RELATÓRIO

Um Estudo das Características de Qualidade de Sistemas Java

An Analysis of the Quality Characteristics of Java Systems

Lorrayne Oliveira [Pontifícia Universidade Católica de Minas Gerais | *lorrayne.marayze@gmail.com*]
Pedro Pires [Pontifícia Universidade Católica de Minas Gerais | *pedro.pires@gmail.com*]

Resumo. Este trabalho apresenta um estudo empírico das características de qualidade de sistemas desenvolvidos em Java, considerando os 1.000 repositórios mais populares dessa linguagem no GitHub. A popularidade, medida pelo número de estrelas, é analisada em conjunto com outros fatores relacionados ao processo de desenvolvimento, como maturidade, tamanho e atividade dos repositórios. O objetivo central é compreender como essas dimensões se relacionam com atributos de qualidade interna, avaliados por meio de métricas como acoplamento entre classes (CBO), profundidade da árvore de herança (DIT) e falta de coesão de métodos (LCOM). A investigação foi conduzida por meio da coleta automatizada de dados utilizando as APIs do GitHub, além da aplicação da ferramenta CK para análise estática do código. Essa abordagem possibilitou a extração estruturada de informações relevantes, permitindo a formulação e análise de quatro questões de pesquisa que examinam as relações entre popularidade, maturidade, atividade e tamanho dos repositórios e seus atributos de qualidade. Os resultados obtidos fornecem evidências quantitativas sobre padrões recorrentes no desenvolvimento de projetos Java de código aberto, contribuindo para a compreensão das dinâmicas de manutenção e evolução de software colaborativo em larga escala.

Abstract. This work presents an empirical study on the quality characteristics of systems developed in Java, focusing on the 1,000 most popular repositories of this language on GitHub. Popularity, measured by the number of stars, is analyzed together with other development-related factors, such as repository maturity, size, and activity. The main goal is to understand how these dimensions relate to internal quality attributes, evaluated through metrics such as coupling between classes (CBO), depth of inheritance tree (DIT), and lack of cohesion of methods (LCOM). The investigation was carried out through automated data collection using GitHub APIs, combined with the CK tool for static code analysis. This approach enabled the structured extraction of relevant information, supporting the formulation and analysis of four research questions that explore the relationships between popularity, maturity, activity, and size of repositories and their quality attributes. The results provide quantitative evidence of recurring patterns in the development of Java open-source projects, contributing to a deeper understanding of the maintenance and evolution dynamics of large-scale collaborative software.

Palavras-chave: Repositórios Java; Qualidade de Software; Métricas de Código; GitHub; Sistemas Open-Source

Keywords: Java Repositories; Software Quality; Code Metrics; GitHub; Open-Source Systems

1 Introdução

O desenvolvimento de software colaborativo, especialmente no contexto open-source, desempenha um papel importante no avanço tecnológico atual. Plataformas como o GitHub possibilitam que projetos em diferentes linguagens de programação, incluindo Java, sejam amplamente compartilhados, mantidos e evoluídos por comunidades distribuídas de desenvolvedores. Nesse cenário, aspectos de qualidade interna, como modularidade, acoplamento e coesão, tornam-se fatores críticos para garantir a manutenibilidade e a evolução saudável dos sistemas.

Este estudo tem como objetivo investigar empiricamente as principais características de qualidade dos repositórios mais populares em Java no GitHub, analisando a relação entre atributos de processo (popularidade, maturidade, atividade e tamanho), e métricas de qualidade extraídas por meio da ferramenta CK. Através da coleta automatizada de dados e da análise estática de código, busca-se compreender como fatores do processo de desenvolvimento influenciam atributos internos dos sistemas, fornecendo evidências quantitativas sobre padrões que caracterizam projetos de software Java de larga escala no ecossistema open-source.

1.1 Hipóteses Informais

Com base nas observações do ecossistema de desenvolvimento open-source, foram elaboradas as seguintes hipóteses informais para orientar a investigação:

IH01: Repositórios mais populares tendem a apresentar melhores indicadores de qualidade interna, uma vez que projetos com maior visibilidade recebem mais atenção da comunidade e passam por processos mais rigorosos de revisão e manutenção.

IH02: Repositórios mais maduros, com maior tempo de existência, podem apresentar métricas de qualidade piores, já que a manutenção contínua e a evolução do código ao longo de muitos anos tornam difícil evitar a acumulação de dívidas técnicas e problemas de modularidade.

