Análise de top repositórios em Java do Github

Alfredo Luis Vieira, Vinicius Salles

16 de setembro de 2025

Introdução

Este estudo se insere no contexto open-source, propondo uma análise sobre os aspectos da qualidade de software em repositórios desenvolvidos na linguagem Java. O objetivo principal é investigar a correlação entre as características do processo de desenvolvimento desses repositórios e suas métricas de qualidade de produto. Para isso, serão examinados os 1.000 repositórios Java mais populares da plataforma GitHub, utilizando a ferramenta de análise estática CK para o cálculo de métricas de produto.

A investigação busca responder a questões fundamentais sobre como fatores como popularidade (número de estrelas), maturidade (idade do repositório), atividade (número de releases) e tamanho (linhas de código) se relacionam com indicadores de qualidade interna, especificamente o Acoplamento Entre Objetos (CBO), a Profundidade da Árvore de Herança (DIT) e a Falta de Coesão dos Métodos (LCOM). Por meio desta análise, o trabalho pretende oferecer insights sobre a evolução da qualidade em projetos open-source de grande relevância e visibilidade na comunidade de desenvolvimento

Metodologia

Para a realização deste estudo, utilizamos algumas técnologias como API GraphQL do github, que foi utilizada como fonte para a mineração dos dados, o CK, que é um sistema capaz de fazer analises quantitativas de repositorios, obtendo dados como KLOC, e etc, e o python, que foi a linguagem na qual os scripts foram desenvolvidos.

Primeiramente, obtivemos os dados a partir de requisições junto a API do github, esse foi o ponto da nossa primeira dificuldade, a API estava retornando 502 muitas vezes, devido a timeout, o que nos obrigou a particionar nossa pesquisa em lotes, que eram representados a partir do numero de estrelas dos repositorios.

Com os dados obtidos, e as métricas de projeto coloteadas utilizando scripts em python, partimos para a proxima etapa do projeto, que foi a utilização do CK para realizar a analise mais profunda destes repositórios. Com o Ck, utilizamos a seguinte estratégia, para cada repositório, clonamos o mesmo utilizando a ferramenta GIT, e a partir do repositório clonado em nossa maquina, o CK realizaria a analise. Uma das principais dificuldades encontradas foi que existiam alguns projetos muito complexos, cujo CK encontrou dificuldades para realizar a analise, alguns foram até impossiveis de serem analisados pela ferramenta, devido a complexidade, ou versão de java não suportada. O tempo tambem foi nosso inimigo, pois o processamento dos repositório estava muito lento, o que nos obrigou a utilizar estrategias de *MultiThreading* para poder executar o trabalho com mais velocidade.

Questões de pesquisa

Neste trabalho, a partir dos dados obtidos através dos repositorios, foram estipulados uma sequencia de perguntas que devem ser respondidas ao final do estudo, sendo elas:

- Qual a relação entre a popularidade dos repositórios e as suas características de qualidade?
- Qual a relação entre a maturidade do repositórios e as suas características de qualidade ?
- Qual a relação entre a atividade dos repositórios e as suas características de qualidade?

- Qual a relação entre o tamanho dos repositórios e as suas características de qualidade?

Ao final do trabalho, utilizando estas perguntas como guia, poderemos ter uma noção sobre qualidade, e como ela está relacionada com os repositórios, a partir de sua maturidade.

Métricas utilizadas

Neste estudo, para chegarmos ao nosso objetivo, estipulamos diversas métricas, algumas de processo, sendo elas:

- Popularidade: número de estrelas
- Tamanho: linhas de código (LOC) e linhas de comentários
- Atividade: número de releases
- Maturidade: idade (em anos) de cada repositório coletado

e também algumas de qualidade, sendo elas:

- CBO: Coupling between objects
- DIT: Depth Inheritance Tree
- LCOM: Lack of Cohesion of Methods

A partir dessas métricas poderemos ter uma visão ampla de como estes projetos estão estruturados, e verificar se sua qualidade conversa diretamente com sua maturidade.

