Um estudo das características de qualidade de sistemas Java

Sumário

1. Resumo

Este relatório apresenta a análise de 1.000 repositórios Java populares no GitHub, cruzando características de processo (popularidade, idade, atividade, tamanho) com métricas de qualidade calculadas pela ferramenta CK (WMC, LCOM, CBO, DIT e demais métricas por classe). Os achados principais são: popularidade (stars) tem correlação fraca com as métricas estáticas; tamanho (LOC) e idade tendem a se associar a maior complexidade (especialmente WMC) e maior acoplamento (CBO); atividade está associada a maior variabilidade das métricas, com picos de complexidade entre repositórios ativos; e LCOM (cohesão) permanece baixa na maioria dos projetos. Também foram detectados outliers extremos que exigem verificação e tratamento porque distorcem médias e visualizações.

2. Introdução

Num cenário open source em que várias pessoas contribuem simultaneamente, atributos internos do software — modularidade, manutenibilidade, legibilidade — podem evoluir de forma imprevisível. Ferramentas de análise estática e práticas como CI/CD tentam mitigar esse risco, mas é importante entender empiricamente como características do processo se relacionam com métricas de produto. Este trabalho busca responder quatro perguntas: (1) popularidade x qualidade, (2) maturidade x qualidade, (3) atividade x qualidade e (4) tamanho x qualidade, usando o conjunto dos 1.000 repositórios Java mais estrelados no GitHub e métricas CK agregadas por repositório.

3. Metodologia

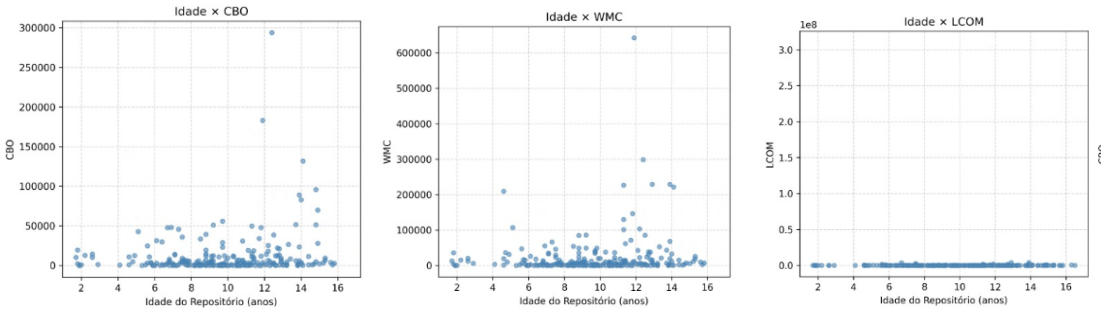
Foram coletados os **1.000 repositórios Java mais populares**, extraídas métricas de processo **via GitHub API** (stars, idade em anos, LOC, número de releases/última atualização) e executada a **ferramenta CK** para obter métricas por classe. Os resultados do CK foram agregados por repositório (média, mediana, desvio padrão) para permitir comparações. Para análise usou-se descrição estatística, visualizações.

4. Resultados e análise (RQ01 a RQ04)

RQ01 — Popularidade x Qualidade. O heatmap de correlação entre stars e as métricas CK indica que, em geral, stars tem correlação muito fraca com WMC, CBO, LOC e as demais métricas analisadas. Em termos práticos, encontramos repositórios populares tanto entre os que têm boa organização interna quanto entre os muito complexos. Isso sugere que número de estrelas não é, isoladamente, um bom preditor de qualidade de código medida por métricas estáticas. Recomendo calcular correlações parciais controlando LOC e idade, porque esses são fatores de confusão óbvios.

RQ02 — Maturidade (idade) x Qualidade. Ao examinar os dados segundo a maturidade dos repositórios, percebemos que os **maiores valores de WMC concentram-se sobretudo entre os projetos mais antigos** — a maioria dos picos aparece em repositórios com cerca de 12 a 14 anos. Repositórios mais jovens raramente atingem níveis semelhantes de complexidade. Em contraste, a **LCOM mostra-se relativamente uniforme ao longo das idades**, o que sugere que, **na maior parte dos projetos, a coesão entre métodos é mantida mesmo em iniciativas recentes**. Já o **CBO tende a ser mais elevado nos repositórios com mais de 10 anos, enquanto os mais novos apresentam valores menores na maior parte das vezes**; um CBO alto indica que classes dependem de muitas outras e, assim, contribui para um aumento da complexidade e de dificuldades em testes e manutenção, impactando também o WMC agregado.

