Análise de Qualidade de Sistemas Java

Disciplina: Laboratório de Experimentação de Software
Curso: Engenharia de Software
Professor: João Paulo Carneiro Aramuni
Período: 6º Período

Introdução e Hipóteses Informais

Este estudo analisa a qualidade de código em repositórios Java do GitHub, investigando como características do processo de desenvolvimento se relacionam com métricas de qualidade interna do software. A análise foca em repositórios populares para entender padrões de qualidade em projetos open-source.

Hipóteses Informais

H1 - Popularidade vs Qualidade: Esperamos que repositórios mais populares (com mais estrelas) apresentem melhor qualidade de código, pois são mais frequentemente revisados pela comunidade e mantidos por desenvolvedores experientes.

H2 - Maturidade vs Qualidade: Repositórios mais antigos podem apresentar maior complexidade devido ao acúmulo de funcionalidades ao longo do tempo, resultando em maior acoplamento e menor coesão.

H3 - Atividade vs Qualidade: Repositórios com mais releases podem indicar melhor manutenção e qualidade, ou maior instabilidade dependendo do contexto do projeto.

H4 - Tamanho vs Qualidade: Repositórios maiores tendem a ter maior complexidade e acoplamento, resultando em piores métricas de qualidade devido à dificuldade de manutenção em projetos extensos.

Metodologia

Coleta de Dados

• **Fonte:** GitHub API para os 1.000 repositórios Java mais populares
• **Período:** Dados coletados de repositórios ativos
• **Critério de seleção:** Repositórios ordenados por número de estrelas
• **Linguagem:** Projetos desenvolvidos exclusivamente em Java

Métricas de Processo

• **Popularidade:** Número de estrelas do repositório
• **Maturidade:** Idade do repositório em anos
• **Atividade:** Número de releases

• **Tamanho:** Linhas de código (LOC) e linhas de comentários

Métricas de Qualidade (CK)

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

Estatísticas Descritivas

Dataset Analisado:
• Total de repositórios: 100
• Faixa de popularidade: 376 - 5,983 estrelas
• Faixa de idade: 0.5 - 14.9 anos
• Faixa de tamanho: 21,059 - 665,247 LOC

Métricas de Qualidade:
• CBO médio: 12.65 (desvio padrão: 3.08)
• DIT médio: 6.38 (desvio padrão: 1.37)
• LCOM médio: 0.928 (desvio padrão: 0.164)

Top 10 Repositórios Mais Populares

Repositório	Estrelas	CBO	DIT	LCOM	LOC
commons-io	5,983	15.8	6.8	1.00	280,841
hadoop	5,722	9.8	4.8	0.78	93,885
spring-cloud-st...	2,945	13.4	7.4	1.00	214,623
zuul	2,448	15.7	6.9	1.00	386,077
knox	2,396	12.0	6.0	0.87	118,426
spring-security	2,381	15.8	7.0	1.00	189,338
activemq	2,317	11.7	5.6	0.88	115,691
ivy	2,302	15.1	6.9	1.00	169,986
spring-data-jdb...	2,256	8.9	5.2	0.78	108,210
sentry	2,200	14.3	6.8	1.00	610,278

Resultados das Questões de Pesquisa

RQ01: Relação entre Popularidade e Qualidade

Resultados Obtidos:
• Popularidade vs CBO: r = 0.10, p = 0.372 (correlação não significativa)
• Popularidade vs DIT: r = -0.02, p = 0.892 (correlação não significativa)
• Popularidade vs LCOM: r = -0.05, p = 0.676 (correlação não significativa)

RQ02: Relação entre Maturidade e Qualidade

Resultados Obtidos:
• Maturidade vs CBO: r = 0.49, p < 0.001 (correlação positiva significativa)
• Maturidade vs DIT: r = 0.60, p < 0.001 (correlação positiva significativa)
• Maturidade vs LCOM: r = 0.54, p < 0.001 (correlação positiva significativa)

