Características de Qualidade de Sistemas Java

Lucca V. P. Bessa[1] , Samuel R. Freitas[1]

[1]Bacharelado em Engenharia de Software
Instituto de Ciências Exatas e Informática - PUC Minas
Ed. Fernanda. Rua Cláudio Manoel, 1.162, Funcionários
Belo Horizonte – MG – Brasil

Introdução

A qualidade do software de código aberto é um fator crítico para seu sucesso. Para manter a alta qualidade, os projetos de código aberto devem ser capazes de gerenciar com eficácia os riscos associados ao desenvolvimento colaborativo.

Um dos aspectos mais importantes da qualidade do código aberto é a qualidade do próprio código. Isso inclui fatores como modularidade, facilidade de manutenção e legibilidade. A qualidade do código pode ser melhorada por meio de uma variedade de práticas, como revisão de código, análise estática e integração e entrega contínuas (CI/CD).

A qualidade de um repositório de software é um fator importante a ser considerado na escolha de um projeto para contribuir ou utilizar em um projeto próprio. Por isso, várias pesquisas têm sido realizadas para entender melhor a relação entre as características de qualidade dos repositórios e fatores como popularidade, maturidade, atividade e tamanho.

Em relação à popularidade dos repositórios, uma hipótese é que os mais populares tendem a ter boas características de qualidade. Isso se deve à atuação ativa da comunidade de software na manutenção desses repositórios, além de uma forte cultura de code review nas contribuições.

Já em relação à maturidade dos repositórios, a hipótese é que os mais antigos tendem a ter características de qualidade melhores. Isso porque, com o tempo, existe uma tendência de refatoração e melhoria dos códigos já desenvolvidos, o que contribui para a melhoria das métricas de qualidade.

No que se refere à atividade dos repositórios, a hipótese é que os mais ativos tendem a ter métricas de qualidade melhor. Isso porque a atividade de um repositório é um dos principais indicadores do esforço de melhoria do código, o que tem impacto direto nas métricas de qualidade do software.

Por fim, em relação ao tamanho dos repositórios, a hipótese é que quanto maior um repositório, maior é a tendência de ele possuir melhores métricas de qualidade. Isso se deve à implementação de testes unitários e de integração, que têm grande impacto tanto nas métricas de qualidade quanto no tamanho do repositório. Além disso, o número de comentários também tem um impacto positivo nas métricas de qualidade e está diretamente associado ao crescimento do número de linhas de um repositório.

Em resumo, essas hipóteses sugerem que há uma forte relação entre as características de qualidade dos repositórios e fatores como popularidade, maturidade, atividade e tamanho. Compreender essas relações pode ajudar a escolher projetos mais promissores e identificar oportunidades para melhorar a qualidade do código de um repositório.

Neste laboratório, analisaremos a qualidade de repositórios Java desenvolvidos em ambiente colaborativo. Iremos correlacionar as características de qualidade desses repositórios com as características de seu processo de desenvolvimento, como popularidade, maturidade, atividade e tamanho. Nosso objetivo é identificar os fatores que contribuem para um código Java de alta qualidade.

Usaremos a ferramenta CK para calcular métricas de produto para os repositórios em nosso estudo. Esta é uma ferramenta para medir a qualidade de produtos de software. Ela inclui várias métricas que podem ser usadas para avaliar a qualidade do código, como complexidade ciclomática, acoplamento e coesão.

Usaremos essas métricas para comparar a qualidade de diferentes repositórios. Também os usaremos para identificar os fatores que estão correlacionados com o código de alta qualidade. Nosso objetivo é fornecer insights que possam ajudar os desenvolvedores a melhorar a qualidade de seus projetos de código aberto.

Esperamos que nossas descobertas sejam de interesse para desenvolvedores e pesquisadores de código aberto. Nossos resultados podem ser usados para identificar os fatores que contribuem para um código Java de alta qualidade. Essas informações podem ser usadas para melhorar o processo de desenvolvimento de projetos de código aberto e para criar melhores produtos de software.

Metodologia

A metodologia adotada neste estudo consiste em uma análise exploratória dos top 1.000 repositórios Java mais populares do GitHub, com o objetivo de investigar a relação entre suas características de processo e qualidade. Para tal, definimos quatro questões de pesquisa, cada uma relacionada a uma métrica de processo específica: popularidade, tamanho, atividade e maturidade.

A coleta dos repositórios foi realizada por meio de uma consulta ao GitHub API, ordenada por popularidade decrescente. Em seguida, foram extraídas as informações relevantes, como número de estrelas, número de releases e idade do repositório.

