resultados indicam que `<INSERIR INTERPRETAÇÃO DOS RESULTADOS COM BASE NOS DADOS OBTIDOS>`. Para RQ01, correlações significativas sugerem associação entre popularidade e propriedades estruturais do código. Para RQ02, associações com maturidade indicam que a idade do repositório pode influenciar mudanças estruturais. Em RQ03, relações com atividade de desenvolvimento refletem possíveis impactos dos ciclos de entrega no design interno. Para RQ04, as correlações com tamanho devem ser interpretadas com cautela devido à colinearidade intrínseca.

4.2 Limitações do Estudo

Como estudo **observacional**, a análise **não permite inferir causalidade**, apenas associação, o que impõe prudência ao extrapolar relações entre processo e qualidade interna. A seleção por popularidade (*top* 1.000 Java por estrelas) pode introduzir viés de seleção e limitar a generalização para repositórios menos populares ou com perfis de manutenção distintos, além de depender da classificação de "linguagem primária" feita pela plataforma.

No cálculo das métricas, a ferramenta CK apresenta particularidades conhecidas (por exemplo, variações de definição de LCOM e contagem de LOC baseada na AST do JDT), o que deve ser levado em conta na comparação entre repositórios e em análises de sensibilidade. A mensuração de linhas de comentários por função dedicada, ainda que robusta a blocos e comentários de linha, é baseada em heurísticas de *parsing* textual e pode superestimar ou

subestimar casos-limite (p.ex., marcadores de comentário dentro de *strings*), recomendando validações amostrais.

Por fim, procedimentos operacionais como clonagem, estrutura multimódulo, exclusão de artefatos gerados e limitações de taxa da API REST podem impactar cobertura e completude dos dados, devendo ser documentados nos scripts e nos metadados do *dataset*.

4.3 Trabalhos Futuros

Recomenda-se uma análise longitudinal, reexecutando a coleta em múltiplos *snapshots*, para observar a coevolução entre métricas de processo e qualidade interna ao longo do tempo, reduzindo a sensibilidade a instantâneos únicos. Extensões multivariadas, como modelos de regressão robusta com controle para tamanho, domínio do projeto ou organização mantenedora, podem esclarecer relações simultâneas e mitigar confundidores.

No eixo técnico, é promissor enriquecer o conjunto de métricas além de CBO, DIT e LCOM (por exemplo, RFC, WMC, TCC/LCC) e comparar resultados com variações de definição de coesão para testar a estabilidade das conclusões. Por fim, replicações em outros ecossistemas (Kotlin/Scala) ou amostras estratificadas por porte e atividade, bem como validações qualitativas com mantenedores, podem fortalecer a validade externa e explicar padrões *outliers*.

4.4 Conclusão Geral

O estudo estruturou um pipeline reprodutível que integra coleta via API REST, análise estática com CK e agregação por repositório para investigar, de forma sistemática, associações entre atributos do processo de desenvolvimento e métricas de qualidade interna em projetos Java populares. As hipóteses foram formalizadas para permitir avaliação por métrica e por questão de pesquisa, contemplando tanto resultados bicaudais quanto sínteses por família de testes, o que favorece uma leitura equilibrada de relações observadas e de sua robustez estatística.

As interpretações finais devem ser feitas à luz das limitações elencadas e, quando aplicável, acompanhadas de análises de sensibilidade e correções por múltiplas comparações, de modo a privilegiar precisão inferencial e prudência na generalização. Com a consolidação dos valores empíricos nas seções de resultados e a eventual extensão longitudinal, o trabalho fornece uma base confiável para discutir como popularidade, maturidade, atividade e tamanho se relacionam (ou não) com acoplamento, profundidade de herança e coesão no contexto de

software livre em Java.

Referências

ANICHE, M. CK: Java code metrics calculator. 2015. Disponível em: https://github.com/mauricioaniche/ck. Acesso em: 17 set. 2025.