IH03: Repositórios com maior atividade, caracterizados por um número elevado de releases, refletem um ciclo de desenvolvimento mais dinâmico, o que tende a impactar positivamente as métricas de qualidade devido à constante evolução e ajustes do sistema.

IH04: Repositórios de maior tamanho, medidos em linhas de código e comentários, apresentam maiores desafios de modularidade e coesão, o que pode resultar em métricas de qualidade menos favoráveis, como maior acoplamento en-

tre classes.

1.2 Objetivos

O objetivo principal deste trabalho é caracterizar empiricamente os repositórios em Java mais populares do GitHub através de métricas quantitativas, respondendo às seguintes questões de pesquisa:

- **RQ01:** Qual a relação entre a popularidade dos repositórios e as suas características de qualidade?

- **RQ02:** Qual a relação entre a maturidade do repositórios e as suas características de qualidade ?

- **RQ03:** Qual a relação entre a atividade dos repositórios e as suas características de qualidade?

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

Como objetivo secundário, pretende-se validar ou refutar as hipóteses informais elaboradas, contribuindo para o entendimento das características que definem o sucesso de projetos Java open-source.

2 Metodologia

2.1 Coleta de Dados

A coleta de dados foi realizada através da API GraphQL do GitHub, utilizando um script Python desenvolvido para esta análise. O processo de coleta contemplou tanto informações de processo (popularidade, maturidade, atividade e tamanho) quanto métricas de qualidade extraídas pela ferramenta CK. As etapas seguiram o seguinte fluxo:

- **Autenticação:** Utilização de token de acesso pessoal do GitHub para autenticar as requisições à API.

- **Consulta e Paginação:** Implementação de consulta GraphQL para coletar os 1.000 repositórios em Java mais populares, considerando o número de estrelas como critério de popularidade. Foi aplicado um mecanismo de paginação para garantir a coleta completa.

- **Extração de Métricas de Processo:** Para cada repositório foram obtidas informações de popularidade (número de estrelas), maturidade (idade em anos), atividade (número de releases) e tamanho (linhas de código e comentários).

- **Extração de Métricas de Qualidade:** A ferramenta CK foi utilizada para calcular métricas de acoplamento (CBO), profundidade da árvore de herança (DIT) e coesão (LCOM), gerando arquivos .csv para posterior análise.

- **Tratamento de Erros:** Implementação de mecanismos para lidar com limitações de taxa (rate limiting) e falhas de requisição, garantindo a consistência dos dados coletados.

2.2 Métricas Coletadas

Para cada repositório analisado, foram coletadas métricas referentes tanto ao processo de desenvolvimento quanto à qualidade interna do código:

- **Métricas de Processo:** - Popularidade: número de estrelas (stargazerCount) - Maturidade: idade do repositório em anos (a partir da data de criação) - Atividade: número total de releases - Tamanho: linhas de código (LOC) e linhas de comentários

- **Métricas de Qualidade (CK):** - CBO (Coupling Between Objects) — grau de acoplamento entre classes - DIT (Depth Inheritance Tree) — profundidade da hierarquia de herança - LCOM (Lack of Cohesion of Methods) — nível de coesão entre métodos da classe

2.3 Processamento dos Dados

Após a coleta, os dados brutos foram processados para calcular as métricas necessárias para responder às questões de pesquisa:

- **Idade do repositório:** Calculada em anos a partir da diferença entre a data atual e a data de criação.

- **Métricas de Qualidade:** Calculada uma média para cada tipo de métrica.

2.4 Exportação e Análise

Os dados processados foram exportados para um arquivo CSV (`repositorios_github_dados.csv`) com o objetivo de facilitar análises posteriores. A análise dos resultados será realizada utilizando medidas de tendência central, como a mediana, conforme especificado.

2.5 Limitações

Este estudo possui algumas limitações que devem ser consideradas na interpretação dos resultados. Primeiramente, a popularidade foi medida exclusivamente pelo número de estrelas no GitHub, o que pode não refletir de forma completa a qualidade, utilidade prática ou impacto real de um repositório na comunidade. Embora as estrelas sejam um indicador amplamente utilizado de popularidade Borges *et al*. [2016b], esse valor pode ser influenciado por fatores externos, como campanhas de divulgação, tendências momentâneas ou a visibilidade de projetos mantidos por grandes organizações.

Outro ponto de limitação refere-se às métricas de qualidade utilizadas (CBO, DIT e LCOM). Embora forneçam informações importantes sobre aspectos estruturais do código, essas métricas não capturam integralmente outras dimensões relevantes da qualidade de software, como usabilidade, desempenho ou facilidade de manutenção em longo prazo. Além disso, os valores obtidos não foram normalizados em função da complexidade ou do domínio de aplicação dos projetos, o que pode favorecer certos tipos de repositórios em detrimento de outros.