Para calcular as métricas de qualidade das linhas de código, utilizamos a ferramenta CK, que é uma ferramenta de análise estática de código-fonte. Utilizamos as métricas de qualidade CBO (Coupling Between Objects), DIT (Depth Inheritance Tree) e LCOM (Lack of Cohesion of Methods), que são amplamente utilizadas em estudos anteriores relacionados à qualidade de software. Cada métrica foi calculada para todos os repositórios coletados.

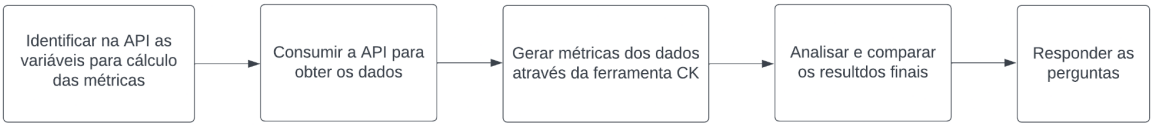

Figura 1. Fluxograma da metodologia

Para responder às questões de pesquisa, realizaremos uma comparação entre as características de processo e os valores obtidos para as métricas de qualidade. Para tanto, utilizaremos gráficos de dispersão, nos quais os valores das métricas de processo serão representados nos eixos X e Y.

Resultados e análise das amostras

A partir dos questionamentos levantados no enunciado do estudo e após a aplicação da metodologia, foram desenvolvidos doze gráficos representativos para ilustrar os resultados. Para cada característica abordada nas questões de pesquisa, foram gerados três gráficos, sendo cada um deles associado a uma métrica de qualidade de software dentre as métricas selecionadas, sendo elas: CBO, DIT e LCOM. Abaixo analisamos os resultados apresentados por cada gráfico.

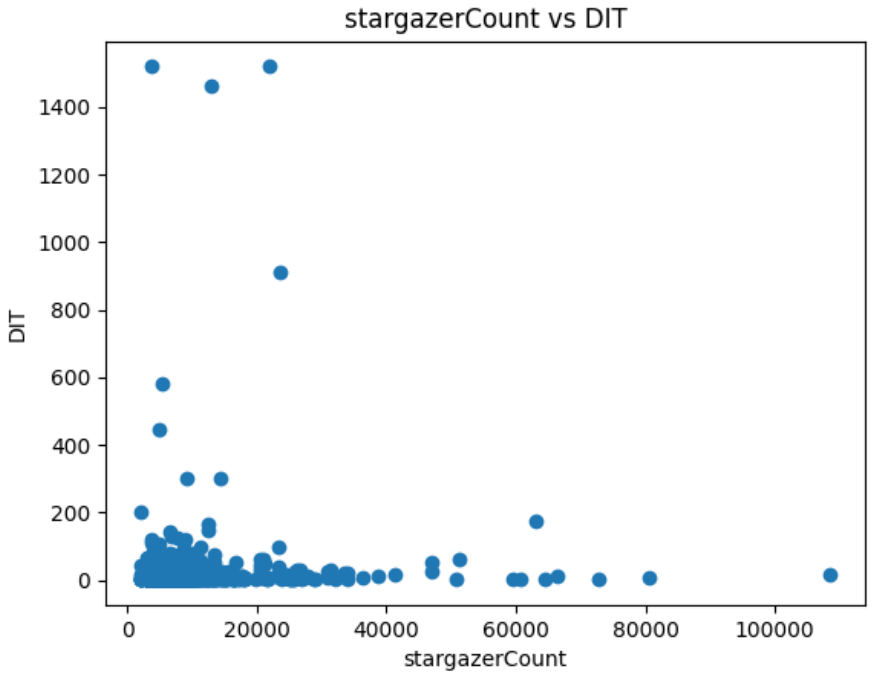

Gráfico 1. Estrelas vs DIT

Com base na dispersão do gráfico acima, é possível perceber que a medida que a popularidade dos repositórios aumenta, a métrica DIT diminui, com isso é possível perceber uma melhoria no código dos repositórios mais populares.

