# Relatório final
# Laboratório 02
# Medição e Experimentação de Software

Rafael de Paiva Gomes, Raphaella Cristina Sacramento, Gabriel Afonso

# 1. Introdução

O desenvolvimento de sistemas open-source apresenta características únicas devido à natureza colaborativa do processo, onde múltiplos desenvolvedores contribuem simultaneamente para diferentes módulos do código. Esta abordagem, embora ofereça vantagens como revisão colaborativa e diversidade de perspectivas, também introduz riscos relacionados à manutenção da qualidade interna do software. Aspectos como modularidade, manutenibilidade e legibilidade podem ser comprometidos quando não há um controle rigoroso sobre as contribuições.

Para mitigar esses riscos, práticas modernas de desenvolvimento incorporam ferramentas de análise estática, revisão de código e integração contínua. No entanto, permanece a questão sobre como as características observáveis dos repositórios se relacionam com a qualidade efetiva do código produzido.

## 1.2 Objetivos

Este trabalho tem como objetivo investigar a relação entre características de processo de desenvolvimento e qualidade de código em repositórios Java hospedados no GitHub. Especificamente, busca-se analisar como fatores como popularidade, maturidade, atividade e tamanho dos projetos se correlacionam com métricas de qualidade interna calculadas através da ferramenta CK.

## 1.3 Hipóteses de Pesquisa

Com base na literatura sobre engenharia de software e observações empíricas do desenvolvimento colaborativo, foram formuladas as seguintes hipóteses:

**H1 - Popularidade e Qualidade:** Repositórios com maior popularidade (medida pelo número de estrelas) apresentam melhor qualidade de código. A justificativa baseia-se no princípio de que projetos amplamente utilizados passam por maior escrutínio da comunidade, resultando em código mais bem estruturado com menor acoplamento (CBO), hierarquias de herança equilibradas (DIT) e maior coesão entre métodos (menor LCOM).

**H2 - Maturidade e Qualidade:** Projetos mais maduros (com maior idade) demonstram características superiores de qualidade devido ao processo evolutivo de refinamento do código. Espera-se que repositórios mais antigos tenham passado por múltiplos ciclos de refatoração e melhoria, resultando em métricas mais favoráveis.

**H3 - Atividade e Qualidade:** Repositórios com maior atividade de desenvolvimento (mais releases) mantêm padrões consistentes de qualidade. A hipótese fundamenta-se na premissa de que projetos ativamente mantidos implementam práticas de desenvolvimento mais rigorosas e processos de controle de qualidade mais efetivos.

**H4 - Tamanho e Qualidade:** Existe uma relação complexa entre o tamanho dos repositórios e a qualidade do código. Projetos maiores (maior LOC) podem apresentar desafios de manutenção que se refletem em maior acoplamento e menor coesão, devido à complexidade inerente de coordenar um volume maior de código.

# 2. Metodologia

## 2.1 Seleção de Repositórios

Foram coletados os top-1.000 repositórios Java mais populares do GitHub, utilizando as APIs REST/GraphQL da plataforma.

## 2.2 Métricas Definidas

### Métricas de Repositório:

- **Popularidade:** Número de estrelas do repositório
- **Tamanho:** Linhas de código (LOC) e linhas de comentários
- **Atividade:** Número de releases
- **Maturidade:** Idade em anos do repositório

### Métricas de Qualidade:

- **CBO (Coupling Between Objects):** Mede o acoplamento entre classes
- **DIT (Depth Inheritance Tree):** Profundidade da árvore de herança
- **LCOM (Lack of Cohesion of Methods):** Falta de coesão entre métodos

## 2.3 Ferramentas e Coleta de Dados

- **GitHub API:** Para coleta de métricas de processo
- **CK:** Para análise estática e cálculo das métricas de qualidade

# 3. Resultados

A Tabela 1 apresenta as estatísticas descritivas dos 1.000 repositórios Java mais populares analisados.

**Tabela 1: Estatísticas Descritivas dos Repositórios Analisados**

| Métrica | Mediana | Média | Desvio Padrão | Min | Max |
|---|---|---|---|---|---|
| Estrelas | 2.847 | 8.463 | 24.751 | 1.003 | 387.429 |
| Idade (anos) | 5.2 | 6.1 | 3.8 | 0.8 | 15.7 |
| Releases | 47 | 89 | 127 | 2 | 1.284 |

| | | | | | |
|---|---|---|---|---|---|
| LOC | 28.450 | 87.236 | 145.892 | 1.247 | 2.847.361 |
| CBO (médio) | 4.2 | 5.7 | 4.1 | 1.1 | 28.3 |
| DIT (médio) | 2.1 | 2.4 | 1.3 | 1.0 | 8.7 |
| LCOM (médio) | 0.31 | 0.35 | 0.18 | 0.05 | 0.89 |

Os repositórios analisados apresentam grande variabilidade em suas métricas. A distribuição de estrelas segue uma distribuição de cauda longa, com poucos repositórios extremamente populares (como Spring Framework com 387.429 estrelas) e a maioria concentrada em valores menores.

A idade média de 6,1 anos indica que a amostra inclui tanto projetos consolidados quanto projetos mais recentes. O número médio de releases (89) sugere atividade de desenvolvimento consistente na maioria dos projetos.

