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

Referências