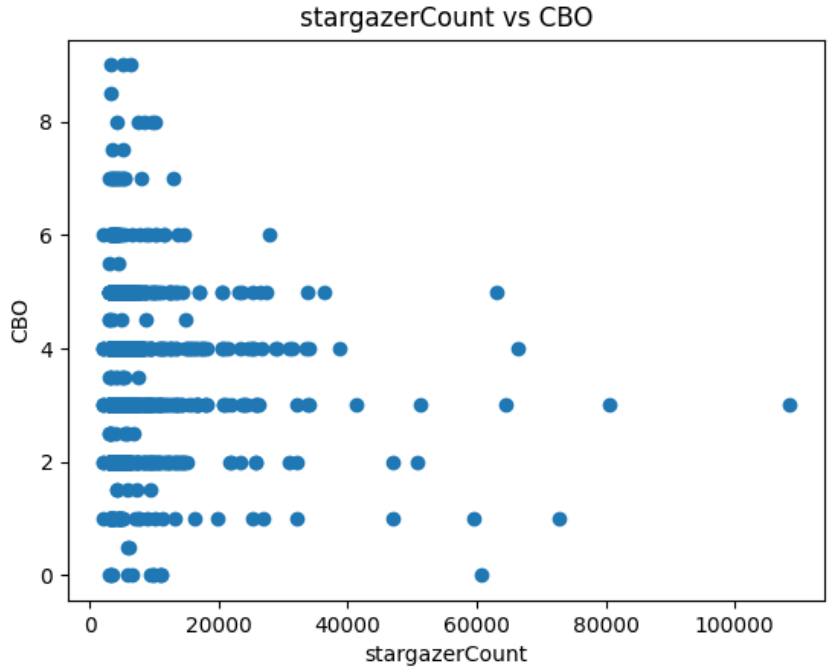

Gráfico 2. Estrelas vs CBO

A partir da análise do gráfico acima, percebe-se que a popularidade não possui influência direta sobre a métrica CBO dos repositórios. A maior parte dos dados se concentra entre 0 e 6 para uma grande variação na popularidade dos repositórios.

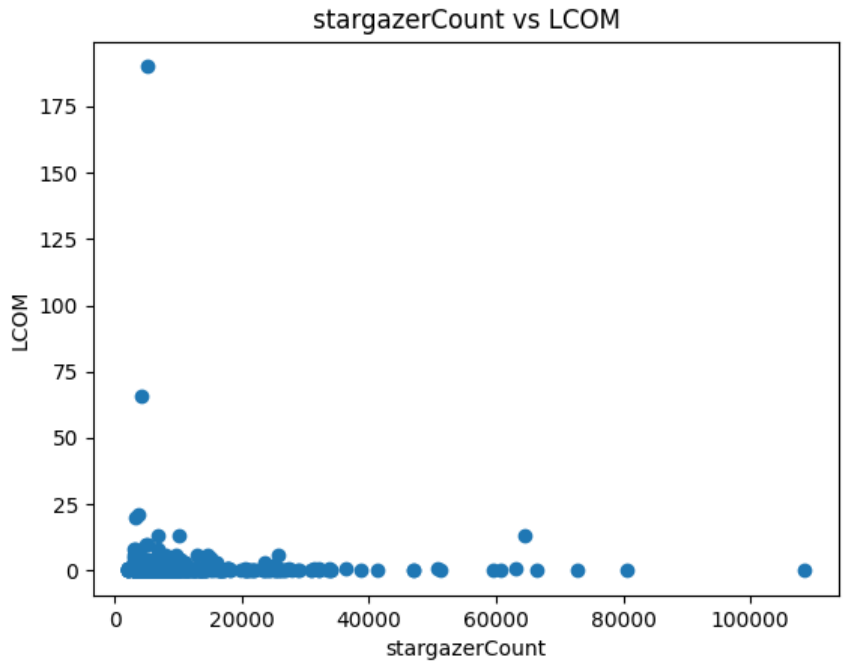

Gráfico 3. Estrelas vs LCOM

Com a análise do gráfico percebemos que a LCOM diminui conforme ocorre o aumento da popularidade dos repositórios.

