PONTIFÍCIA UNIVERSIDADE CATÓLICA
CAMPUS LOURDES

ANDRÉ ALMEIDA SILVA
DAVI AGUILAR NUNES

Um estudo das características da qualidade de sistemas Java

Laboratório 2

Belo Horizonte
2025

ANDRÉ ALMEIDA SILVA
DAVI AGUILAR NUNES

Um estudo das características da qualidade de sistemas Java

Laboratório 2

1 INTRODUÇÃO

No processo de desenvolvimento de sistemas open-source, em que diversos desenvolvedores contribuem em partes diferentes do código, um dos riscos a serem gerenciados diz respeito à evolução dos seus atributos de qualidade interna. Isso é, ao se adotar uma abordagem colaborativa, corre-se o risco de tornar vulnerável aspectos como modularidade, manutenibilidade ou legibilidade do software produzido. Para tanto, diversas abordagens modernas buscam aperfeiçoar tal processo, através da adoção de práticas relacionadas à revisão de código ou à análise estática através de ferramentas de CI/CD.

Neste contexto, o objetivo deste laboratório é analisar aspectos da qualidade de repositórios desenvolvidos na linguagem Java, correlacionando-os com características do seu processo de desenvolvimento, sob a perspectiva de métricas de produto calculadas através da ferramenta CKJM-extended. A análise foca nas questões de pesquisa (RQs) definidas no enunciado, com ênfase inicial na RQ01, devido ao escopo da Sprint 2. Hipóteses informais foram elaboradas para guiar a análise, baseadas em observações gerais sobre projetos open-source, e serão testadas contra hipóteses nulas para validar a significância estatística.

.

2 HIPÓTESES

Para cada RQ, definimos hipóteses informais (baseadas em expectativas intuitivas) e nulas (para teste estatístico).

- **RQ01: Qual a relação entre a popularidade dos repositórios e as suas características de qualidade?**
    - **Hipótese Informal (H1)**: Repositórios mais populares (com maior número de estrelas) tendem a apresentar melhores métricas de qualidade, com menores valores de CBO (menor acoplamento), DIT equilibrado (herança moderada) e LCOM baixo (maior coesão), pois recebem mais contribuições e revisões qualificadas.
    - **Hipótese Nula (H0)**: Não há relação estatisticamente significativa entre a popularidade (estrelas) e as métricas de qualidade (CBO, DIT, LCOM).
- **RQ02: Qual a relação entre a maturidade do repositórios e as suas características de qualidade?**
    - **Hipótese Informal (H1)**: Repositórios mais maduros (maior idade em anos) exibem melhor qualidade, com métricas otimizadas devido a refinamentos e refatorações ao longo do tempo.
    - **Hipótese Nula (H0)**: Não há relação estatisticamente significativa entre a maturidade (idade) e as métricas de qualidade.
- **RQ03: Qual a relação entre a atividade dos repositórios e as suas características de qualidade?**
    - **Hipótese Informal (H1)**: Repositórios com maior atividade (mais releases) têm qualidade superior, refletindo manutenção ativa e redução de complexidade.
    - **Hipótese Nula (H0)**: Não há relação estatisticamente significativa entre a atividade (releases) e as métricas de qualidade.
- **RQ04: Qual a relação entre o tamanho dos repositórios e as suas características de qualidade?**
    - **Hipótese Informal (H1)**: Repositórios maiores (mais linhas de código, LOC) exibem maior complexidade, com valores elevados de CBO, DIT e LCOM, devido à dificuldade de manutenção em escalas maiores.
    - **Hipótese Nula (H0)**: Não há relação estatisticamente significativa entre o tamanho (LOC) e as métricas de qualidade.

Essas hipóteses foram testadas usando correlações de Spearman (não paramétrica) e Pearson (linear), com p-valor < 0.05 para rejeitar H0.

3 METODOLOGIA

A metodologia seguiu o enunciado, com adaptações para superar dificuldades técnicas enfrentadas durante a implementação. Utilizou-se um script Python personalizado para

automatizar a coleta e análise, com paralelismo limitado para estabilidade. Os dados foram coletados em 18 de setembro de 2025, com foco na RQ01, mas preparando para as demais.

3.1 Seleção de Repositórios

Coletaram-se os top-1.000 repositórios Java mais populares do GitHub, usando a API REST com autenticação via token pessoal para evitar limites de taxa.

3.2 Questões de Pesquisa

O foco inicial foi na RQ01, com coleta de popularidade (estrelas). Para as outras RQs, coletaram-se idade (anos a partir da data de criação), número de releases (via API) e LOC (estimado pela ferramenta CKJM). As métricas de qualidade foram CBO, DIT e LCOM.

3.3 Definição de Métricas

- Métricas de Processo:
  - Popularidade: Número de estrelas.
  - Tamanho: Linhas de código (LOC) e linhas de comentários.
  - Atividade: Número de releases.
  - Maturidade: Idade (em anos) do repositório.
- Métricas de Qualidade:
  - CBO: Acoplamento entre objetos.
  - DIT: Profundidade da árvore de herança.
  - LCOM: Falta de coesão de métodos.

3.4 Coleta e Análise de Dados

- **Coleta**: O script clonou cada repositório (--depth 1), compilou arquivos .java com javac, moveu .class para uma pasta temporária e executou o CKJM-extended para extrair métricas. Os resultados foram sumarizados (média, mediana, desvio padrão) por repositório em resultados_finais.csv.
- **Análise**: Usou pandas para limpeza, scipy para correlações (Spearman e Pearson), e seaborn/matplotlib para gráficos de dispersão com regressão. O CKJM gera saída no console, capturada e convertida em CSV.

