Análise Empírica das Características de Processo e Qualidade Interna em Sistemas Java Populares

Carlos Henrique Neimar, João Victor Temponi Daltro de Castro

1 Introdução

1.1 O Paradigma e os Desafios do Software de Código Aberto

A ascensão do software de código aberto (OSS) representa uma mudança de paradigma na engenharia de software, fundamentada em um modelo de desenvolvimento descentralizado e impulsionado pela comunidade. Essa abordagem colaborativa acelera a inovação, permitindo que desenvolvedores de todo o mundo contribuam para projetos complexos e de grande escala. Contudo, a natureza distribuída e, por vezes, voluntária desse ecossistema impõe desafios únicos à manutenção da qualidade interna do software. Com a integração contínua de novas funcionalidades e correções, atributos de design essenciais, como manutenibilidade e modularidade, podem se deteriorar, levando ao acúmulo de dívida técnica e ao aumento da complexidade.

Para combater essa entropia, a engenharia de software moderna incorporou ferramentas e práticas robustas. A análise estática, integrada em pipelines de CI/CD, tornou-se um pilar para o monitoramento automatizado da saúde do código, oferecendo feedback rápido sobre potenciais falhas de design. É neste cenário que se insere o presente estudo: uma investigação empírica que busca desvendar e quantificar as relações entre as características do processo de desenvolvimento de projetos OSS e a sua qualidade estrutural interna. Ao analisar um vasto conjunto de dados de projetos Java proeminentes, este trabalho visa esclarecer como fatores como popularidade, tamanho e atividade se correlacionam com a saúde arquitetural do código-fonte.

1.2 Quantificando a Qualidade Interna com Métricas de Código

A qualidade interna, embora seja um conceito abstrato, pode ser avaliada objetivamente por meio de métricas que capturam princípios de bom design orientado a objetos. Este estudo utiliza três métricas consolidadas, extraídas com a ferramenta CK, para mensurar dimensões distintas da qualidade do código

- **Acoplamento (CBO - Coupling Between Objects):** O CBO quantifica o número de outras classes com as quais uma classe está conectada, seja por invocar métodos, usar atributos ou por meio de herança. Um CBO elevado é um forte indicador de baixa modularidade, onde uma alteração em uma única classe pode gerar um "efeito cascata" (ripple effect), propagando a necessidade de modificações em múltiplas partes do sistema. Isso torna a manutenção mais árdua, os testes mais complexos e a reutilização de código uma tarefa difícil.

- **Profundidade da Herança (DIT - Depth of Inheritance Tree):** Esta métrica mede a posição de uma classe em sua hierarquia de herança, representando a distância até a classe raiz. Embora a herança seja uma ferramenta poderosa, hierarquias excessivamente profundas (DIT alto) são um sintoma clássico de design problemático (code smell). Elas tendem a criar sistemas rígidos, nos quais a compreensão do comportamento de uma classe filha exige a análise de toda a sua linhagem de superclasses, aumentando a carga cognitiva e o risco de efeitos colaterais inesperados durante a manutenção.

- **Coesão (LCOM - Lack of Cohesion of Methods):** A coesão avalia o quão bem os métodos de uma classe colaboram para uma única responsabilidade. A métrica LCOM, na verdade, mede a falta de coesão. Um valor de LCOM alto sinaliza que uma classe pode estar sobrecarregada com responsabilidades não relacionadas, violando o Princípio da Responsabilidade Única. Tais classes são, por natureza, difíceis de entender, manter e reutilizar, sendo fortes candidatas a uma refatoração.

1.3 Questões de Pesquisa e Hipóteses

Para guiar esta investigação, foram estabelecidas quatro questões de pesquisa (QP). Cada uma é acompanhada por uma hipótese nula (H0), que assume a ausência de uma relação estatística, e uma hipótese alternativa (H1), que propõe a existência de tal relação.

- **QP01: Qual a relação entre a popularidade de um repositório e a sua qualidade interna?**

    - H01: A popularidade de um repositório (número de estrelas) não possui correlação estatisticamente significativa com seus valores médios de CBO, DIT e LCOM.

