Relatório Final - Laboratório 02

Eric Rodrigues Diniz, Ian Nascimento, Pablo Guilherme Benevenuto

INTRODUÇÃO E HIPÓTESES

Este estudo tem como objetivo analisar aspectos relacionados à qualidade de repositórios open-source desenvolvidos em Java, especialmente no que se refere à modularidade, manutenibilidade e legibilidade. A análise será feita a partir de métricas de produto extraídas por meio da ferramenta CK. Antes do início da pesquisa, foram definidas algumas questões e, com base nelas, elaboradas as seguintes hipóteses:

- **RQ 01**: Acreditamos que os repositórios mais populares tendem a apresentar maior qualidade, pois é improvável que a comunidade apoie projetos que não atendam a um padrão mínimo de qualidade.

- **RQ 02**: Espera-se que a maturidade do repositório (sua idade) esteja positivamente correlacionada com a sua qualidade, já que projetos mais antigos tiveram mais tempo para corrigir falhas e aprimorar-se.

- **RQ 03**: No que diz respeito à atividade do repositório, nossa suposição é que repositórios com maior frequência de atualizações (releases) apresentem melhor qualidade, considerando que atualizações constantes indicam esforço contínuo de melhoria.

- **RQ 04**: Quanto ao tamanho, espera-se que repositórios de maior qualidade sejam mais concisos, com menos linhas de código, o que pode facilitar a manutenção e reduzir a complexidade do sistema.

METODOLOGIA

Para responder às questões propostas na pesquisa, foi desenvolvido um script em Python responsável pela coleta de dados por meio da API GraphQL do GitHub. A consulta foi elaborada para recuperar os 1000 repositórios escritos em Java com o maior número de estrelas na plataforma.

É importante destacar que a extração das métricas de qualidade foi realizada utilizando a ferramenta CK. Vale notar que essa ferramenta é compatível apenas com repositórios que utilizam Maven e que estejam configurados para versões do Java até a 11.

Os dados extraídos de cada repositório foram os seguintes:

- Nome do repositório

- Número de estrelas

- Linhas de código (kLOC)

- Número de releases

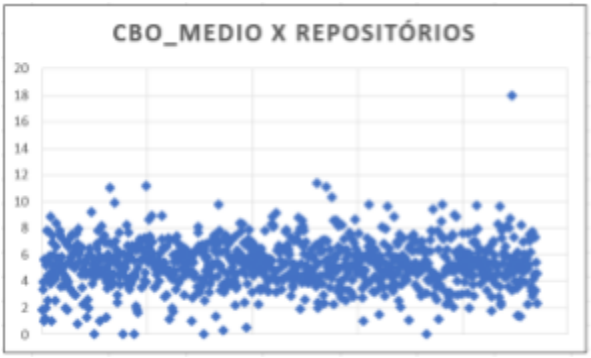
- Idade (em anos)
- CBO (Coupling Between Objects)
- DIT (Depth of Inheritance Tree)

Essas informações foram processadas e consolidadas em um arquivo CSV. A partir desse arquivo, a análise foi realizada com o auxílio da linguagem Python, utilizando bibliotecas como **Pandas** e **Matplotlib** para tratamento dos dados e geração de gráficos que apoiam a interpretação dos resultados.

RESULTADOS OBTIDOS

Após a análise dos dados e a geração dos gráficos, foi possível visualizar as métricas associadas a cada uma das questões de pesquisa. A seguir, são apresentadas as interpretações dos resultados, em relação às hipóteses estabelecidas na Seção 1.

Antes, porém, são exibidos dois gráficos que ilustram a distribuição geral das métricas de qualidade entre todos os repositórios analisados.