1 Considerações gerais

Abaixo analisaremos os valores obtidos atráves das métricas analisadas nos respositorios, a titulo de informação acerca da situação atual dos top repositórios de JAVA

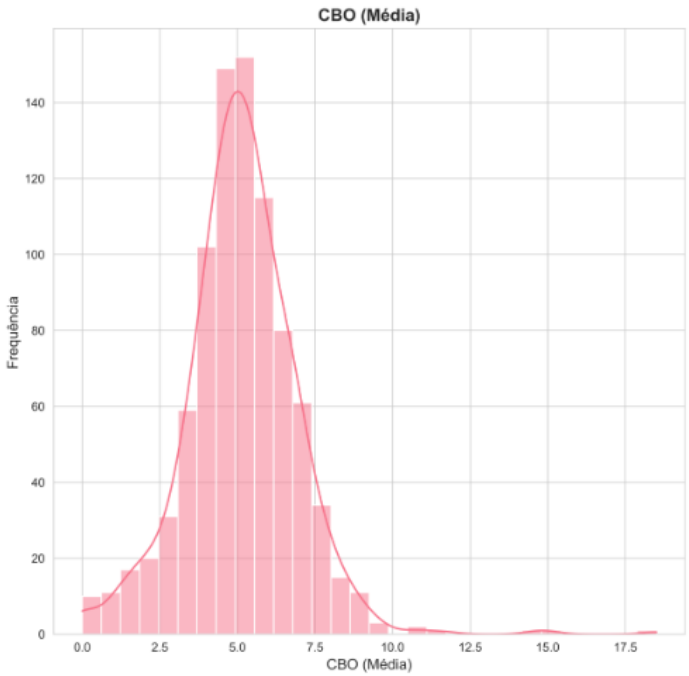

Figura 1: CBO

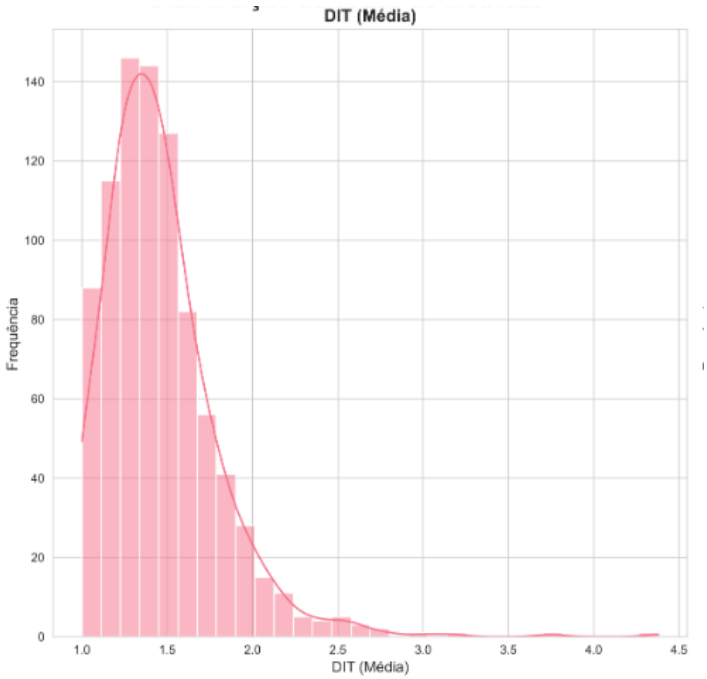

Figura 2: DIT

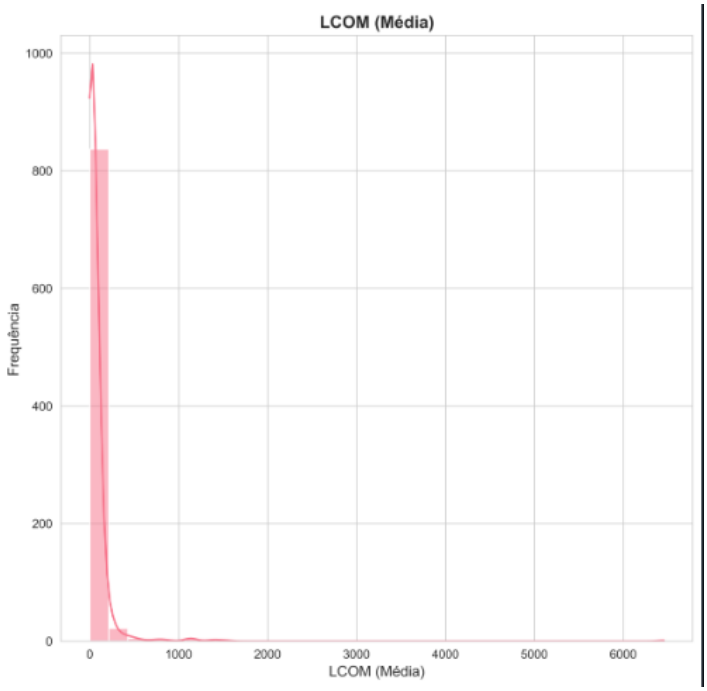

Figura 3: LCOM

Primeiramente analisando o CBO, podemos identificar uma moda que gira entorno do 5.0, mas com repositorios com valores um pouco maiores, ou um pouco menores que sua mediana, o que nos mostra uma tendencia a valores entre 3.0 e 7.5.

Já sobre o DIT, são identificados um grande numero de repositorios com valores em torno de 1.5, sendo sua maioria, flutuando entre essa média.

E por fim, o LCOM, podemos observar uma maioria esmagadora, que flutua até o limite máximo de 200, e um numero muito pequeno, possuindo valores acima desse limite.

Estas analises iniciais nos mostram uma estabilidade em questão de métricas de qualidade com relação a estes repositorios, pois os mesmos possuem valores semelhantes, podendo até ser observado um padrão de valores para estas métricas.

2 Resultados obtidos

Nesta sessão apresentaremos os resultados obtidos através das análises, sumarizados seguindo a ordem das questões apresentadas.

2.1 Qual a relação entre a popularidade dos repositórios e as suas características de qualidade?

A primeira coisa que analisamos foi o CBO, que fala sobre o acoplamento entre classes e objetos, no boxplot abaixo, observamos que em repositorios com mais de 50k de estrelas, que são considerados repositorios de alta popularidade, a taxa de CBO é menor, flutuando entre 2,0 e 2,1, enquanto em repositorios menos populares, a média do cbo chega em 5.0. Podemos tirar disso a informação que repositorios mais populares tendem a ter sua arquitetura mais organizada e bem planejada, evitando acoplamentos desnecessarios, podemos tambem dizer que esta arquitetura mais limpa permite que o projeto escale, ganhando mais popularidade e engajamento dos usuários.