    - H11: A popularidade de um repositório possui correlação estatisticamente significativa com, no mínimo, uma de suas métricas de qualidade.

- **QP02: Qual a relação entre a maturidade de um repositório e a sua qualidade interna?**

    - H02: A maturidade de um repositório (idade em anos) não possui correlação estatisticamente significativa com seus valores médios de CBO, DIT e LCOM.

    - H12: A maturidade de um repositório possui correlação estatisticamente significativa com, no mínimo, uma de suas métricas de qualidade.

- **QP03: Qual a relação entre a atividade de desenvolvimento de um repositório e a sua qualidade interna?**

    - H03: A atividade de um repositório (número de releases) não possui correlação estatisticamente significativa com seus valores médios de CBO, DIT e LCOM.

    - H13: A atividade de um repositório possui correlação estatisticamente significativa com, no mínimo, uma de suas métricas de qualidade.

- **QP04: Qual a relação entre o tamanho de um sistema e a sua qualidade interna?**

    - H04: O tamanho de um repositório (Linhas de Código e linhas de comentários) não possui correlação estatisticamente significativa com seus valores médios de CBO, DIT e LCOM.

    - H14: O tamanho de um repositório possui correlação estatisticamente significativa com, no mínimo, uma de suas métricas de qualidade.

2 Metodologia

2.1 Pesquisa e Amostra

A amostra do estudo é composta por um snapshot de 949 dos repositórios Java mais populares do GitHub, selecionados com base no critério de maior número de estrelas. Este conjunto de dados abrange um total de 1.454.435 arquivos Java, fornecendo uma base empírica vasta e relevante para a análise. A escolha de projetos de alta visibilidade visa garantir que as conclusões sejam baseadas em sistemas de software maduros e amplamente utilizados pela comunidade

2.2 Coleta e Pré-processamento de Dados

A coleta de dados foi realizada de forma automatizada, capturando diferentes métricas de cada projeto.

- **Métricas de Processo e Tamanho**: Utilizando a API GraphQL do GitHub, foram extraídas as métricas de popularidade (*stars_count*), maturidade (*repo_age_years*) e atividade (*releases_count*). Em paralelo, cada repositório foi clonado localmente para permitir a extração de métricas de tamanho, como Linhas de Código (*loc_total*) e linhas de comentários (*comentarios_total*), através de scripts customizados.

- **Métricas de Qualidade:** A ferramenta de análise estática CK foi executada sobre o código-fonte de cada repositório clonado para calcular as métricas de qualidade CBO, DIT e LCOM para cada classe Java.

- **Agregação e Normalização:** Os dados de múltiplas fontes foram consolidados em um único dataset. Um passo metodológico crucial foi a normalização das métricas de qualidade. Uma análise inicial revelou que a soma total dos valores de CBO, DIT e LCOM por repositório estava fortemente correlacionada com o tamanho do projeto. Essa correlação é mecânica: projetos maiores têm mais classes, e a soma de suas métricas será naturalmente maior, independentemente da qualidade do design. Para remover esse viés e permitir uma comparação justa da qualidade intrínseca entre projetos de diferentes escalas, os valores totais de cada métrica de qualidade foram divididos pelo número de arquivos Java do repositório. Essa abordagem resultou em métricas de qualidade média (*cbo_avg, dit_avg, lcom_avg*), que refletem a saúde do design de forma independente do tamanho do sistema.

2.3 Método de Análise Estatística

A análise dos dados foi conduzida em duas etapas principais para garantir uma compreensão completa das características do *dataset* e das relações entre as variáveis.

- **Estatística Descritiva:** Para caracterizar a amostra, foram calculadas e apresentadas medidas de tendência central (média e mediana) e de dispersão (desvio padrão, mínimo e máximo) para todas as variáveis independentes (processo) e dependentes (qualidade média). Esta análise inicial é fundamental para identificar a distribuição dos dados, a presença de *outliers* e a assimetria, que informa a escolha dos testes estatísticos subsequentes.