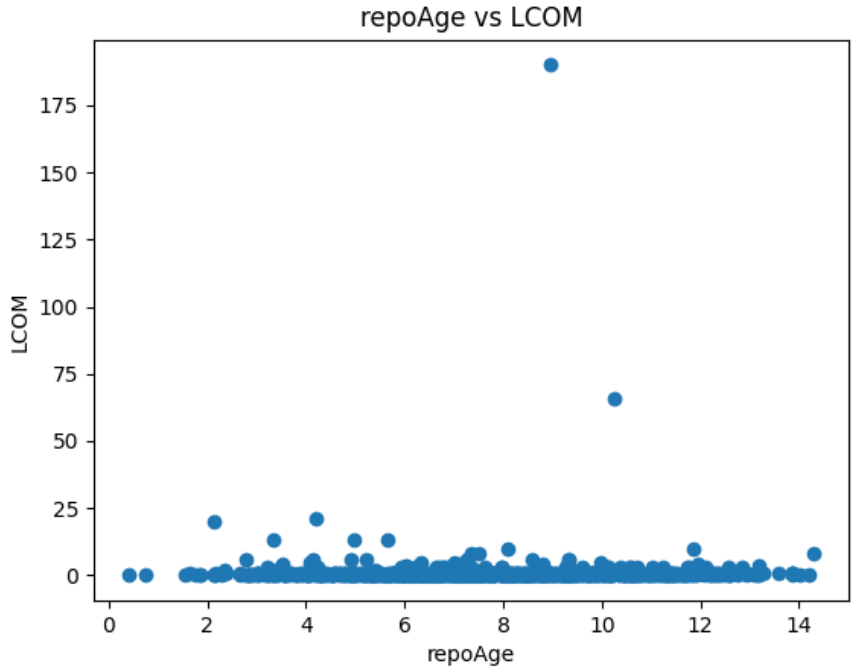

Gráfico 4. Idade vs LCOM

À medida que a idade dos repositórios aumenta, a tendência é que a métrica LCOM se mantenha constante. A métrica varia pouco para uma grande dispersão de idades de repositórios.

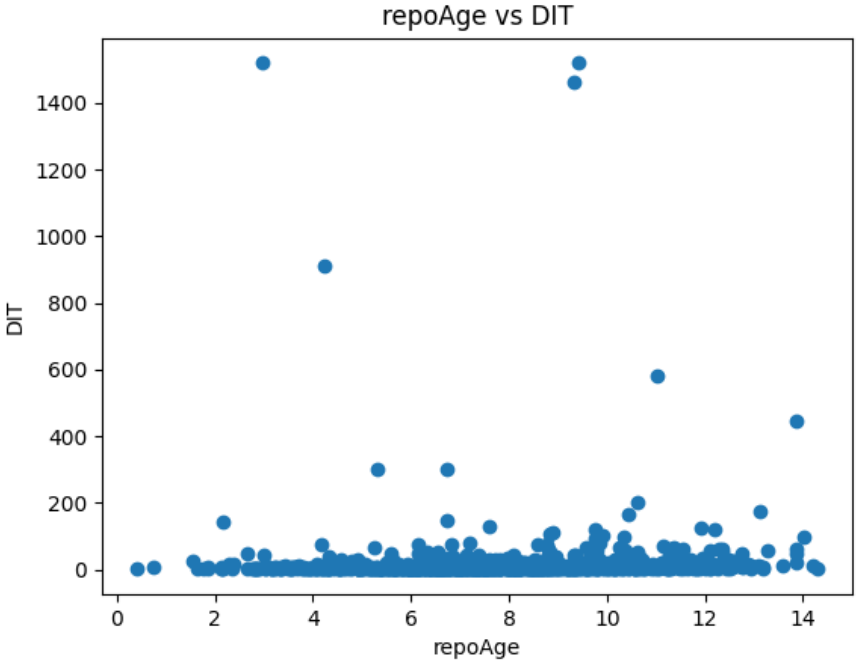

Gráfico 5. Idade vs DIT

A distribuição acima evidencia a independência entre a métrica DIT e a idade dos repositórios. Conforme a idade dos repositórios aumenta, a métrica se mantém com uma pequena variação.

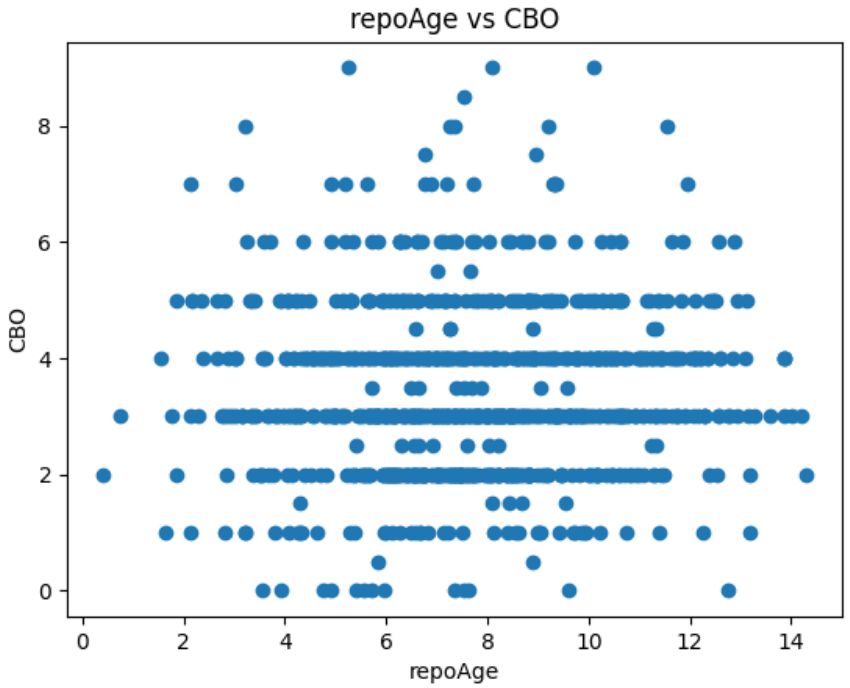

Gráfico 6. Idade vs CBO

A idade dos repositórios não demonstra ter impacto direto sobre a métrica CBO. A variação da métrica se manteve uniforme mesmo com o aumento da idade dos repositórios.