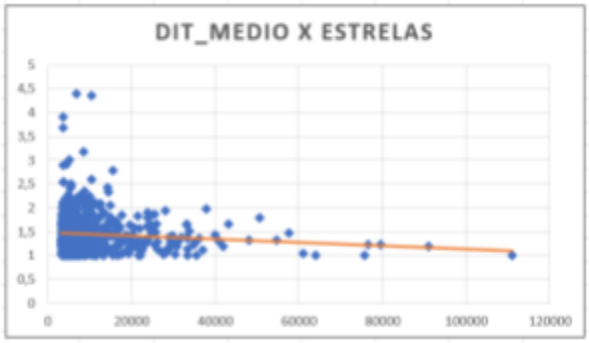


O CBO m´edio dos repositórios foi de ≈ 5, 27.

O DIT m´edio dos repositórios foi de ≈ 1, 46.

RQ 01:

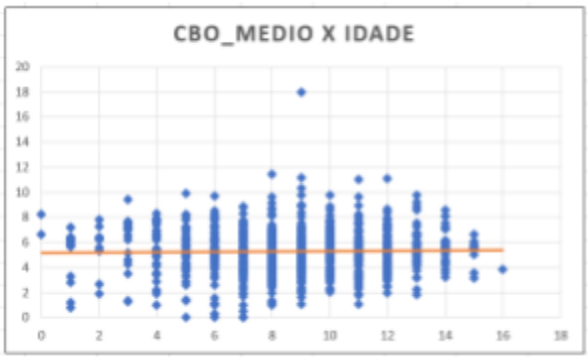
Em relação à popularidade (número de estrelas), não foi identificada uma correlação direta com as métricas de qualidade analisadas. De modo geral, observou-se que repositórios com menos de 10 mil estrelas apresentaram valores mais elevados nas métricas CBO e DIT, indicando maior acoplamento entre classes e maior profundidade na hierarquia de herança.

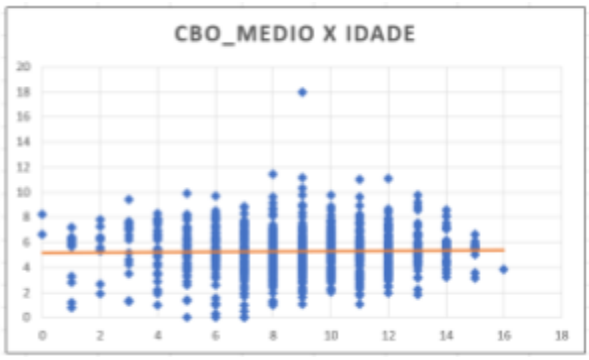


RQ 02:

No que diz respeito à idade do repositório, observou-se que ela não exerce uma influência significativa sobre sua qualidade. Foi possível identificar repositórios mais recentes que apresentam níveis de qualidade semelhantes aos de projetos mais antigos.

Dessa forma, a hipótese de que repositórios mais antigos tendem a ter maior qualidade não se confirmou completamente. Isso pode ser explicado pelo fato de que muitos repositórios novos já são desenvolvidos seguindo boas práticas e padrões de qualidade previamente estabelecidos.

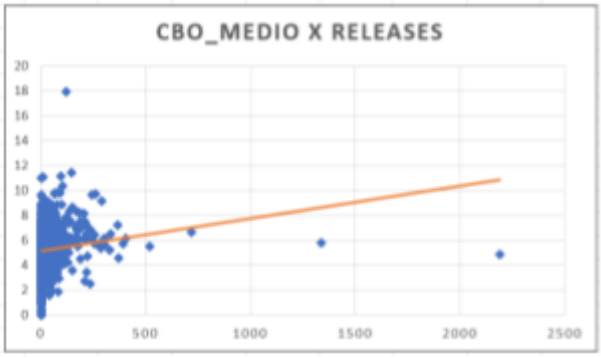


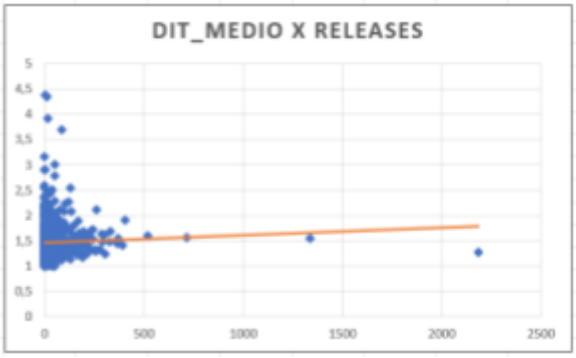


RQ 03:

No caso do número de releases, a hipótese inicialmente formulada se mostrou incorreta. Os dados indicam que a qualidade dos repositórios se mantém estável, mesmo entre aqueles que possuem até 200 releases.