3 Resultados

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

A análise da RQ01 buscou compreender a relação entre a popularidade dos repositórios (medida pelo número de estrelas no GitHub) e suas características de qualidade, representadas pelas métricas CBO (acoplamento entre objetos), DIT (profundidade da árvore de herança) e LCOM (falta de coesão de métodos).

Os resultados abaixo mostram que repositórios com até 20.000 estrelas apresentam valores altos de LCOM (106.054,28), DIT (1.293,8) e CBO (4.741,9), indicando forte acoplamento e baixa coesão. Isso sugere que esses sistemas são de difícil modularidade e manutenção, o que pode explicar em parte por que, apesar de conhecidos, não alcançaram maior popularidade.

Figura 1. Popularidade x Métricas de Qualidade

Na faixa de 100.001–120.000 estrelas, os valores das métricas caem drasticamente (CBO = 1,87; DIT = 2,0; LCOM = 4,71), sugerindo projetos mais organizados, com menor acoplamento, melhor coesão e hierarquias de herança pouco profundas. Essa configuração indica arquiteturas mais simples e de melhor qualidade estrutural, possivelmente contribuindo para sua alta popularidade.

Na faixa de 140.001–160.000 estrelas, observa-se valores próximos de zero para todas as métricas, possivelmente reflexo de repositórios muito específicos, de código reduzido ou altamente modulados.

Entre as faixas intermediárias (20.001–100.000 estrelas), as métricas apresentam estabilização em patamares mais baixos (ex.: CBO entre 7,31 e 356,07; LCOM entre 37,66 e 6.940,06; DIT entre 2,44 e 92,67). Essa estabilidade sugere que muitos projetos de popularidade intermediária conseguem manter qualidade razoável, equilibrando crescimento e modularidade. Contudo, há oscilações que podem indicar que alguns projetos mais antigos acumulam dívidas técnicas, refletindo-se em aumentos pontuais no acoplamento e na perda de coesão.

Conectando com a hipótese informal IH01, que previa que projetos mais populares tenderiam a apresentar melhores indicadores de qualidade, os resultados indicam uma confirmação parcial. De fato, em faixas muito altas de popularidade, as métricas são mais baixas, refletindo boa qualidade interna. Porém, em repositórios de até 20.000 estrelas, encontramos o oposto: alto acoplamento e baixa coesão, o que sugere que a menor popularidade pode estar associada a limitações de qualidade estrutural ou ao domínio tecnológico dos projetos. Assim, a análise mostra que a popularidade sozinha não explica ou se relaciona diretamente com a qualidade do código.

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

Os resultados revelam um padrão que contraria parcialmente a hipótese informal IH02. Nos repositórios mais jovens (menos de 1 ano), observamos valores relativamente baixos para todas as métricas: CBO (95,65), DIT (24,63) e LCOM (749,2), indicando estruturas iniciais mais simples e organizadas.

À medida que os projetos amadurecem (2-4 anos e 4-6 anos), há um crescimento gradual nas métricas, com LCOM atingindo 6.354,05 e CBO chegando a 369,65 na faixa de 4-6 anos. Este aumento pode sugerir o início da acumulação de complexidade estrutural do repositório, conforme o código evolui e novas funcionalidades são adicionadas.

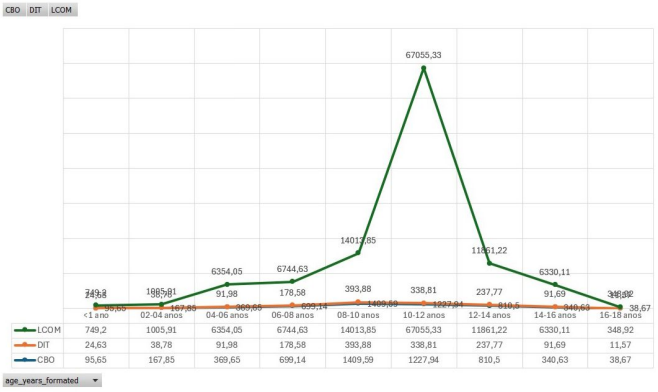

Figura 2. Maturidade x Métricas de Qualidade

O ponto crítico ocorre na faixa de 10-12 anos, onde podemos observar o pico máximo de todas as métricas: LCOM atinge 67.055,33, DIT chega a 338,81 e CBO alcança 1.227,94, indicando que estes repositórios acumulam um alto número de dívidas técnicas, problemas de modularidade e alta complexidade estrutural.

