PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Relatório Final - Laboratório 2
Laboratório de Experimentação de Software

Bruno Rocha Corrêa Urbano
Henrique Dani Franco Nezio
Pedro de Barros Alves

Belo Horizonte 2025

Introdução

Este trabalho tem como objetivo investigar como diferentes características do processo de desenvolvimento — como popularidade, maturidade, atividade e tamanho dos repositórios — podem estar relacionadas com atributos de qualidade do código-fonte, mensurados por métricas como CBO (Coupling Between Objects), DIT (Depth of Inheritance Tree) e LCOM (Lack of Cohesion of Methods).

As hipóteses informais levantadas pelo grupo foram as seguintes:

- **H1 (RQ01)**: Repositórios mais populares tendem a apresentar melhores métricas de qualidade, pois recebem maior atenção da comunidade e maior cuidado com a manutenção.

- **H2 (RQ02)**: Repositórios mais maduros tendem a ter um código mais organizado e coeso, pois passaram por mais ciclos de evolução e refatoração.

- **H3 (RQ03)**: Repositórios mais ativos tendem a apresentar menor acoplamento e maior coesão, por conta de processos contínuos de revisão e melhoria.

- **H4 (RQ04)**: Repositórios maiores tendem a apresentar piores métricas de qualidade, especialmente em termos de coesão, devido à complexidade gerada pelo volume de código.

Metodologia

Para responder às questões de pesquisa propostas, a metodologia foi dividida em quatro etapas principais:

1. Seleção de Repositórios

Selecionamos os 1.000 repositórios mais populares escritos em Java disponíveis no GitHub, utilizando a API GraphQL.

2. Coleta de Dados

Foram coletados dois tipos de dados:

- Métricas de processo:
    - Popularidade: número de estrelas
    - Maturidade: tempo de existência do repositório
    - Atividade: número de releases
    - Tamanho: número de linhas de código e de comentários

- Métricas de qualidade de software:
    - CBO (Coupling Between Objects)
    - DIT (Depth of Inheritance Tree)
    - LCOM (Lack of Cohesion of Methods)

A ferramenta CK foi utilizada localmente para analisar o código de cada repositório Java, gerando arquivos .csv com os resultados das métricas por classe.

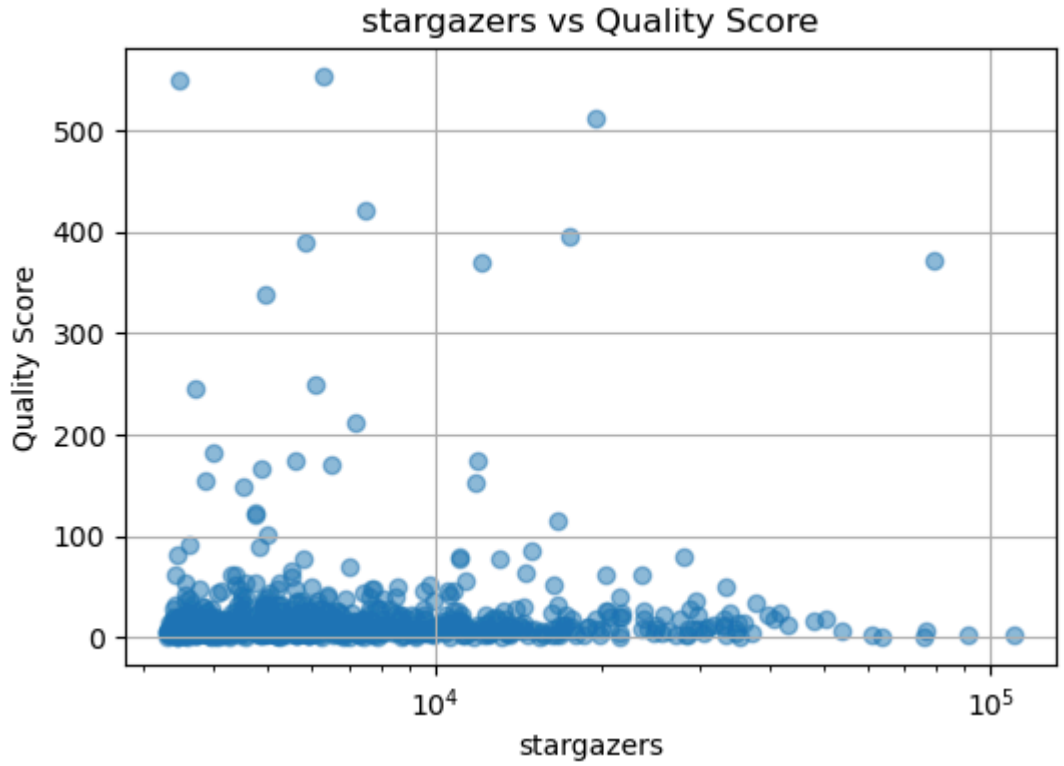
3. Tratamento e Análise dos Dados

Todos os dados foram organizados e tratados em notebooks rodando Python utilizando a biblioteca Pandas. Os dados foram sumarizados através de média, mediana e desvio padrão.