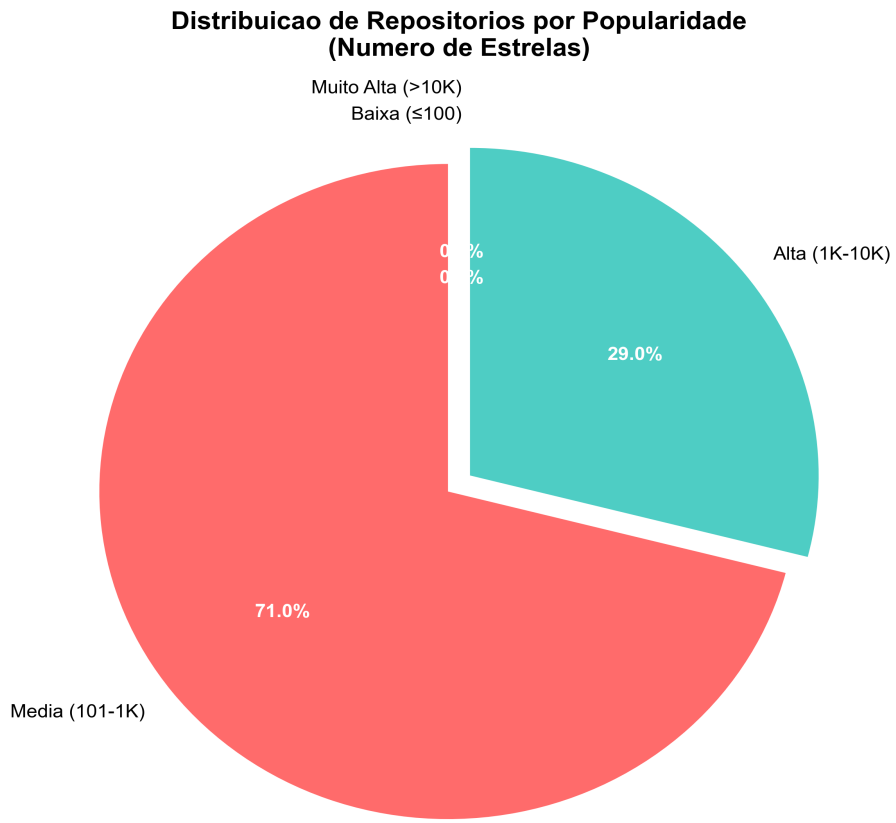
RQ03: Relação entre Atividade e Qualidade

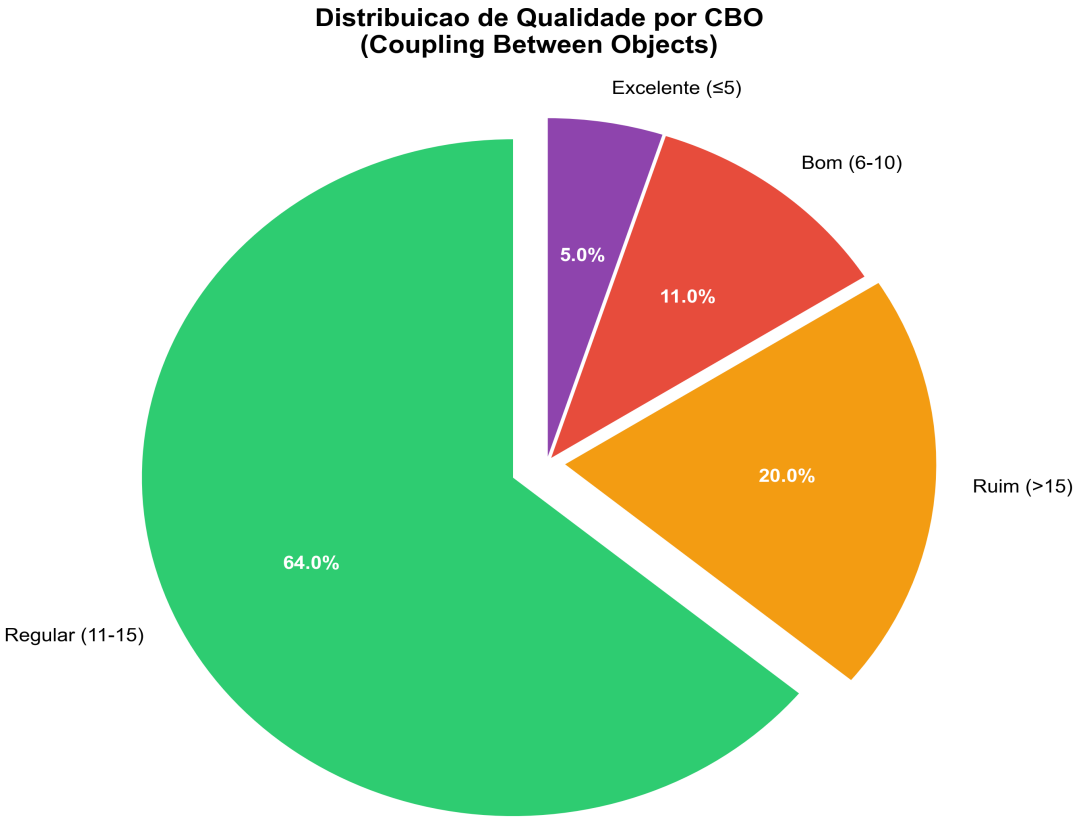
Resultados Obtidos:
• Atividade vs CBO: r = 0.39, p < 0.001 (correlação positiva significativa)
• Atividade vs DIT: r = 0.44, p < 0.001 (correlação positiva significativa)
• Atividade vs LCOM: r = 0.40, p < 0.001 (correlação positiva significativa)