Por fim, nos repositórios mais maduros (14-16 anos e 16-18 anos), as métricas decrescem, com valores próximos aos projetos jovens (LCOM = 348,92; CBO = 38,67; DIT = 11,57 na faixa mais madura). Este padrão pode indicar que projetos que sobrevivem por muito tempo passam por processos de refatoração, remoção de código legado ou representam sistemas altamente especializados que mantiveram sua simplicidade arquitetural.

A hipótese IH02 é confirmada parcialmente: há uma fase intermediária onde a maturidade está associada à deterioração da qualidade, mas os projetos mais antigos apresentam métricas melhores, possivelmente devido a processos de manutenção evolutiva ou seleção natural de arquiteturas sustentáveis.

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

A análise da RQ03 examinou como a atividade dos repositórios, medida pelo número de releases, influencia as métricas de qualidade estrutural.

Os dados mostram uma relação que desafia a hipótese IH03. Repositórios com baixa atividade (0-250 releases) apresentam valores elevados nas três métricas: LCOM (67.879,14), CBO (1.401,32) e DIT (429,11). Esta configuração sugere projetos com desenvolvimento menos frequente, possivelmente acumulando problemas estruturais devido à falta de manutenção contínua.

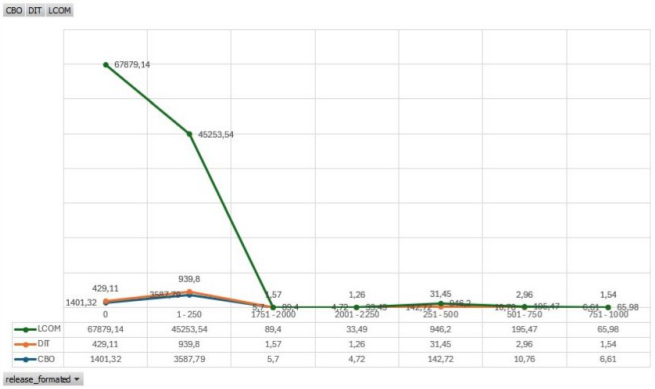

Figura 3. Atividade x Métricas de Qualidade

Figura 4. Tamanho em Linhas de Comentário x Métricas de Qualidade

Na faixa intermediária de atividade (1.251-2.000 releases), observa-se uma redução significativa em todas as métricas: LCOM cai para 89,4, CBO para 5,7 e DIT para 1,57. Este padrão indica que um nível moderado de atividade está associado a melhor qualidade estrutural, sugerindo que releases regulares permitem ajustes incrementais e manutenção da qualidade.

Entretanto, na faixa de alta atividade (2.001-2.250 releases), as métricas sobem novamente, com CBO aumentando para 33,49 e LCOM para 1,26, mantendo DIT baixo. Nas faixas de atividade mais extremas (acima de 2.250 releases), as métricas se estabilizam em valores baixos, com oscilações mínimas.

A hipótese IH03, que previa impacto positivo da alta atividade na qualidade, não pode ser confirmada. Os resultados mostram que existe um ponto de atividade (faixa intermediária) onde as métricas são boas. Muito pouca atividade está associada a problemas de qualidade, possivelmente devido à falta de manutenção. Atividade extremamente alta também pode introduzir complexidade adicional, enquanto atividade moderada parece proporcionar o equilíbrio ideal entre evolução contínua e manutenção da qualidade estrutural.

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

A análise da RQ04 investigou como o tamanho dos repositórios, medido tanto em linhas de código quanto em linhas de comentários, se relaciona com as métricas de qualidade estrutural CBO, DIT e LCOM.

Os resultados para linhas de comentários mostram uma relação que confirma parcialmente a hipótese IH04. Repositórios com mais de 100.000 linhas de comentários apresentam valores extremamente elevados para todas as métricas: LCOM atinge 64.434,53, CBO chega a 533 e DIT alcança 120,43. Estes valores indicam sistemas complexos, com alto acoplamento entre classes, baixa coesão e hierarquias de herança.

Nas faixas intermediárias (1.001-100.000 linhas de comentários), as métricas apresentam valores elevados mas oscilantes: LCOM varia entre 18.501,5 e 23.382,02, CBO entre 1.563,87 e 1.827,72, e DIT entre 403,42 e 516,24, sugerindo que maior volume de documentação está associado ao aumento da complexidade do sistema. Já nas faixas menores (até 1.000 linhas de comentários), as métricas se estabilizam em valores consideravelmente mais baixos (LCOM entre 1.257,59 e 6.887,58, CBO entre 83,79 e 985,98), indicando que repositórios com documentação mais enxuta tendem a apresentar melhor qualidade estrutural.