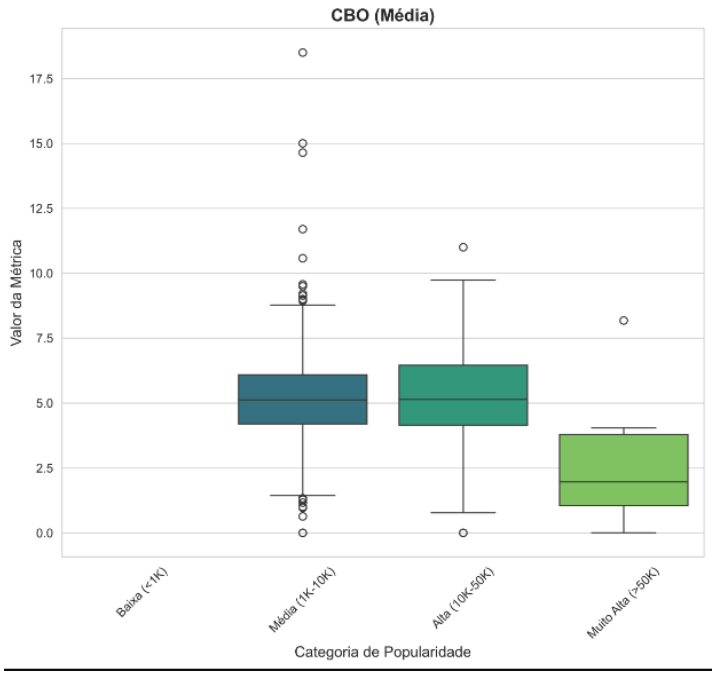

Figura 4: CBO x popularidade

A seguir, analisamos o DIT, que é uma métrica especifica para OOP, onde as heranças de um projeto são analisadas, essa métrica basicamente, mede comprimento do caminho mais longo desde essa classe até a classe raiz (a classe mais alta na hierarquia), conforme o BoxPLot abaixo, podemos identificar que o DIT tende a diminuir a medida que a popularidade do projeto aumenta, mas isso nem sempre é positivo, esta métrica tende a ser como uma faca de dois gumes, com valores altos, tendo como ponto negativo, quanto mais profunda a classe na hierarquia, mais métodos e atributos ela pode herdar. Isso aumenta a complexidade para entender o comportamento da classe, tornando-a mais difícil de prever e manter,e como positivo; uma hierarquia mais profunda sugere um maior potencial de reutilização de código, pois as classes filhas podem aproveitar os métodos herdados das classes pai, e valores baixos: ponto Positivo: Implica menor complexidade, tornando o sistema potencialmente mais

fácil de entender e manter, e ponto Negativo: Pode indicar menor reutilização de código através do mecanismo de herança.

Podemos identificar que a medida que os repositorios vão possuindo maior popularidade, eles vão ficando menos complexos, diminuindo seu DIT, mas tambem diminuindo sua reusabilidade de código. Podemos identificar outliers, principalmente em repositórios com uma média de 1k-10k estrelas, estes repositórios possuem DIT altíssimo, o que indicam a presença de classes complexas, e uma possível necessidade de refatoração.

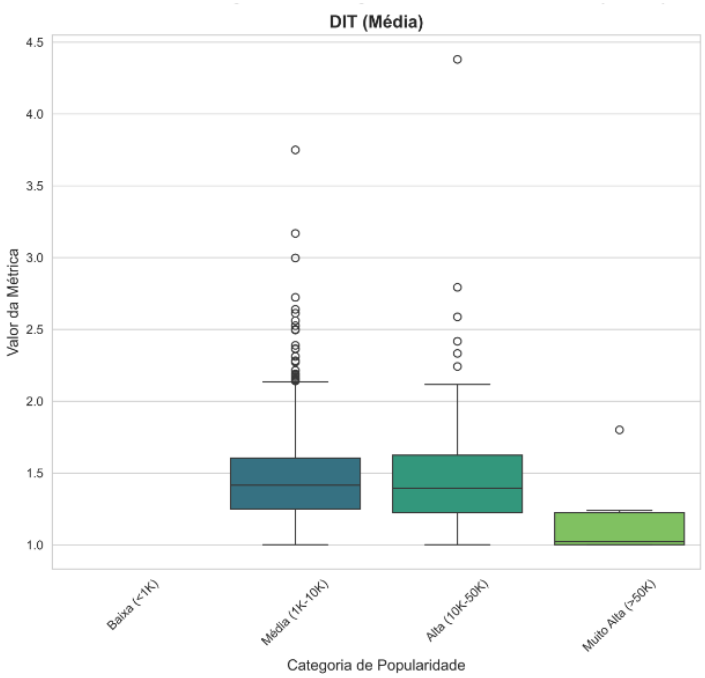

Figura 5: DIT x popularidade

Por último, analisamos o LCOM, que em português significa Falta de Coesão nos Métodos, mede o quão bem os métodos de uma classe trabalham juntos para cumprir um propósito único e bem definido. Nas nossas análises exibidas no boxplot abaixo, podemos também observar os níveis de LCOM sendo reduzidos à medida que a popularidade de um projeto aumenta, o que prova o que foi observado anteriormente, que repositórios mais populares tendem a ter uma arquitetura mais organizada, que são fáceis de serem entendidos e modificados, o que gera grande interesse por parte dos usuários para colaborações.