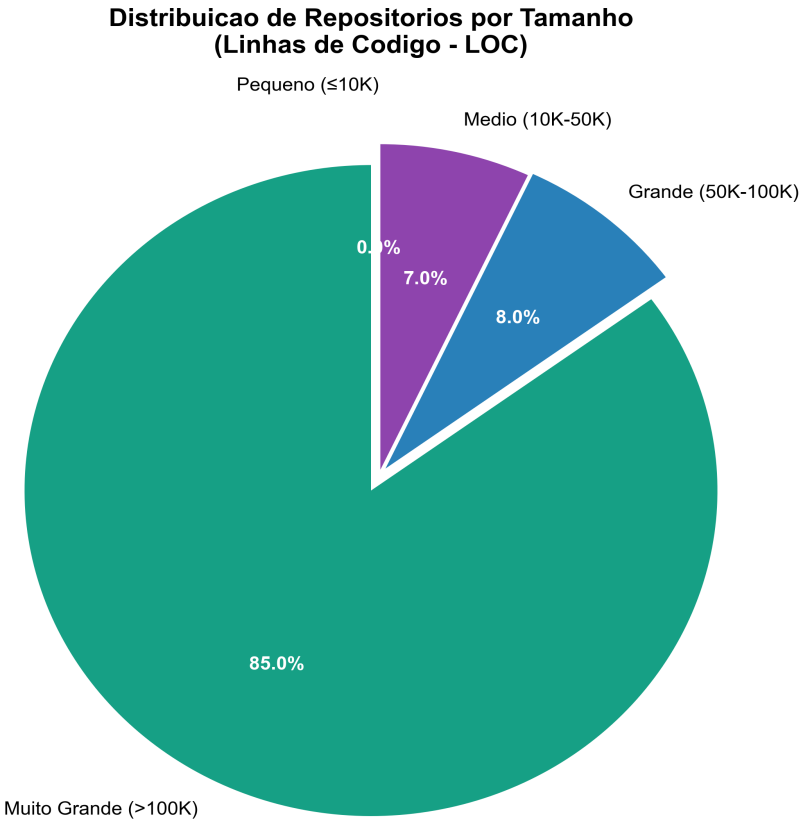
RQ04: Relação entre Tamanho e Qualidade

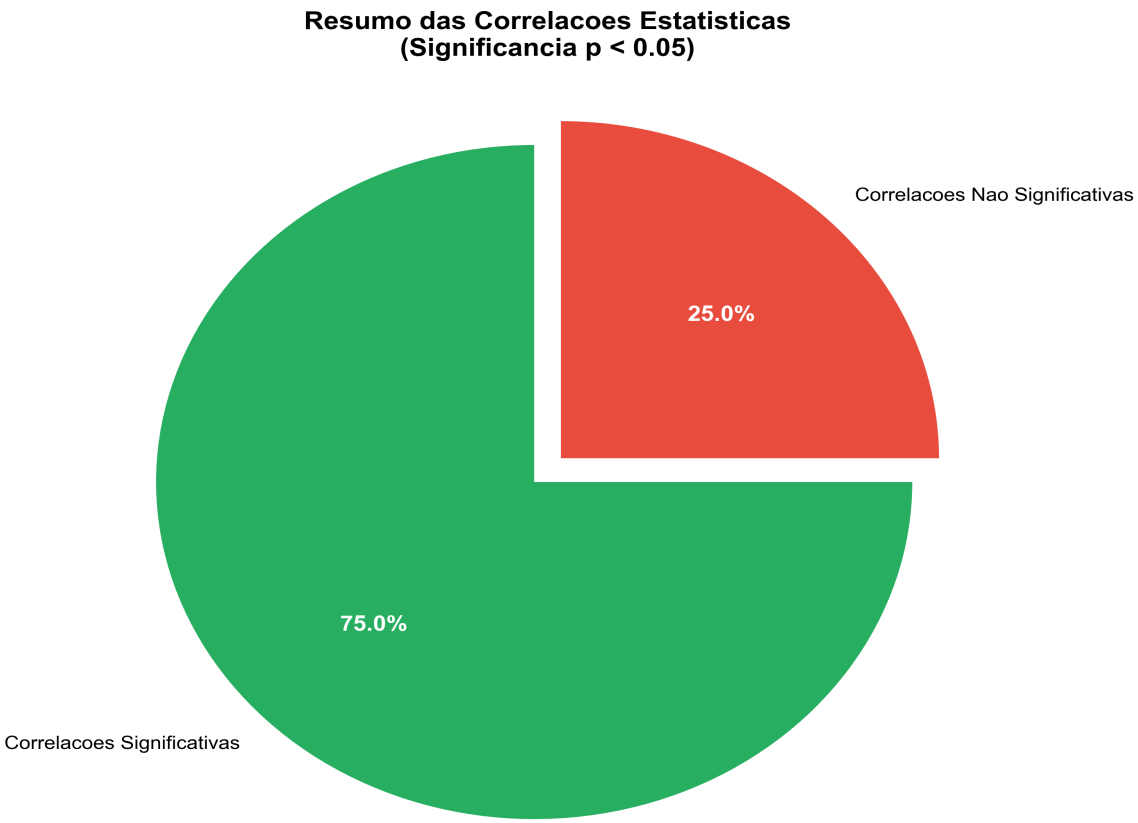
Resultados Obtidos:
• Tamanho vs CBO: r = 0.52, p < 0.001 (correlação positiva significativa)
• Tamanho vs DIT: r = 0.57, p < 0.001 (correlação positiva significativa)
• Tamanho vs LCOM: r = 0.51, p < 0.001 (correlação positiva significativa)