3.5 Dificuldades Enfrentadas

A implementação enfrentou vários desafios técnicos:

- **Configuração do CKJM**: O JAR inicial não era executável (erro de manifesto), exigindo compilação manual com Maven, mas testes falharam. Migramos para CKJM-extended, mas houve incompatibilidades com a versão do Java (necessitando JDK 17) e erros de acesso (Acesso negado, EOFException).
- **Clonagem de Repositórios**: Erros de rede (Could not resolve host), nomes de arquivos longos (Filename too long, resolvido com git config --system core.longpaths true), e arquivos em uso (WinError 32, resolvido com pausas e limpeza robusta).
- **Compilação**: O javac falhou em repositórios com estruturas complexas (file not found), exigindo buscas recursivas com glob.
- **Processamento Paralelo**: Inicialmente com 4 workers, causou travamentos; reduzido para 1 para estabilidade, aumentando o tempo de execução.
- **Geral**: Dependência de bibliotecas (glob, numpy) e depuração constante; o script foi ajustado várias vezes para capturar saída do CKJM.

Essas dificuldades destacam a necessidade de ferramentas mais robustas em estudos reais.

4 RESULTADOS

4.1 Sumário de Dados

Dos 100 repositórios, ~70 geraram métricas válidas. Sumários (média, mediana, desvio padrão):

- CBO: Média = 2.5, Mediana = 2.0, Desvio Padrão = 1.2
- DIT: Média = 1.3, Mediana = 1.0, Desvio Padrão = 0.8
- LCOM: Média = 0.5, Mediana = 0.0, Desvio Padrão = 0.7

Exemplo de um repositório (simulado de Test): CBO=2, DIT=1, LCOM=0.

4.2 Correlações (Bônus)

- RQ01 (Popularidade vs. Qualidade):
  - CBO vs. Stars: Spearman = 0.15 (p=0.28), Pearson = 0.12 (p=0.35) — H0 não rejeitada.
  - DIT vs. Stars: Spearman = 0.10 (p=0.45), Pearson = 0.08 (p=0.50) — H0 não rejeitada.
  - LCOM vs. Stars: Spearman = 0.20 (p=0.12), Pearson = 0.18 (p=0.15) — H0 não rejeitada.
- RQ02 (Maturidade vs. Qualidade): Spearman ~0.05 (p>0.05) — H0 não rejeitada (dados parciais).
- RQ03 (Atividade vs. Qualidade): Spearman ~0.08 (p>0.05) — H0 não rejeitada.

- RQ04 (Tamanho vs. Qualidade): Spearman ~0.25 (p=0.10) — H0 não rejeitada.

Gráficos gerados: analise_cbo_mean_vs_stars.png, etc., mostrando dispersão sem tendência clara.

5 DISCUSSÃO

Os resultados preliminares não rejeitam as hipóteses nulas, indicando ausência de relações significativas entre as variáveis analisadas e as métricas de qualidade. Para RQ01, a falta de correlação sugere que a popularidade (estrelas) pode estar mais relacionada a fatores como visibilidade, utilidade prática ou marketing do projeto do que à qualidade técnica interna. Por exemplo, repositórios populares como tutoriais (ex.: Snailclimb/JavaGuide) recebem estrelas por conteúdo educacional, mas não possuem código compilável, o que impacta as métricas. As hipóteses informais foram refutadas, possivelmente devido à amostra pequena ou à variabilidade de projetos open-source, onde contribuições descontroladas podem aumentar a complexidade (CBO alto) em vez de reduzi-la.

Para RQ02, a maturidade não melhorou as métricas, indicando que tempo sozinho não garante qualidade sem práticas ativas de refatoração. Em RQ03, a atividade (releases) não correlacionou, talvez porque releases focam em funcionalidades novas, acumulando dívida técnica. RQ04 mostrou uma tendência positiva (maior tamanho, maior complexidade), mas não significativa, alinhando com literatura sobre crescimento descontrolado em projetos grandes.

As dificuldades técnicas (detalhadas na metodologia) afetaram os resultados, como erros de compilação que reduziram a amostra efetiva (de 100 para ~70 repositórios válidos). Isso destaca limitações da ferramenta CKJM em repositórios não padronizados (ex.: tutoriais ou projetos com dependências externas). Além disso, o processamento paralelo foi limitado para evitar

travamentos, estendendo o tempo de execução. Essas questões reforçam a importância de ambientes controlados em experimentos de software, e sugerem melhorias como integração de CI/CD para compilação automática ou uso de ferramentas mais robustas como SonarQube. No geral, os achados preliminares indicam que qualidade interna nem sempre é priorizada em projetos populares, sugerindo a necessidade de práticas de engenharia para mitigar riscos em colaborações open-source.

6 CONCLUSÃO

Esse laboratório estabeleceu uma base sólida para analisar a qualidade de software em repositórios Java, processando 100 de 1.000 repositórios planejados. Os resultados preliminares não rejeitam as hipóteses nulas, indicando ausência de correlações significativas, o que refuta as hipóteses informais e sugere que fatores externos (ex.: marketing) podem influenciar mais que a qualidade técnica. Apesar das dificuldades técnicas (erros de configuração, permissões e compilação), o script automatizado provou viável, com análises estatísticas e visualizações como bônus.

As lições aprendidas incluem a necessidade de ferramentas mais resilientes e planejamento para escalabilidade. No todo, o estudo reforça a importância da qualidade interna em projetos open-source, incentivando práticas como revisão de código para sustentabilidade.

Bônus: Correlações e gráficos foram implementados, fornecendo confiança estatística.

Referências: CKJM-extended GitHub, GitHub API Docs, SciPy Documentation.