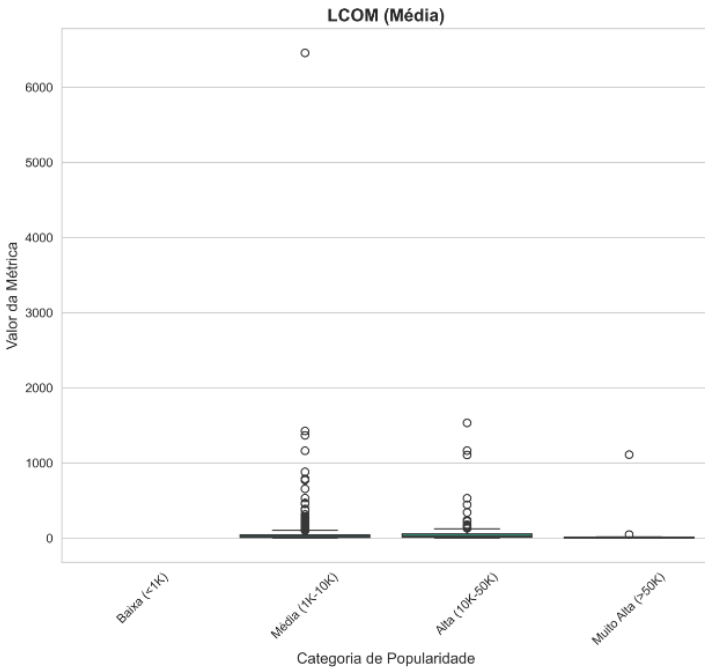

Figura 6: LCOM x popularidade

A análise dos dados revela uma forte correlação positiva entre a popularidade de um repositório Java e sua qualidade de código interna. Projetos mais populares consistentemente apresentam melhores métricas de acoplamento (CBO), herança (DIT) e coesão (LCOM).Essa relação sugere um ciclo virtuoso: uma arquitetura de software bem estruturada facilita a colaboração, o que atrai mais desenvolvedores e impulsiona a popularidade. Com mais contribuidores e escrutínio, a qualidade do projeto é continuamente refinada, um processo que se intensifica à medida que o projeto amadurece e se torna mais popular.

Em suma, investir em uma boa arquitetura desde o início é um fator crucial para a escalabilidade e o sucesso de um projeto open-source, pois a qualidade do código é um motor fundamental para o engajamento da comunidade.

2.2 Qual a relação entre a maturidade do repositórios e as suas características de qualidade ?

Agora analisaremos estas métricas com relação a maturidade dos repositorios, uma hipotese que temos de antemão, é que os resultados vão seguir a mesma direção dos resultados anteriores. A seguir, temos uma lista de gráficos, que comparam todas as métricas de qualidade CBO e DIT anteriormente com os seus atributos de maturidade, que definimos como a idade do repositório.