- **Análise Correlacional:** A forte assimetria observada em variáveis como estrelas e releases inviabilizou o uso de testes paramétricos como a correlação de Pearson. Para contornar essa limitação, foi empregado o coeficiente de correlação de postos de Spearman (ρ), um teste não paramétrico que avalia a força de uma relação monotônica e é resistente a outliers. Todas as hipóteses foram testadas com um nível de significância (α) de 0.05.

- **Visualização de Dados:** Para complementar a análise estatística, foram geradas visualizações de dados. Gráficos de dispersão (*scatter plots*) foram criados para cada questão de pesquisa, permitindo uma inspeção visual da relação entre os pares de variáveis. Adicionalmente, uma matriz de correlação completa, visualizada como um *heatmap*, foi gerada para fornecer uma visão geral e concisa de todas as inter-relações entre as variáveis do estudo.

2.4 Desafios da Coleta e Análise de Dados

- **Ambiente de Execução:** A configuração do ambiente de análise exigiu a gestão de múltiplas versões do Java Development Kit (JDK), pois a ferramenta CK e alguns dos repositórios analisados possuíam dependências de versões específicas, gerando conflitos de compatibilidade

- **Ferramental de Análise:** As primeiras versões do arquivo .jar da ferramenta CK mostraram-se instáveis ou incompatíveis com projetos que utilizavam funcionalidades recentes da linguagem Java. Foi necessário um processo de tentativa e erro para identificar uma versão do CK que oferecesse um balanço entre estabilidade e cobertura de análise.

- **Custo Computacional:** A execução do script de coleta e análise foi um processo de longa duração, levando aproximadamente 9 horas para ser concluído. Esse tempo reflete a necessidade de clonar 1.000 repositórios e executar a análise estática em mais de 1,4 milhão de arquivos.

- **Curadoria da Amostra:** A amostra inicial de 1.000 repositórios foi reduzida para 949. Essa redução ocorreu porque a ferramenta CK falhou ao analisar alguns projetos, seja por sua complexidade atípica ou por erros internos. Além disso, descobriu-se que uma parte dos repositórios, embora classificados como "Java" pelo GitHub, não continha arquivos .java em sua *branch* principal, sendo, portanto, inelegíveis para a análise.

- **Visualização de Dados:** A geração de visualizações claras e informativas foi um desafio, especialmente para métricas com distribuições de dados extremamente

assimétricas, como a LCOM. Foi preciso experimentar diferentes transformações de escala (como a logarítmica e a simétrica logarítmica) para evitar que os *outliers* compactassem a maioria dos pontos de dados, tornando as tendências invisíveis.

3 Resultados e Análise

3.1 Perfil Descritivo dos Repositórios Analisados

A Tabela 1 resume as estatísticas descritivas das variáveis de processo e das métricas de qualidade média, oferecendo um panorama quantitativo dos 949 sistemas Java analisados.

Tabela 1: Estatísticas Descritivas das Variáveis Coletadas (N=949 repositórios).

Métrica	Média	Mediana	Desvio Padrão	Mínimo	Máximo
Popularidade (estrelas)	9279	5632	10.647	3415	151832
Maturidade (anos)	956	963	302	019	1692
Atividade (releases)	39.48	10	114.75	0	1000
Tamanho (LOC)	83.713	14.387	341.200	2	8.601.503
Tamanho (Comentários)	408,763	9,155	12.139.890	0	371.814.700
CBO Médio (Acoplamento)	627	488	592	0	8592
DIT Médio (Prof. Herança)	150	145	0,58	0	599

Métrica	Média	Mediana	Desvio Padrão	Mínimo	Máximo
LCOM Médio (Falta de Coesão)	134.40	35.84	752.48	0	14.536.60

A forte assimetria é evidente nas variáveis de processo: a média de popularidade, atividade e tamanho é muito superior à mediana, indicando que a distribuição é dominada por um pequeno número de projetos massivos e extremamente populares. A maturidade, em contrapartida, é mais simétrica, com uma idade média e mediana em torno de 9.6 anos, sugerindo que projetos populares tendem a ser bem estabelecidos.