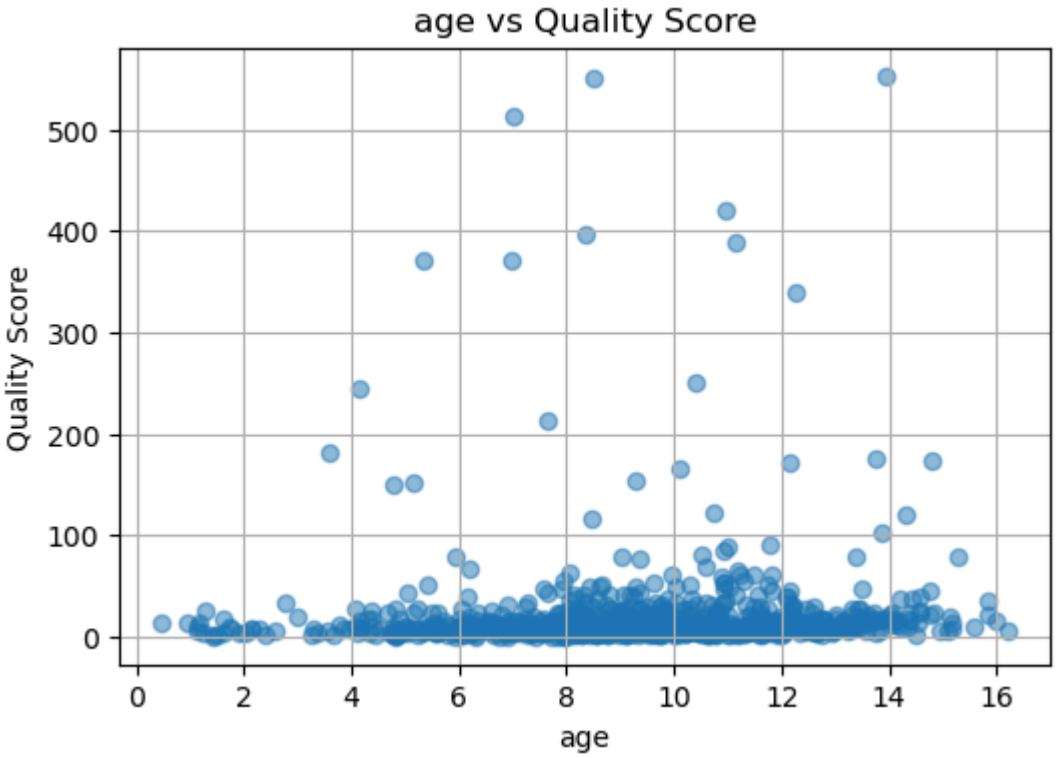
No nosso estudo, utilizamos o Quality Score para representar a qualidade interna do código nos repositórios Java analisados. No entanto, ao interpretar os dados, percebemos que quanto maior o Quality Score, pior a qualidade do código. Isso acontece porque a métrica é influenciada diretamente por características negativas, como alto acoplamento (CBO), baixa coesão (LCOM) e hierarquia de

herança profunda (DIT).