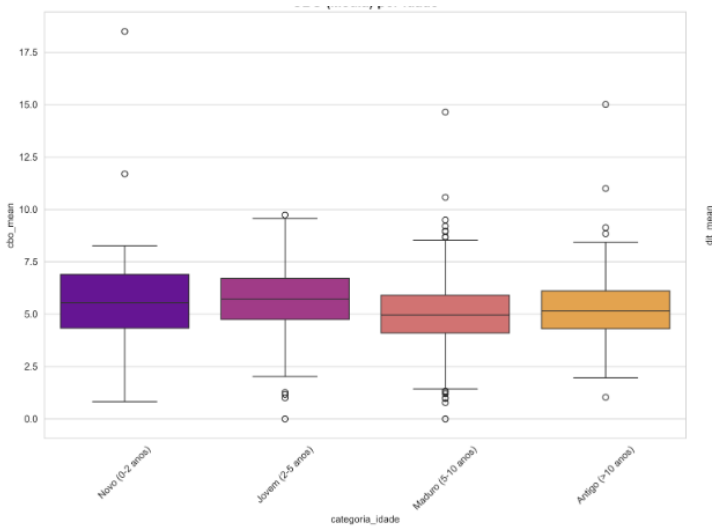

Figura 7: CBO x maturidade

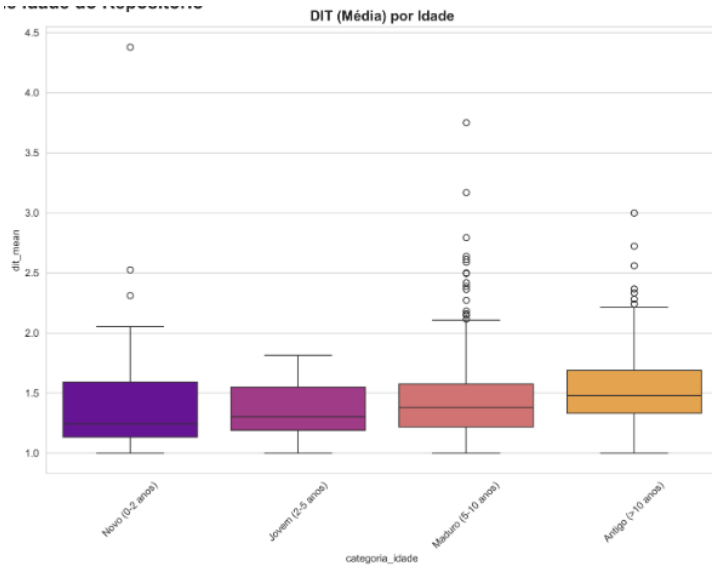

Figura 8: DIT x maturidade

Podemos identificar que, nestes dados coletados, a maturidade não influencia diretamente nos dados coletados, pois observamos valores que flutuam entre a mesma faixa: 5.0 a 5.2 no CBO e 1.0 e 1.5 no DIT. Isso pode nos mostrar que, por mais que repositórios possam ser maduros, sua arquitetura ainda tende a seguir um padrão de qualidade, provando esta não influência. Como nos casos apresentados, isso pode ser justificado com uma série de sentenças, como o fato de que a qualidade arquitetural de um projeto é frequentemente definida em sua fundação e se torna "congelada"pelo peso do código legado, enquanto projetos novos atingem alta qualidade rapidamente com ferramentas modernas, tendendo a se estabilizar ao longo do seu ciclo de vida.

2.3 Qual a relação entre a atividade dos repositórios e as suas características de qualidade?

Para comparar qualidade com a questão do nivel de atividade de um repositorio, levaremos em consideração métricas de CBO, e o numero de releases que o repositório possui, como mostrado no grafico abaixo. A partir de uma análise do mesmo, podemos observar que a maioria dos repositórios coletados

possui o número de releases em um intervalo com limite superior igual a 200, também podemos observar que estes repositórios possuem níveis diferentes de CBO, o que implica uma grande variação nos níveis de acoplamento entre estes repositórios, mas por mais diversificados que sejam, possuem uma leve tendência a aumentarem, à medida que o número de releases sobe, como mostra a linha tracejada, o que nos mostra que existe uma leve influência do número de releases nos níveis de CBO, o que pode ser justificado pelo fato de que com o passar das releases, os sistemas vão ficando mais complexos, e que se não existir uma gestão de arquitetura eficiente, a qualidade pode ser afetada. Isso nos mostra o quão importante a gestão arquitetural, revisões constantes, e aprimoramentos são em um sistema, evitando a deteriorização do mesmo, que acontece através de redução de qualidade, dívidas técnicas e etc.