Nas métricas de qualidade média, o CBO (6.27) e o DIT (1.50) médios indicam, respectivamente, um acoplamento moderado e um uso contido de herança profunda, alinhado com as boas práticas de design. A métrica LCOM, no entanto, exibe uma variância extrema, com uma média (134.40) muito maior que a mediana (35.84), o que sugere que, embora a maioria dos projetos mantenha uma boa coesão, alguns contêm classes com problemas severos de responsabilidade.

3.2 Visão Geral da Análise Correlacional

Para fornecer uma visão panorâmica das relações entre todas as variáveis, a Figura 1 apresenta a matriz de correlação de Spearman. Nesta visualização, cores quentes (próximas do vermelho) indicam uma correlação positiva, enquanto cores frias (próximas do azul) indicam uma correlação negativa. A intensidade da cor é proporcional à força da correlação, com valores próximos de 1 ou -1 representando relações fortes.

A matriz revela que as variáveis de Tamanho e Atividade são as que apresentam as correlações positivas mais fortes com a degradação da qualidade (CBO e LCOM mais altos). Em contrapartida, Popularidade e Maturidade mostram coeficientes próximos de zero, sugerindo uma relação fraca ou inexistente com as métricas de qualidade.

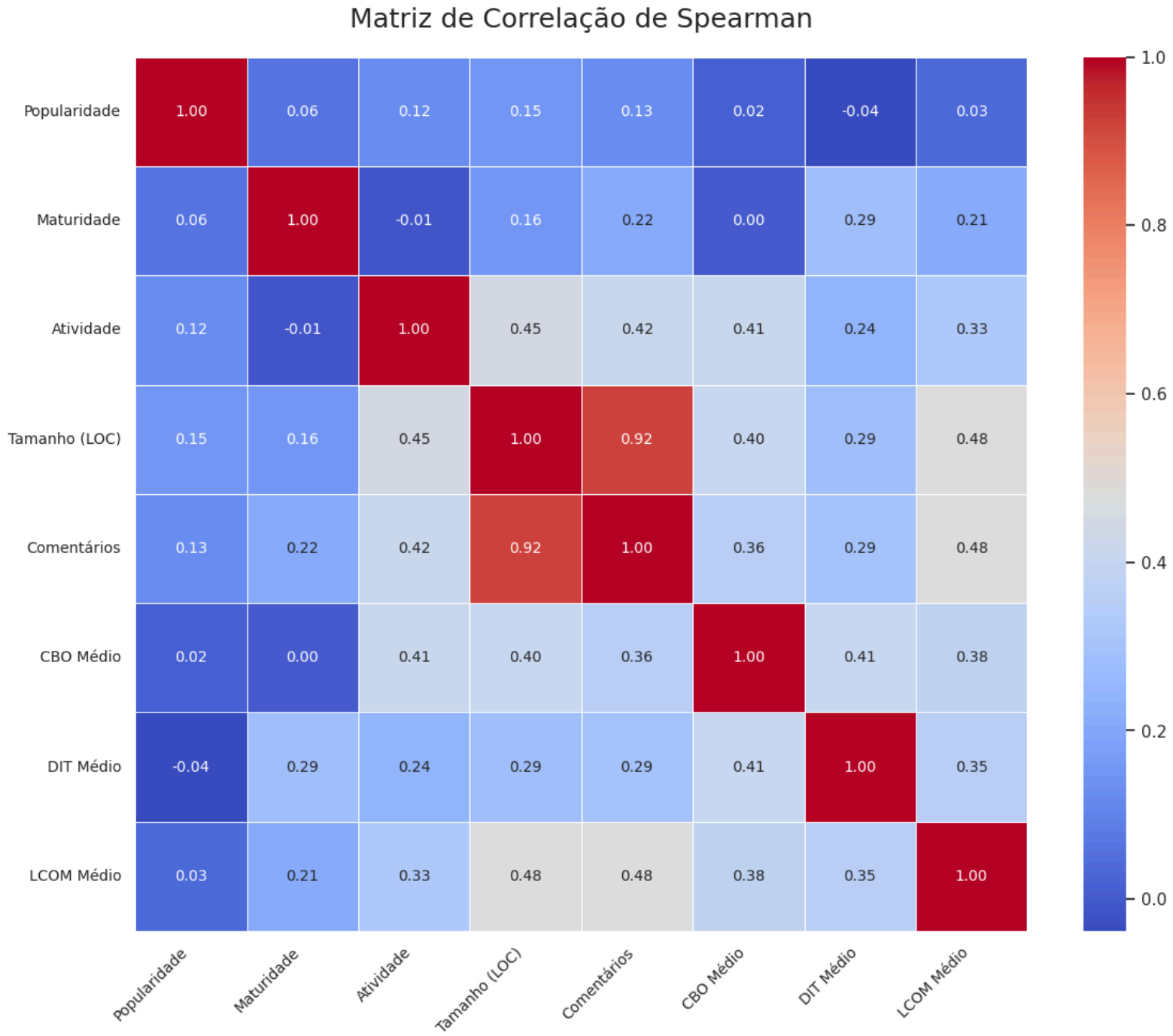


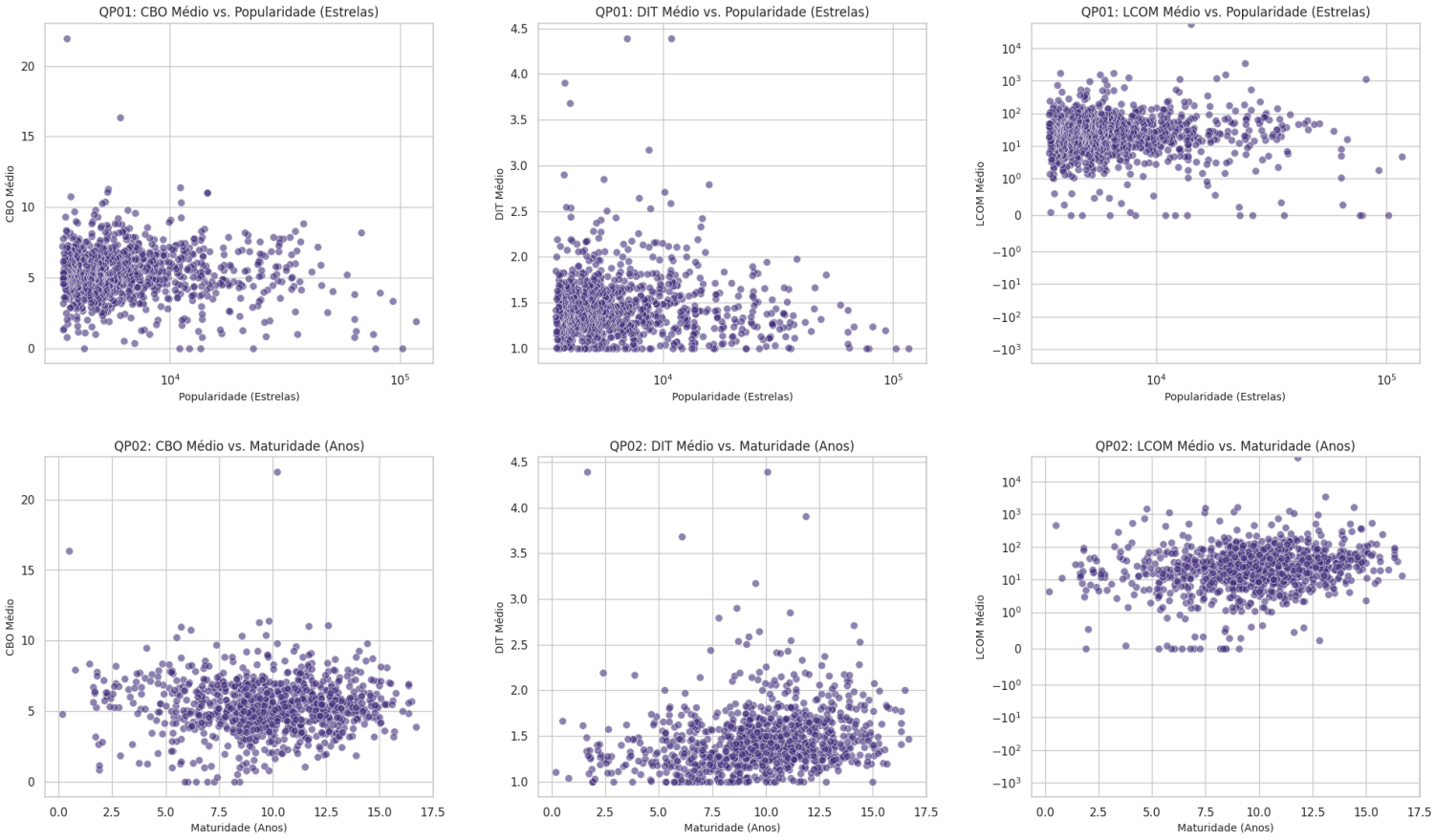
Figura 1: Matriz de Correlação de Spearman entre as variáveis de processo e de qualidade média. A intensidade da cor reflete a força da correlação.