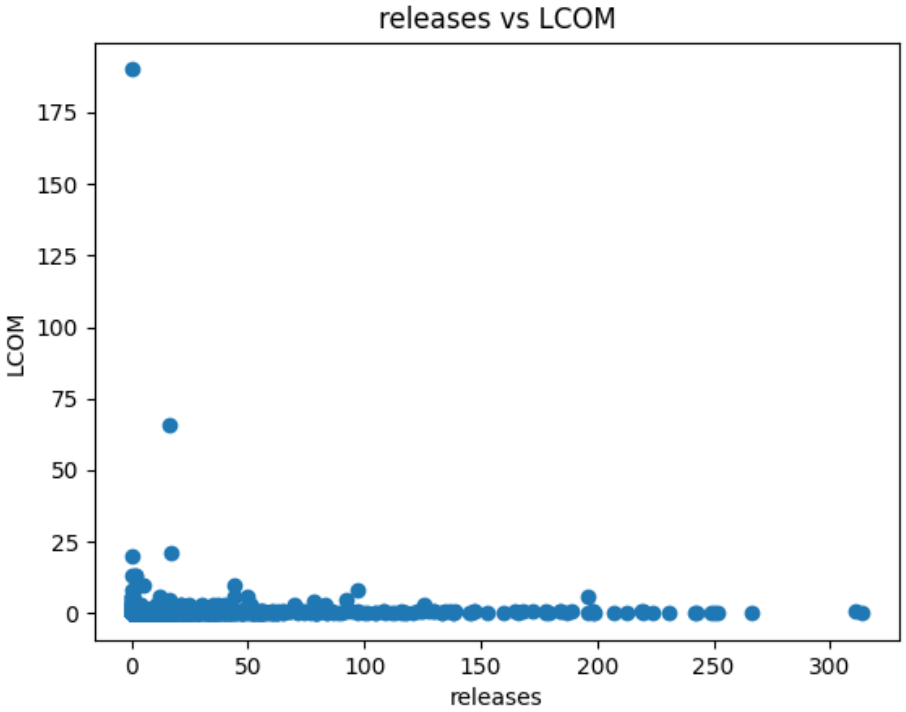

Gráfico 7. Número de Releases vs LCOM

Com base no gráfico é possível perceber que conforme o número de releases aumenta, a métrica LCOM diminui. Isso é um indicativo de melhora no código conforme o número de releases aumenta.

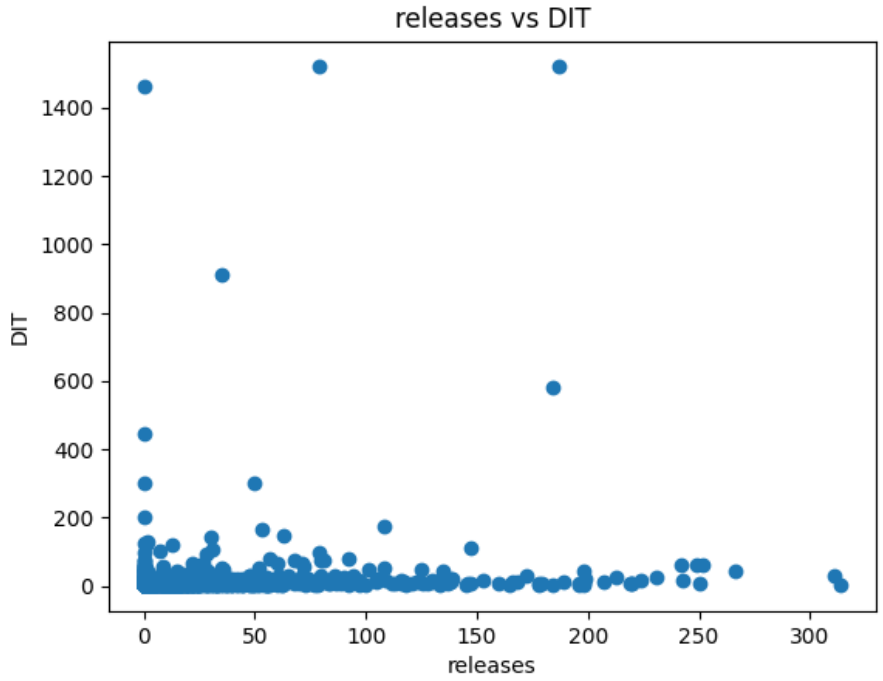

Gráfico 8. Número de Releases vs DIT

A dispersão do gráfico evidencia uma diminuição na métrica DIT conforme o número de releases aumenta.