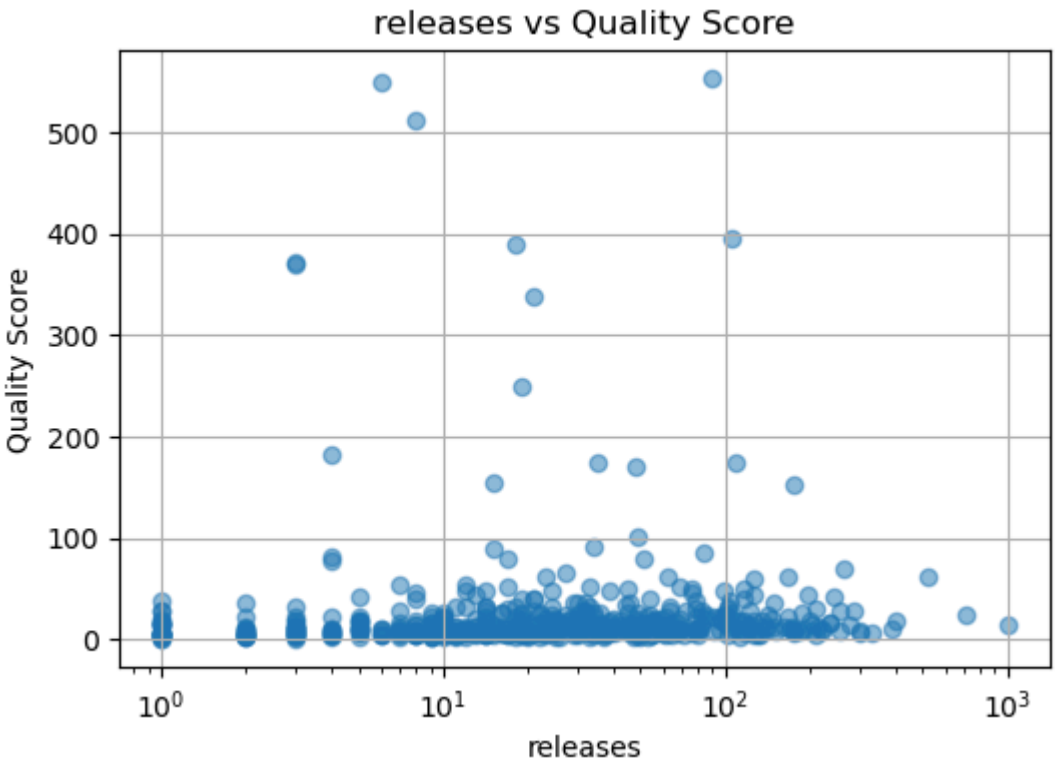
A primeira análise realizada foi uma comparação entre o número de estrelas com o Quality Score. Nela podemos ver que o Quality Score independe do número de estrelas de um repositório (e existe uma tendência de que repositórios menos populares possuírem maior Quality Score).



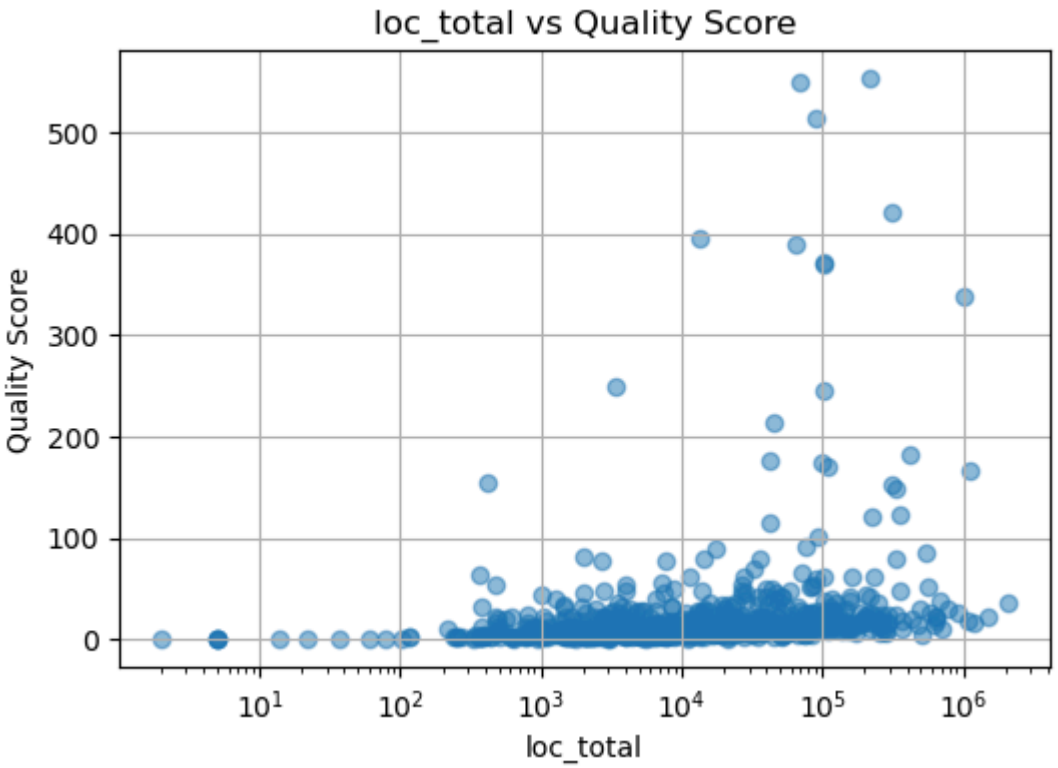
A segunda análise foi comparando a idade dos repositórios com o Quality Score. Aqui percebemos que o Quality Score também independe da idade do repositório (apesar de que repositórios com idades entre 0 a 3 anos raramente possuem um Quality Score alto).