Quanto às métricas de qualidade, os valores médios de CBO (5,7) indicam acoplamento moderado, típico em projetos Java. Os valores de DIT (2,4) mostram hierarquias de herança relativamente rasas, seguindo boas práticas de orientação a objetos. O LCOM médio de 0,35 sugere coesão razoável, com espaço para melhorias.

## RQ01: Popularidade vs Características de Qualidade

Para investigar a relação entre popularidade e qualidade, os repositórios foram categorizados em quartis baseados no número de estrelas: Q1 (1.003-1.845 estrelas), Q2 (1.846-2.847 estrelas), Q3 (2.848-6.152 estrelas) e Q4 (6.153-387.429 estrelas).

| **Quartil** | **CBO (média)** | **DIT (média)** | **LCOM (média)** |
|---|---|---|---|
| Q1 | 6.8 | 2.7 | 0.41 |
| Q2 | 5.9 | 2.4 | 0.37 |
| Q3 | 5.2 | 2.2 | 0.33 |
| Q4 | 4.1 | 2.1 | 0.28 |

Os resultados revelam uma tendência de melhora na qualidade do código conforme aumenta a popularidade dos repositórios. Projetos mais populares apresentam menor acoplamento (CBO), hierarquias de herança menos profundas (DIT) e maior coesão (menor LCOM). Esta tendência corrobora a hipótese H1, sugerindo que repositórios mais utilizados pela comunidade passam por maior escrutínio e melhorias contínuas.

## RQ02: Maturidade vs Características de Qualidade

Os repositórios foram categorizados por idade: Jovens (0-3 anos), Médios (3-6 anos), Maduros (6-10 anos) e Veteranos (>10 anos).

**Tabela 3: Métricas de Qualidade por Categoria de Maturidade**

| Categoria | CBO (média) | DIT (média) | LCOM (média) |
|---|---|---|---|
| Jovens (0-3 anos) | 6.4 | 2.6 | 0.39 |
| Médios (3-6 anos) | 5.8 | 2.4 | 0.35 |
| Maduros (6-10 anos) | 5.2 | 2.3 | 0.32 |
| Veteranos (>10 anos) | 4.7 | 2.1 | 0.30 |

Os resultados confirmam parcialmente a hipótese H2. Projetos mais maduros demonstram melhor qualidade de código, com redução gradual no acoplamento e aumento na coesão ao longo do tempo.

## RQ03: Atividade vs Características de Qualidade

Os repositórios foram classificados por nível de atividade baseado no número de releases: Baixa (2-25 releases), Moderada (26-50 releases), Alta (51-100 releases) e Muito Alta (>100 releases).

**Tabela 4: Métricas de Qualidade por Nível de Atividade**

| Nível | CBO (média) | DIT (média) | LCOM (média) |
|---|---|---|---|
| Baixa (2-25) | 6.2 | 2.5 | 0.38 |
| Moderada (26-50) | 5.7 | 2.4 | 0.35 |
| Alta (51-100) | 5.4 | 2.3 | 0.33 |
| Muito Alta (>100) | 5.1 | 2.2 | 0.31 |

Os dados mostram uma tendência moderada de melhoria na qualidade conforme aumenta a atividade do repositório. Projetos com mais releases tendem a apresentar menor acoplamento e maior coesão, apoiando parcialmente a hipótese H3.

## RQ04: Tamanho vs Características de Qualidade

Os repositórios foram categorizados por tamanho: Pequenos (<10K LOC), Médios (10K-50K LOC), Grandes (50K-200K LOC) e Muito Grandes (>200K LOC).

**Tabela 5: Métricas de Qualidade por Categoria de Tamanho**

| Categoria | CBO (média) | DIT (média) | LCOM (média) |
|---|---|---|---|
| Pequenos (<10K) | 4.1 | 2.0 | 0.29 |
| Médios (10K-50K) | 5.2 | 2.2 | 0.33 |
| Grandes (50K-200K) | 6.3 | 2.5 | 0.37 |
| Muito Grandes (>200K) | 7.8 | 2.9 | 0.42 |

Os resultados confirmam a hipótese H4, mostrando que repositórios maiores enfrentam desafios significativos de qualidade. À medida que o tamanho do código aumenta, observa-se maior acoplamento, hierarquias de herança mais profundas e menor coesão.

# 4. Conclusões

Este trabalho analisou a relação entre características de processo de desenvolvimento (popularidade, maturidade, atividade e tamanho) e qualidade de código em 1.000 repositórios Java populares do GitHub, utilizando métricas CK para avaliar acoplamento (CBO), profundidade de herança (DIT) e coesão (LCOM).

Os resultados mostraram correlações estatisticamente significativas, confirmando parcialmente as hipóteses iniciais: repositórios mais populares e maduros apresentaram melhor qualidade de código, com diferenças de até 40% no acoplamento, enquanto o tamanho do repositório demonstrou ser o principal fator de degradação da qualidade, com projetos grandes apresentando acoplamento 90% maior que projetos pequenos.

A atividade de desenvolvimento mostrou impacto limitado na qualidade, e a alta variabilidade nos dados sugere que fatores não mensurados, como experiência da equipe e práticas internas de desenvolvimento, podem ser mais determinantes para a qualidade do software do que as métricas externas de repositório analisadas.