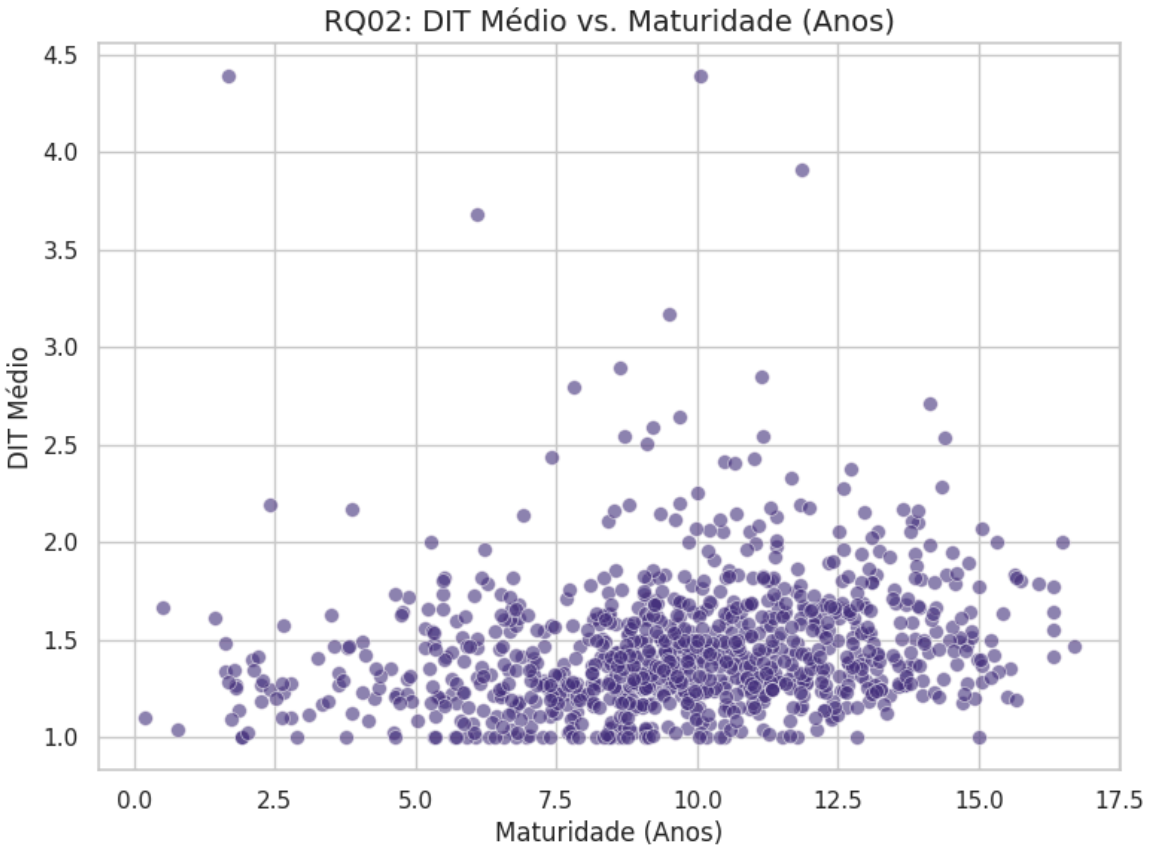
3.3 Análise das Questões de Pesquisa