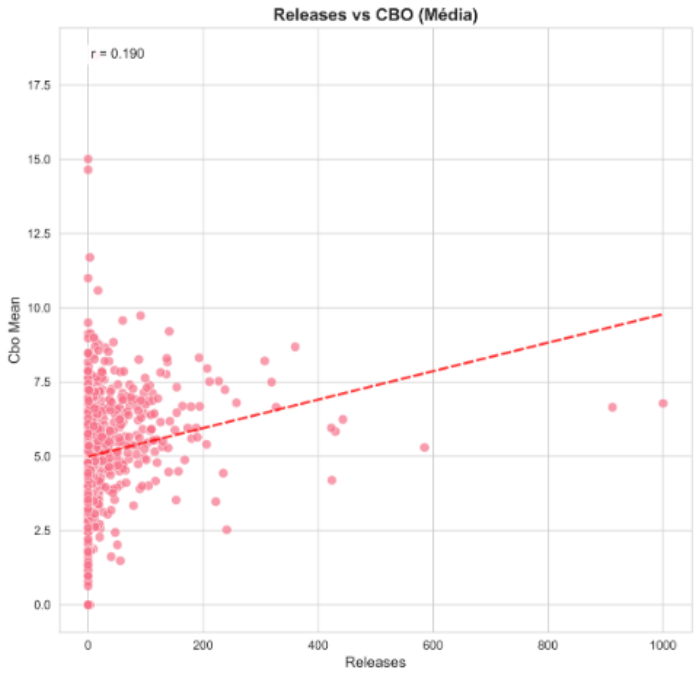

Figura 9: CBO x releases

2.4 Qual a relação entre o tamanho dos repositórios e as suas características de qualidade?

Neste tópico podemos entender como tamanho de repositório, a partir da métrica de LOC (lines of code), que conta as linhas do código, que foram em torno de 70 milhões, e do CBO, que é a métrica relacionada ao acoplamento entre os objetos, podemos entender que a partir das linhas de código, como mostrada no gráfico abaixo, uma relação decrescente entre o numero de linhas de código, e sua qualidade, podemos assim afirmar, que conforme as linhas de código vão sendo adicionadas, o acoplamento vai aumentando, prejudicando a qualidade do sistema, isso pode ocorrer devido a vários fatores, como um acompanhamento arquitetural não tão frequente, a modelagem/planejamento não tão forte durante o projeto e etc.

3 Considerações finais

A análise dos repositórios Java mais populares do GitHub revela uma correlação direta e consistente entre a popularidade de um projeto e sua qualidade de código interna. Observou-se que quanto maior o número de estrelas, melhores são os indicadores de acoplamento (CBO), profundidade de herança (DIT) e coesão (LCOM). Em contrapartida, a maturidade, definida pela idade do repositório, não demonstrou influenciar diretamente a qualidade, com as métricas permanecendo estáveis ao longo do tempo. Adicionalmente, um aumento na atividade, como o número de releases, mostrou uma leve

tendência de deteriorar a qualidade, aumentando o acoplamento se não houver uma gestão arquitetural eficiente.

Esses resultados sugerem a existência de um "ciclo virtuoso", onde uma arquitetura bem planejada atrai mais colaboradores, e o escrutínio dessa comunidade ajuda a manter e a refinar a qualidade do software. A ausência de uma correlação com a maturidade pode ser justificada por fatores como a importância da fundação arquitetural inicial, o "congelamento"da qualidade devido ao peso do código legado, e um viés de sobrevivência na amostra, que inclui apenas projetos já bem-sucedidos. Conclui-se que o engajamento da comunidade e a gestão ativa da arquitetura são mais determinantes para a qualidade do que o tempo de existência do projeto.

Portanto, as implicações práticas deste estudo são claras: investir em uma boa arquitetura desde o início é um fator crucial para a escalabilidade e o sucesso de um projeto open-source. Repositórios em fase de crescimento, especialmente na faixa de popularidade intermediária, devem estar atentos a "dores do crescimento", como a introdução de classes complexas que necessitam de refatoração. A manutenção da qualidade não é um resultado passivo do tempo, mas um esforço ativo que exige disciplina, revisões constantes e aprimoramentos para evitar a deterioração do sistema ao longo de seu ciclo de vida.