Acreditamos que esse resultado esteja relacionado ao que foi observado na hipótese anterior: muitos repositórios já são criados seguindo padrões de qualidade bem definidos desde o início. Isso implica em menos erros que comprometam a qualidade do código, o que, por sua vez, reduz a necessidade de lançamentos frequentes para correções.

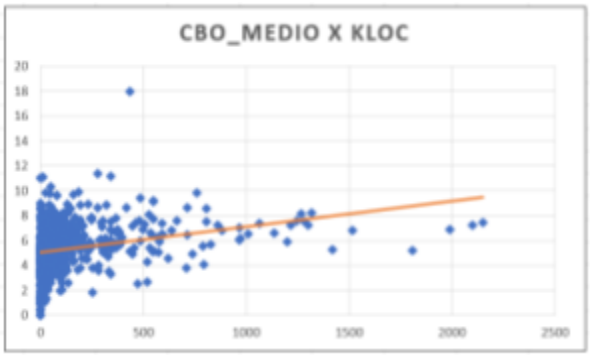


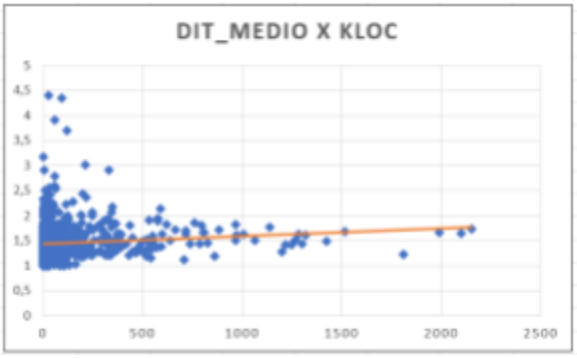


RQ 04:

No que se refere ao fator linhas de código (consideradas em milhar - kLOC), a hipótese foi confirmada. Os resultados mostraram que repositórios com menor quantidade de código tendem a manter uma melhor qualidade, possivelmente por serem menos complexos e mais fáceis de manter.

No entanto, também foi possível identificar casos de repositórios com mais de 2000 kLOC (ou 2 milhões de linhas) que conseguem manter níveis de qualidade comparáveis aos demais, indicando que um bom controle de qualidade pode ser mantido mesmo em sistemas de grande porte.





CONCLUSÃO

Como conclusão deste estudo, observa-se que a qualidade passou a ser um aspecto essencial no desenvolvimento de softwares, sendo considerada desde as etapas iniciais do projeto, muitas vezes antes mesmo do início efetivo da codificação.

Conforme discutido na seção anterior, já existem diversos padrões de qualidade amplamente aceitos e exigidos pela comunidade de desenvolvedores. É importante destacar que todos os repositórios analisados nesta pesquisa possuem mais de 3.000 estrelas no GitHub, um indicativo de relevância e visibilidade, e atingiram esse patamar justamente por adotarem boas práticas e padrões de qualidade desde o início de seu desenvolvimento.

Referências

- GitHub GraphQL API. Disponível em: https://api.github.com/graphql
- CK - Chidamber and Kemerer Java Metrics. Disponível em: https://github.com/mauricioaniche/ck
- Pandas: Powerful Python data analysis toolkit. Disponível em: https://pandas.pydata.org/
- Matplotlib: Visualization with Python. Disponível em: https://matplotlib.org/
- GitHub REST and GraphQL documentation. Disponível em: https://docs.github.com/en/graphql