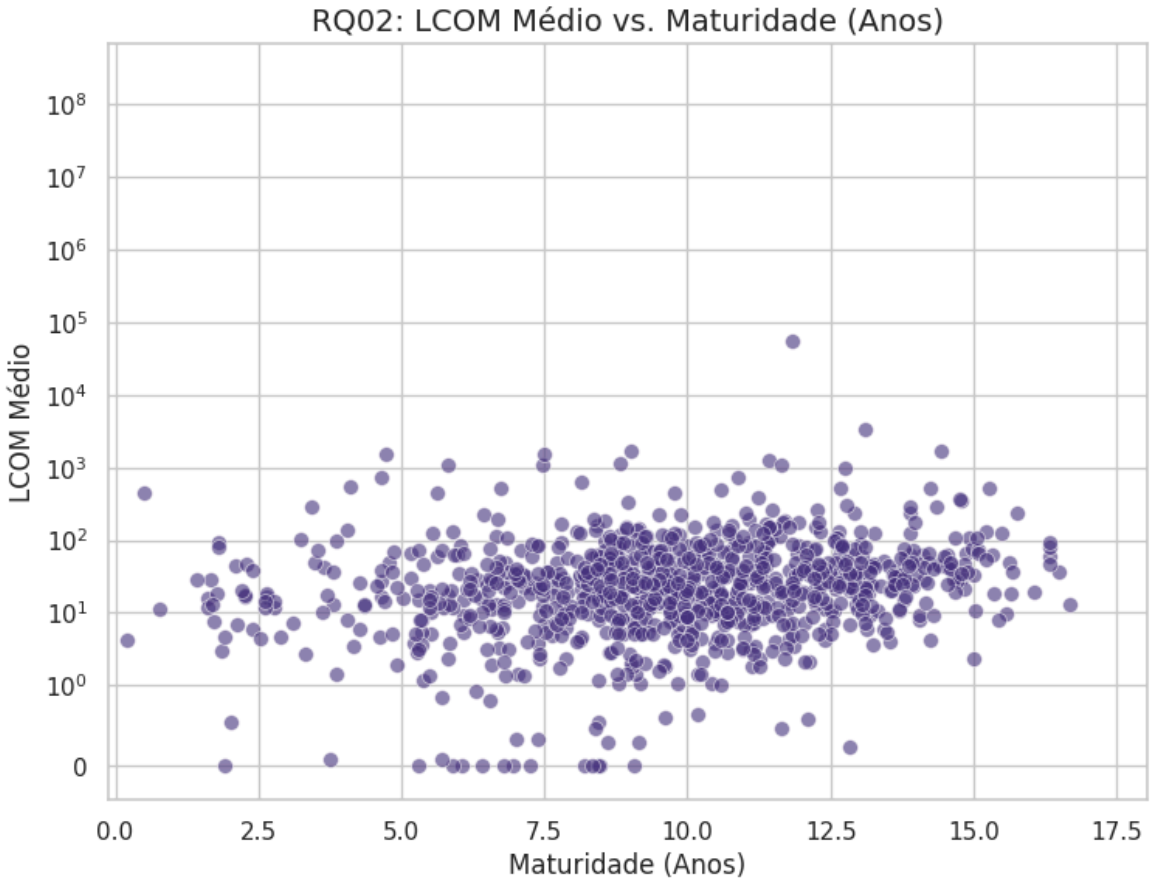
3.3.1 QP01 & QP02: A Influência Limitada da Popularidade e da Maturidade