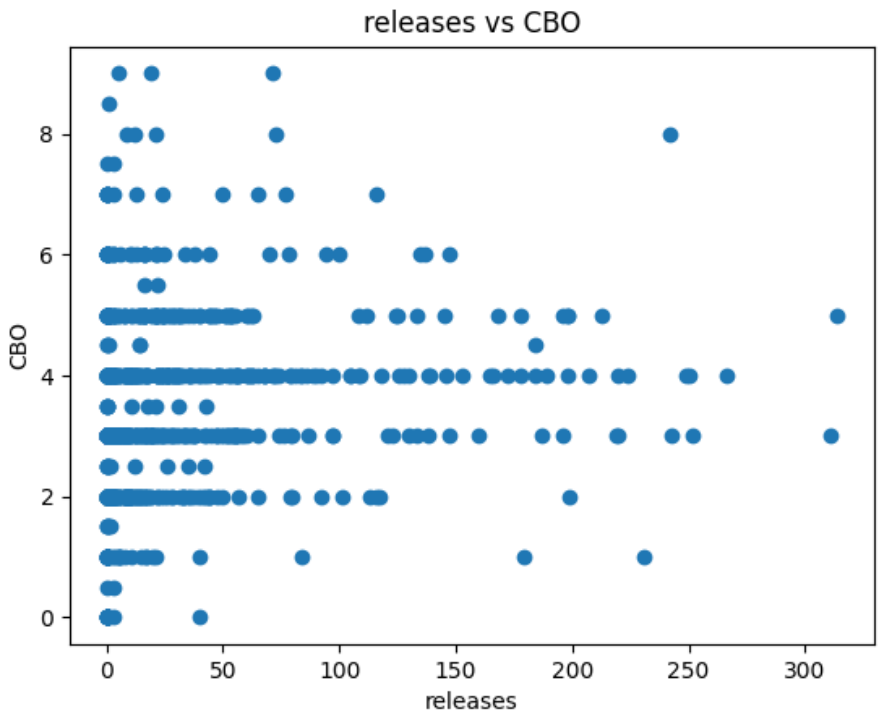

Gráfico 9. Número de Releases vs CBO

Com base no gráfico é possível perceber que o número de releases não influencia na métrica CBO. A dispersão não demonstra um padrão de variação conforme o número de releases aumenta.