A análise por linhas de código revela um padrão mais acentuado. Repositórios pequenos (até 1.000 linhas) apresentam valores relativamente baixos e estáveis: LCOM (1.618,4), CBO (110,04) e DIT (24,19). Esta configuração sugere que projetos pequenos conseguem manter estruturas simples e organizadas.

Figura 5. Tamanho em Linhas de Código x Métricas de Qualidade

Conforme o tamanho aumenta, as métricas crescem gradualmente até atingirem o pico na faixa de 100.001-1.000.000 linhas de código: LCOM alcança 75.398,48, CBO atinge 1.099,92 e DIT chega a 258,62, confirmando fortemente a hipótese IH04 de que repositórios muito grandes enfrentam sérios desafios de modularidade e acoplamento. E nos repositórios ainda maiores, as métricas caem drasticamente para valores próximos de zero, podendo indicar sistemas que passaram por processos rigorosos de refatoração ou representam arquiteturas altamente especializadas.

A hipótese IH04 é confirmada pelos dados, especialmente na análise por linhas de código. Os resultados mostram que existe uma correlação clara entre o aumento do tamanho dos repositórios e a deterioração das métricas de qualidade, com o pico de problemas ocorrendo em repositórios de tamanho entre 100.001 a 1.000.000 linhas.

A análise por linhas de comentários também confirma a hipótese, mostrando que sistemas que requerem uma documentação extensa (mais de 100.000 linhas de comentários) estão associados a alta complexidade estrutural. Isso sugere que a necessidade de um alto número de comentários pode ser um indicador de arquitetura complexa que requer explicações detalhadas.

Os dados revelam que tanto o volume das linhas de código quanto a quantidade de linhas de comentário são indicadores importantes da complexidade estrutural, com repositórios de tamanho intermediário geralmente apresentando o melhor equilíbrio entre funcionalidade e qualidade de código.

4 Conclusão

Com base nos resultados deste trabalho, observamos que as relações entre popularidade, maturidade, atividade e tamanho dos repositórios com suas métricas de qualidade não seguem padrões. A popularidade mostrou que repositórios muito populares têm métricas melhores, mas os de popularidade média apresentaram muita variação. Para maturidade, encontramos que repositórios de 10-12 anos têm as piores métricas, mas os mais antigos conseguem manter qualidade melhor. A atividade revelou que existe um nível ideal de releases onde as métricas são melhores, e tanto pouca quanto muita atividade podem ser problemáticas.

O tamanho dos repositórios foi o fator que mais influenciou a qualidade do código. Repositórios maiores, tanto em linhas de código quanto em comentários, apresentaram consistentemente piores valores para as métricas CBO, DIT e LCOM. Isso confirma que sistemas grandes enfrentam mais dificuldades para manter boa modularidade e baixo acoplamento. Os resultados mostram que gerenciar o crescimento do código é um desafio importante no desenvolvimento de software, e que projetos grandes precisam de mais cuidado na organização da arquitetura para evitar problemas de qualidade.

Referências

Ait Houaich, Y. *et al*. (2015). Measuring the maturity of open source software. *ResearchGate*. Preprint.

Badreddin, O., Keshta, I., and Bekhet, S. (2014). A study on maturity model of open source software community to estimate the quality of products. *Procedia Computer Science*, 35:1282–1291. DOI: 10.1016/j.procs.2014.08.233.

Borges, H., Hora, A., and Valente, M. T. (2016a). Predicting the popularity of github repositories. In *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 1–10. ACM. DOI: 10.1145/2972958.2972966.

Borges, H., Hora, A., and Valente, M. T. (2016b). Understanding the factors that impact the popularity of github repositories. In *Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 334–344. IEEE. DOI: 10.1109/SANER.2016.31.

Ferreira, K. R., Bigonha, R. S., Bigonha, M. A., Mendes, L. M. G., and Almeida, H. (2012). A tool for source code analysis of java programs. In *Proceedings of the 2012 Brazilian Symposium on Software Engineering*, pages 1–10.

FINOS (2024). The open source maturity model (osmm). Financial Industry Open Source - Open Source Readiness.

GitHub (2024). Graphql api reference. Documentação oficial da API GraphQL do GitHub.

Hostinger (2025). As linguagens de programação mais usadas em 2025. Acesso em: 27 ago. 2025.

Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2 edition.