Conclusão que obtive: Projetos antigos costumam acumular mais complexidade e acoplamento, embora possam preservar boa coesão; portanto, a idade por si só não garante maior qualidade do software.



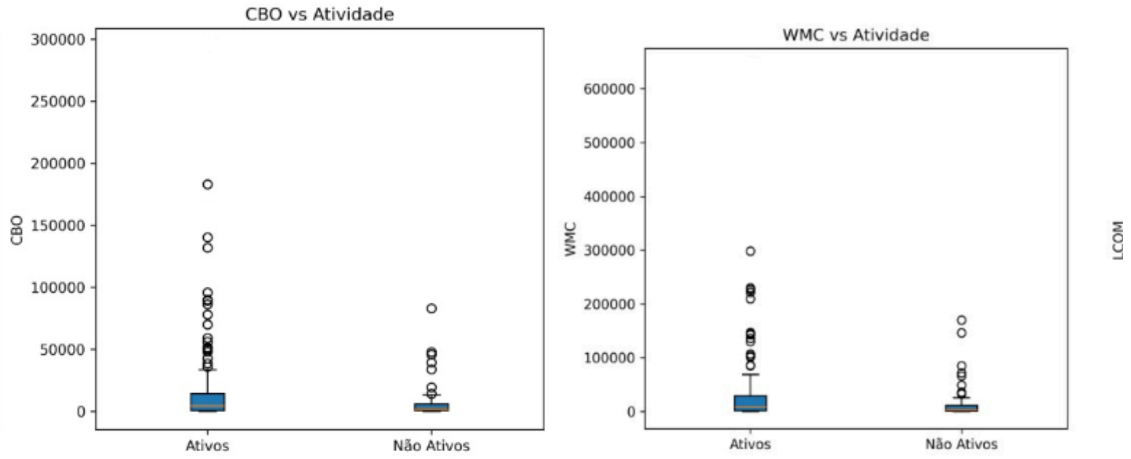
RQ03 — Atividade x Qualidade. Observamos que, em relação ao **WMC, os repositórios ativos apresentam uma maior variação**. Já os **repositórios inativos costumam ter WMC menores**. Isso sugere que **projetos ativos tendem a ter classes mais complexas, possivelmente porque continuam em evolução e acumulando funcionalidades**. No entanto, alguns dados indicam que alguns projetos ativos podem estar excessivamente complexos, sinalizando a necessidade de refatoração.

No caso do **LCOM, os valores são geralmente baixos em ambos os grupos**, mas há outliers extremos, especialmente entre os repositórios inativos. Esses casos podem representar projetos com classes

desorganizadas ou que não foram mantidos, reforçando que a falta de atividade pode levar a problemas na qualidade estrutural do código.

Quanto ao **CBO, ele tende a ser maior e mais disperso em repositórios ativos**, enquanto os inativos apresentam valores menores, ainda que existam alguns outliers. Isso indica que projetos ativos possuem maior acoplamento entre classes, provavelmente devido à adição contínua de funcionalidades e integrações, enquanto os inativos permanecem mais simples ou congelados.

Conclusão que obtive: Portanto, **a maior atividade de um repositório não significa necessariamente que ele tenha melhor qualidade**. Projetos ativos podem ser mais complexos, enquanto os inativos podem ter menor acoplamento, mas não necessariamente uma organização melhor. Em resumo, atividade recente está mais relacionada à evolução funcional e à manutenção contínua do que à qualidade estrutural do código.