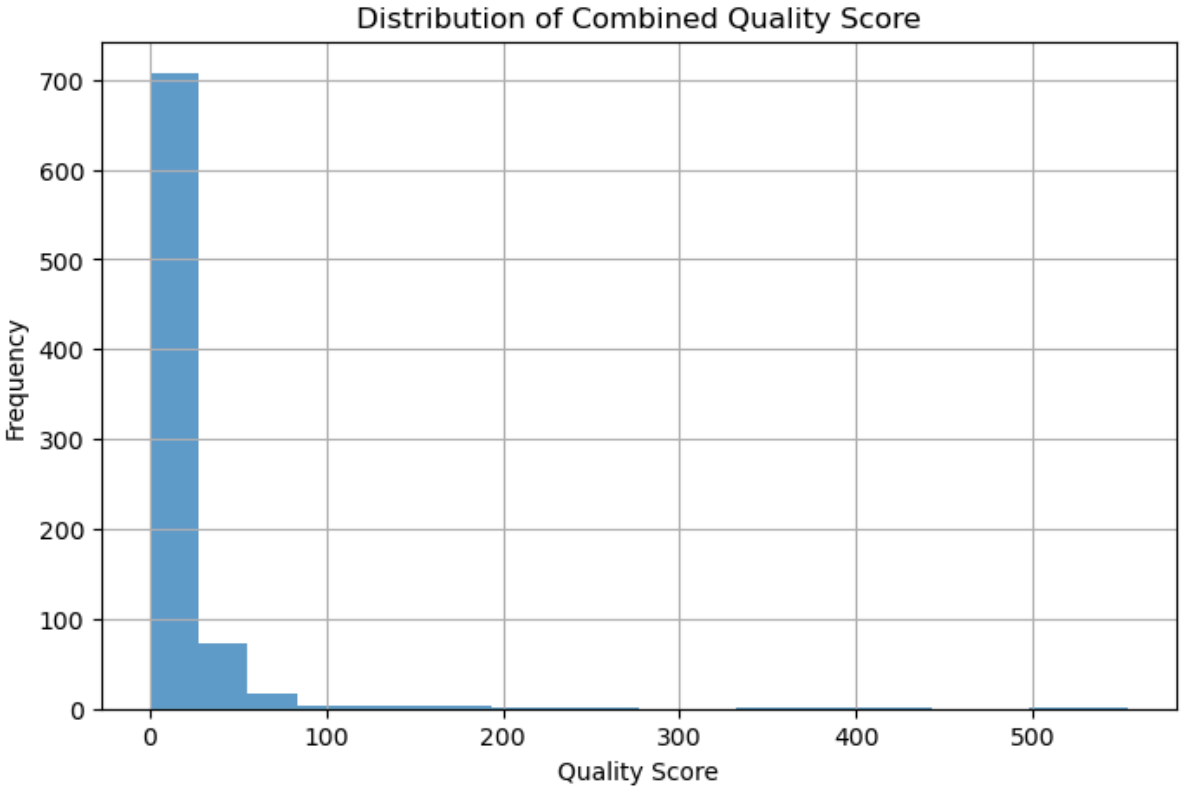


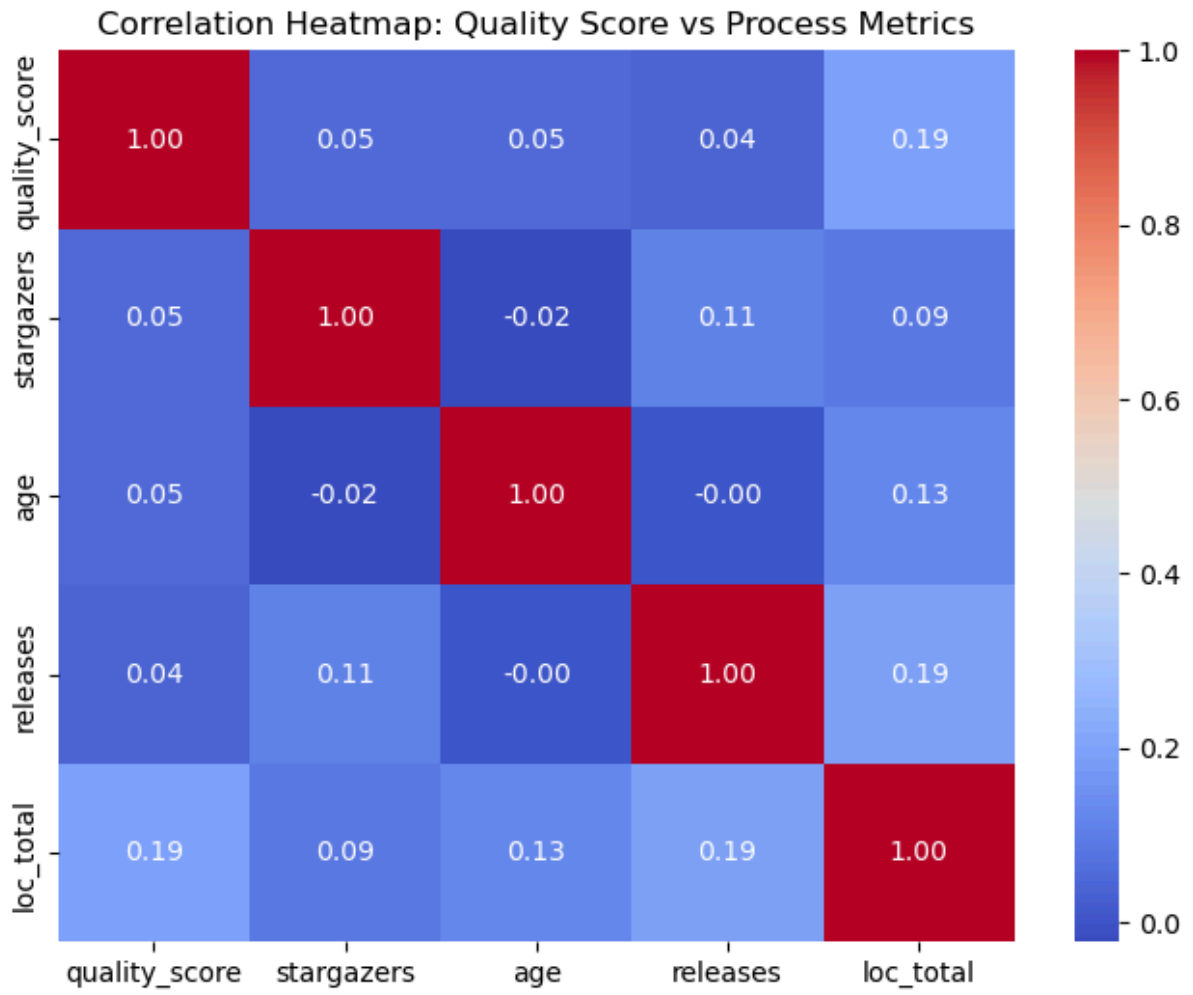
A terceira análise foi levando em consideração o número de releases contra o Quality Score. Assim como nos outros dados, podemos ver que o Quality Score independe do número de releases, mas de novo, repositórios com menos releases raramente possuem um Quality Score alto.



Na nossa quarta análise, comparamos o número de linhas dos repositórios com o Quality Score. Percebemos aqui uma leve tendência de que quanto mais linhas de código um repositório possui, maior é o Quality Score.


loc_total vs Quality Score
Quality Score
loc_total
0
100
200
300
400
500
10^1
10^2
10^3
10^4
10^5
10^6


Distribution of Combined Quality Score
Frequency
Quality Score
0
100
200
300
400
500
600
700



4. Interpretação dos Resultados

Os resultados estatísticos obtidos foram interpretados à luz das hipóteses formuladas inicialmente, buscando confirmar ou refutar as hipóteses informais a partir das características observadas nos dados.

R1 : Repositórios mais populares não necessariamente apresentam melhores métricas de qualidade. Inicialmente pensamos que existiria uma tendência de que repositórios mais populares apresentassem melhores métricas de qualidade.

R2: Não necessariamente repositórios mais antigos possuem métricas de qualidade melhores. Inicialmente pensamos que existiria uma tendência de que

quanto mais antigo fosse o repositório, melhores seriam suas métricas de qualidade.

R3: Não necessariamente repositórios mais ativos possuem métricas de qualidade melhores. Inicialmente também pensamos que quanto mais ativo fosse um repositório, melhor seriam suas métricas de qualidade.

R4: Quanto mais linhas de código um repositório possui, maior é a tendência de suas métricas de qualidade serem melhores. Essa resposta foi oposta a nossa hipótese inicial, de que quanto maior fosse um código, pior seriam suas métricas de qualidade.