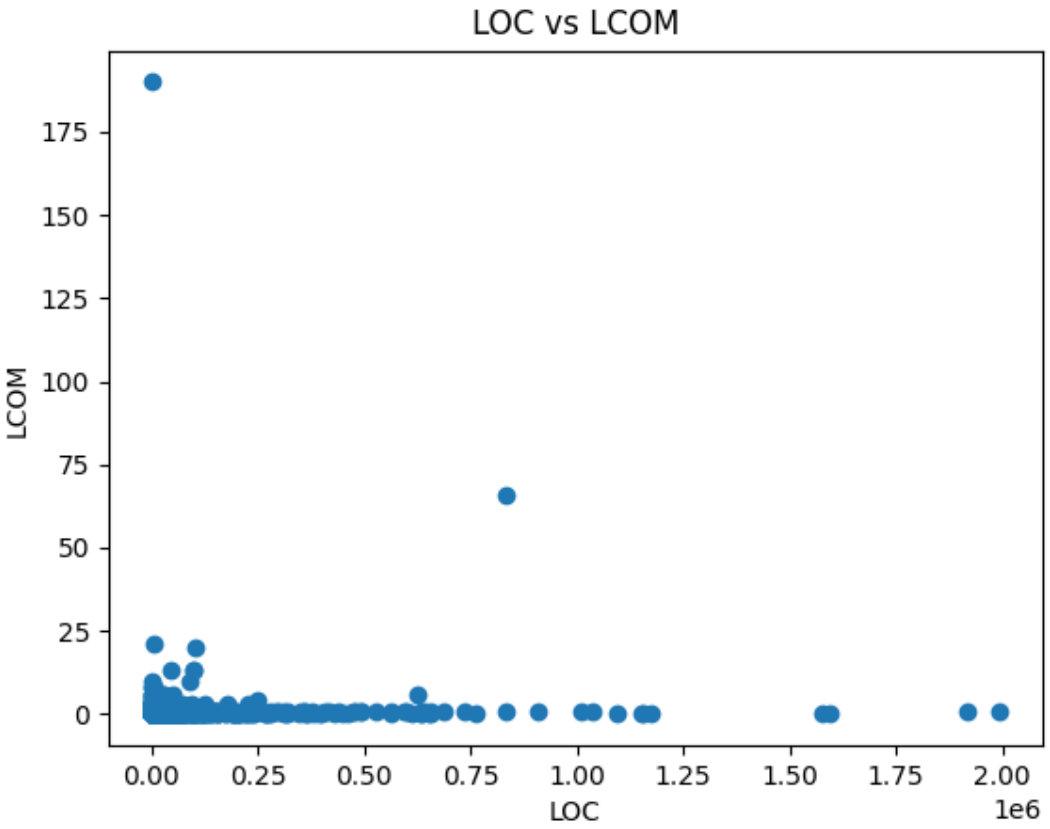

Gráfico 10. Número de linhas de código vs LCOM

A dispersão do gráfico indica que conforme o número de linhas de um repositório aumenta, a métrica LCOM para este repositório diminui.

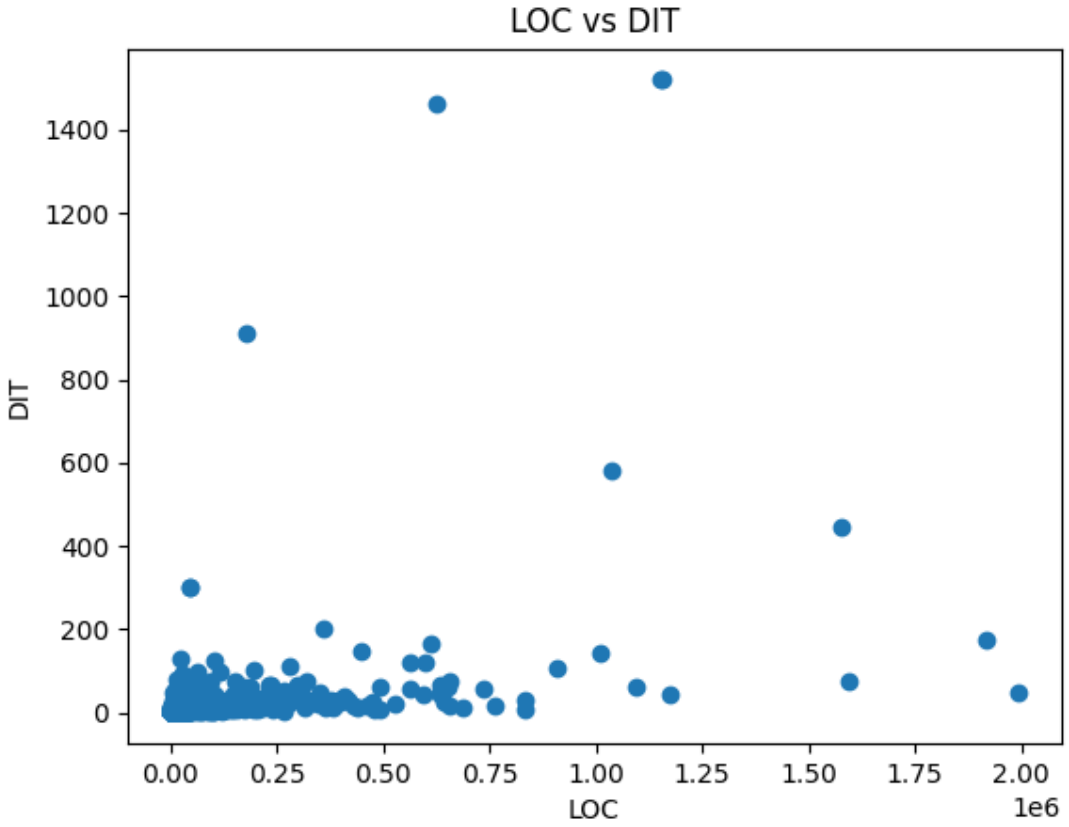

Gráfico 11. Número de linhas de código vs DIT

O gráfico acima possui uma grande concentração de dados para para repositórios com uma menor quantidade de linhas. Isso indica que conforme o número de linhas de um repositório aumenta, a tendência é que a métrica DIT para esse repositório diminua.