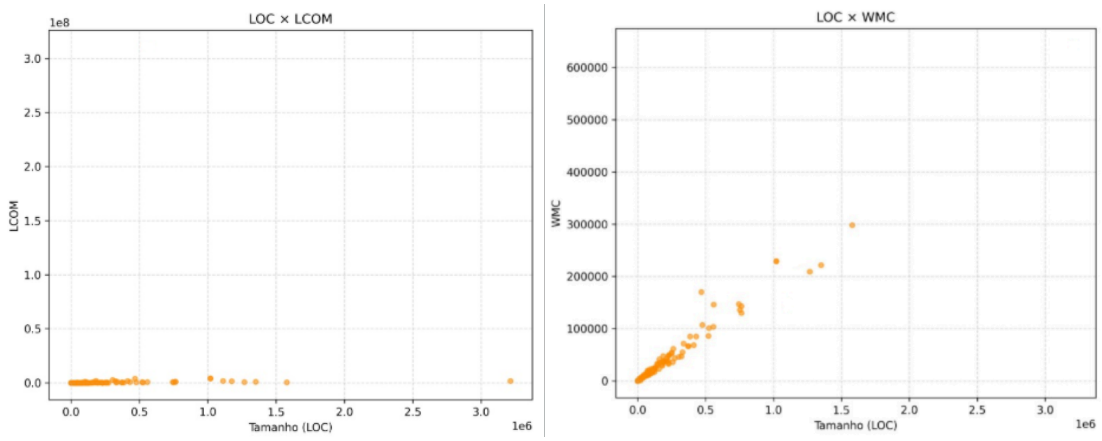


RQ04 — Tamanho (LOC) x Qualidade. Para investigar a relação entre tamanho e qualidade, utilizamos o LOC (Lines of Code) como medida de tamanho e examinamos como ele se relaciona com indicadores de qualidade, como WMC (Weighted Methods per Class) e LCOM (Lack of Cohesion of Methods).

A análise do gráfico que relaciona LOC e WMC mostra uma tendência crescente: **repositórios maiores geralmente possuem classes com mais métodos**, indicando que a complexidade tende a aumentar junto com o tamanho do projeto.

Já na análise do LCOM, observamos que repositórios populares apresentam baixa coesão normalmente. Isso sugere que, mesmo em projetos grandes e complexos, os módulos podem se manter relativamente coesos, refletindo uma boa organização do código.

Conclusão que obtive: Em resumo, **o crescimento do tamanho de um repositório está associado ao aumento da complexidade das classes**, o que pode tornar a manutenção e a compreensão do software mais desafiadoras. No entanto, a baixa LCOM em projetos populares indica que equipes bem estruturadas conseguem manter a coesão e a qualidade do código mesmo em sistemas maiores.



5. Problemas detectados nos dados

Houve outliers extremos repetidos: WMC na casa das centenas de milhares, CBO também em centenas de milhares e um LCOM massivamente alto. Possíveis causas são agregação incorreta (somando em vez de tirar média por classe), inclusão de código de terceiros ou erro na execução/leitura do CK. Esses outliers distorcem médias e pioram a leitura dos gráficos em escala linear. Outra questão é a definição de "ativo" que muda resultados se a janela escolhida for diferente. Por isso, para todas as análises usei e recomendo: mediana, IQR, Spearman e, visualmente, gráficos em escala log quando houver outliers.

6. Discussão e interpretação geral

No conjunto, não existe um fator isolado que garanta qualidade. Popularidade não corresponde necessariamente a qualidade de código. **Tamanho e idade estão associados ao aumento da complexidade e do acoplamento**, o que faz sentido do ponto de vista prático: com mais tempo e mais linhas, há mais espaço para crescer acoplamentos e método-por-classe. Atividade aumenta variabilidade e picos, porque evolução contínua frequentemente adiciona funcionalidades e integrações. Ainda assim, muitas equipes conseguem manter coesão (LCOM baixa), sugerindo que boas práticas — revisão, design modular, CI/CD — mitigam o acúmulo de débitos. Os outliers detectados podem representar casos reais de monolitos ou erros de contagem; por isso, antes de afirmar conclusões políticas, valem inspeção manual desses casos.

7. Conclusão

Sumarizando, a investigação mostra que popularidade por si só não garante qualidade, maturidade e tamanho tendem a aumentar complexidade e acoplamento, e atividade acompanha maior variabilidade das métricas. Coesão é frequentemente preservada, o que é bom. No entanto, outliers extremos exigem limpeza de dados e verificação de agregação antes de qualquer conclusão

definitiva. Com os ajustes recomendados e testes estatísticos robustos, os resultados podem ser quantificados com confiança.