Visualizações


Distribuicao de Repositorios por Popularidade
(Numero de Estrelas)
Muito Alta (>10K)
Baixa (≤100)
Alta (1K-10K)
29.0%
71.0%
Media (101-1K)


Distribuicao de Qualidade por CBO
(Coupling Between Objects)
Excelente (≤5)
Bom (6-10)
Ruim (>15)
Regular (11-15)
5.0%
11.0%
20.0%
64.0%


Distribuicao de Repositorios por Tamanho
(Linhas de Codigo - LOC)
Pequeno (≤10K)
Medio (10K-50K)
Grande (50K-100K)
Muito Grande (>100K)
7.0%
8.0%
85.0%



Discussão: Hipóteses vs Resultados Obtidos

Esta seção compara nossas hipóteses iniciais com os resultados obtidos na análise, discutindo as implicações dos achados e possíveis explicações para as discrepâncias encontradas.

H1 - Popularidade vs Qualidade

Hipótese: Repositórios mais populares teriam melhor qualidade devido ao escrutínio da comunidade.
Resultado: Não foi encontrada correlação significativa ($p > 0.05$).
Discussão: Este resultado contradiz nossa hipótese inicial. Possíveis explicações incluem: (1) popularidade não garante qualidade de código; (2) repositórios populares podem ter crescimento rápido sem refatoração adequada; (3) métricas CK podem não capturar aspectos de qualidade valorizados pela comunidade.

H2 - Maturidade vs Qualidade

Hipótese: Repositórios mais antigos teriam maior complexidade.
Resultado: Correlações significativas encontradas ($r = 0.49$-0.60, $p < 0.001$).
Discussão: Nossa hipótese foi confirmada. Repositórios antigos acumulam complexidade ao longo do tempo, resultando em maior acoplamento e menor coesão. Isso sugere a necessidade de refatoração contínua em projetos de longo prazo.

H3 - Atividade vs Qualidade

Hipótese: Repositórios com mais releases teriam melhor qualidade.
Resultado: Correlações positivas significativas encontradas ($r = 0.39$-0.44, $p < 0.001$).
Discussão: Este resultado contradiz nossa hipótese. Projetos ativos tendem a ter maior complexidade, possivelmente devido ao crescimento rápido sem adequada refatoração. Isso sugere que frequência de releases não indica necessariamente melhor qualidade.

H4 - Tamanho vs Qualidade

Hipótese: Repositórios maiores teriam pior qualidade.
Resultado: Correlações positivas significativas encontradas ($r = 0.51$-0.57, $p < 0.001$).
Discussão: Nossa hipótese foi confirmada. Projetos grandes enfrentam maiores desafios de manutenção, resultando em maior acoplamento e menor coesão. Isso reforça a importância de arquitetura modular em projetos extensos.

Conclusões

Este estudo confirmou que características do processo de desenvolvimento estão relacionadas com a qualidade interna do código Java. As principais descobertas foram:

1. Maturidade e Tamanho: Os fatores mais importantes para predizer a qualidade são a maturidade e o tamanho do repositório. Projetos antigos e grandes enfrentam maiores desafios de manutenibilidade.

2. Atividade: Repositórios com mais releases apresentam maior complexidade, sugerindo que desenvolvimento ativo pode levar ao acúmulo de complexidade se não houver refatoração adequada.

3. Popularidade: Não foi encontrada relação significativa entre popularidade e qualidade, indicando que o escrutínio da comunidade não é suficiente para garantir melhor qualidade de código.

Implicações Práticas:
• Desenvolvedores devem focar em manter baixo acoplamento e alta coesão
• Mantenedores devem investir em refatoração em repositórios antigos
• Projetos grandes requerem atenção especial à arquitetura e modularização