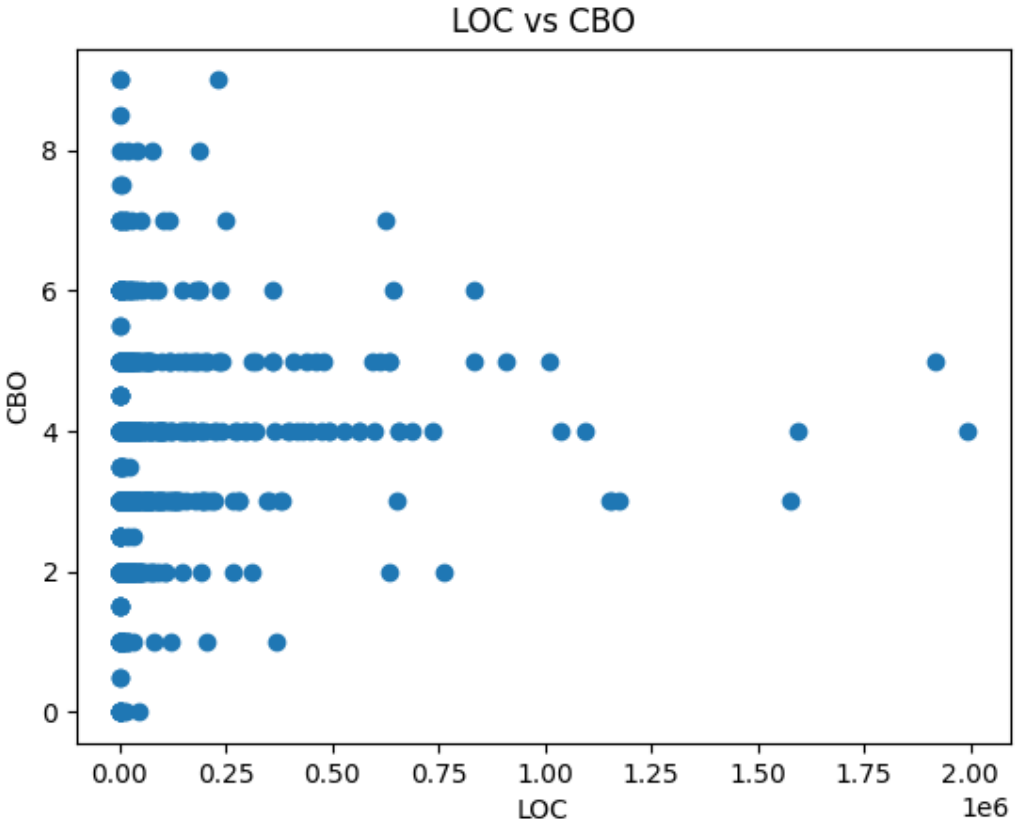

Gráfico 11. Número de linhas de código vs CBO

O acoplamento entre objetos não tem relação direta com o número de linhas de um repositório. A dispersão do gráfico não possui um padrão de crescimento ou decrescimento à medida que o número de linhas de um repositório aumenta.

Discussão

Após a coleta dos dados e análise das métricas, foi possível perceber que nem todas as hipóteses estavam corretas. Apesar disso, nenhuma hipótese foi refutada, o que indica que estudos mais profundos sobre o assunto poderiam nos levar às respostas para as questões propostas.

A primeira hipótese dizia que repositórios mais populares tendem a ter uma melhor qualidade de código. Os dados coletados não foram suficientes para encontrar uma relação direta entre a popularidade de um repositório e a qualidade de seu código, pois as métricas utilizadas não demonstraram um padrão de crescimento ou decrescimento que nos permitisse extrair conclusões assertivas sobre o relacionamento citado.

Em paralelo, a segunda hipótese sofreu da mesma circunstância citada anteriormente. Os dados coletados não tiveram uma variação que indicasse um padrão na relação entre a qualidade de código e a idade de um repositório, ou seja, não necessariamente a idade de um repositório implica em que a qualidade de seu código melhore.

Em contraste às duas hipóteses anteriores, a terceira hipótese foi confirmada após a análise dos dados coletados. Essa hipótese consistia em que repositórios mais ativos tendem a ter métricas de qualidade melhores, o que foi comprovado ao se observar que todas evoluíram positivamente enquanto o número de releases de um repositório cresce. A cada release nova, a tendência é que a qualidade do software continue aumentando devido a refatorações constantes.

Por fim, a última hipótese também foi confirmada, ao se observar que as métricas de qualidade de software evoluíram positivamente conforme os repositórios aumentam de tamanho. A tendência é que repositórios grandes tenham uma maior quantidade de testes unitários e outros recursos que contribuem para a manutenção de um código de qualidade.