Segue abaixo uma matriz de correlação entre os dados obtidos neste estudo:

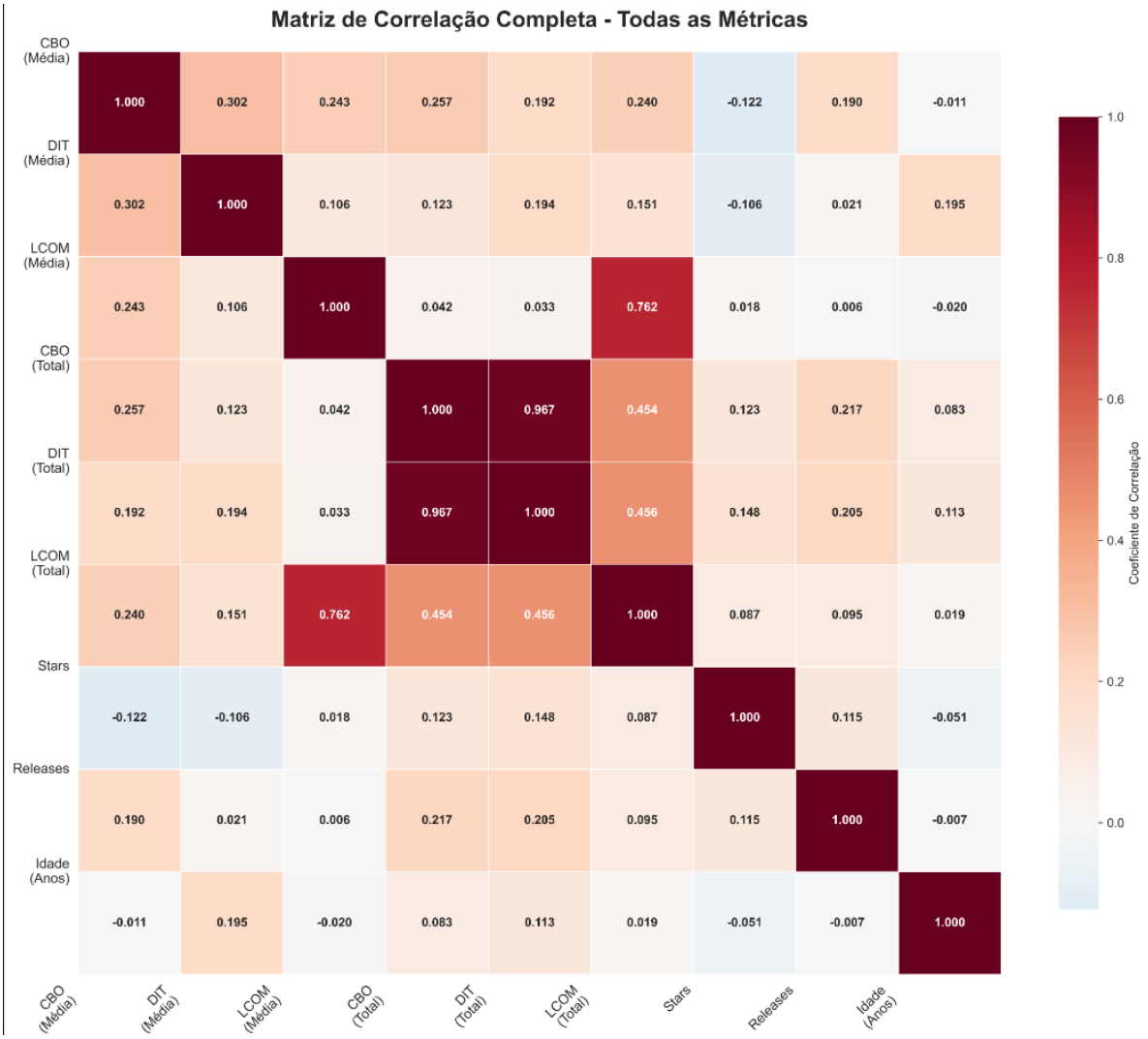

Figura 10: Matriz de correlação