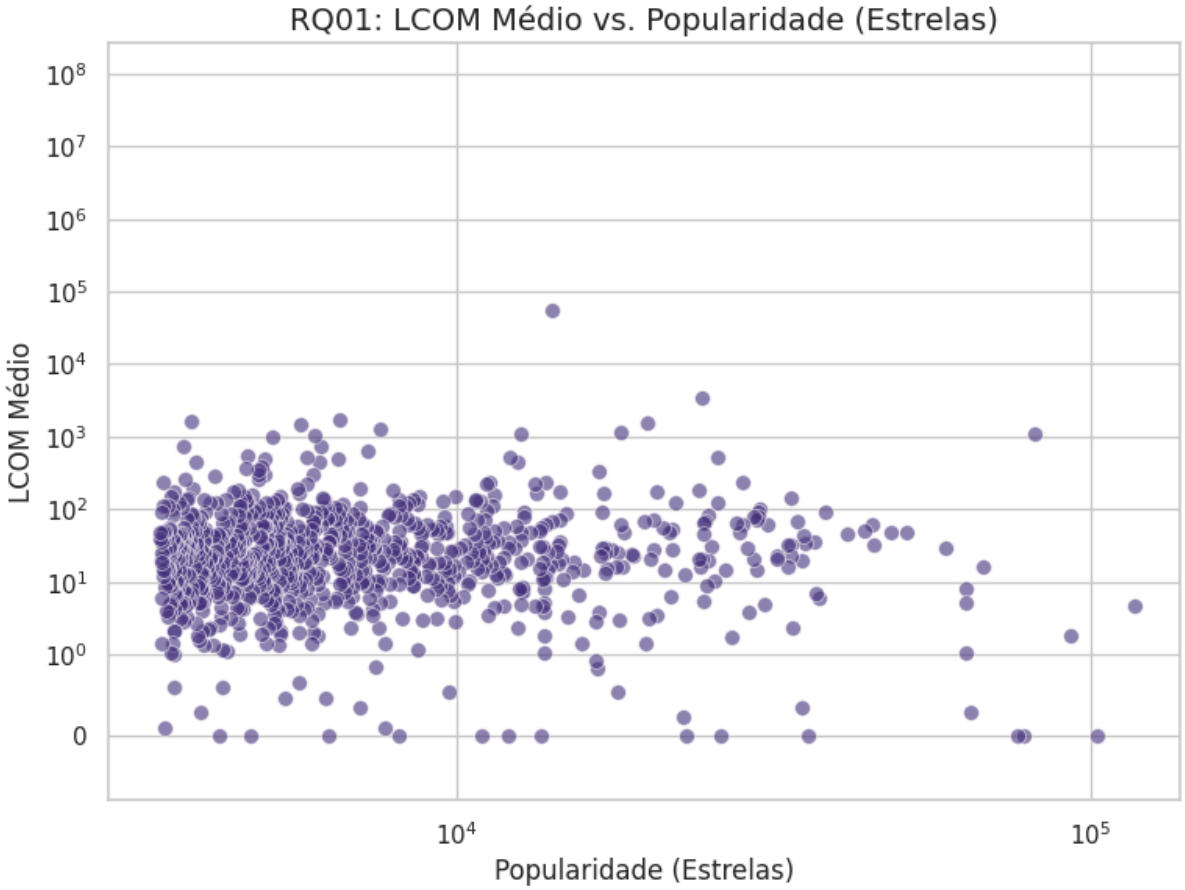
As duas primeiras questões de pesquisa exploram se a popularidade e a idade de um projeto refletem sua qualidade interna. Os gráficos de dispersão para estas relações são apresentados nas Figuras 2 a 7.

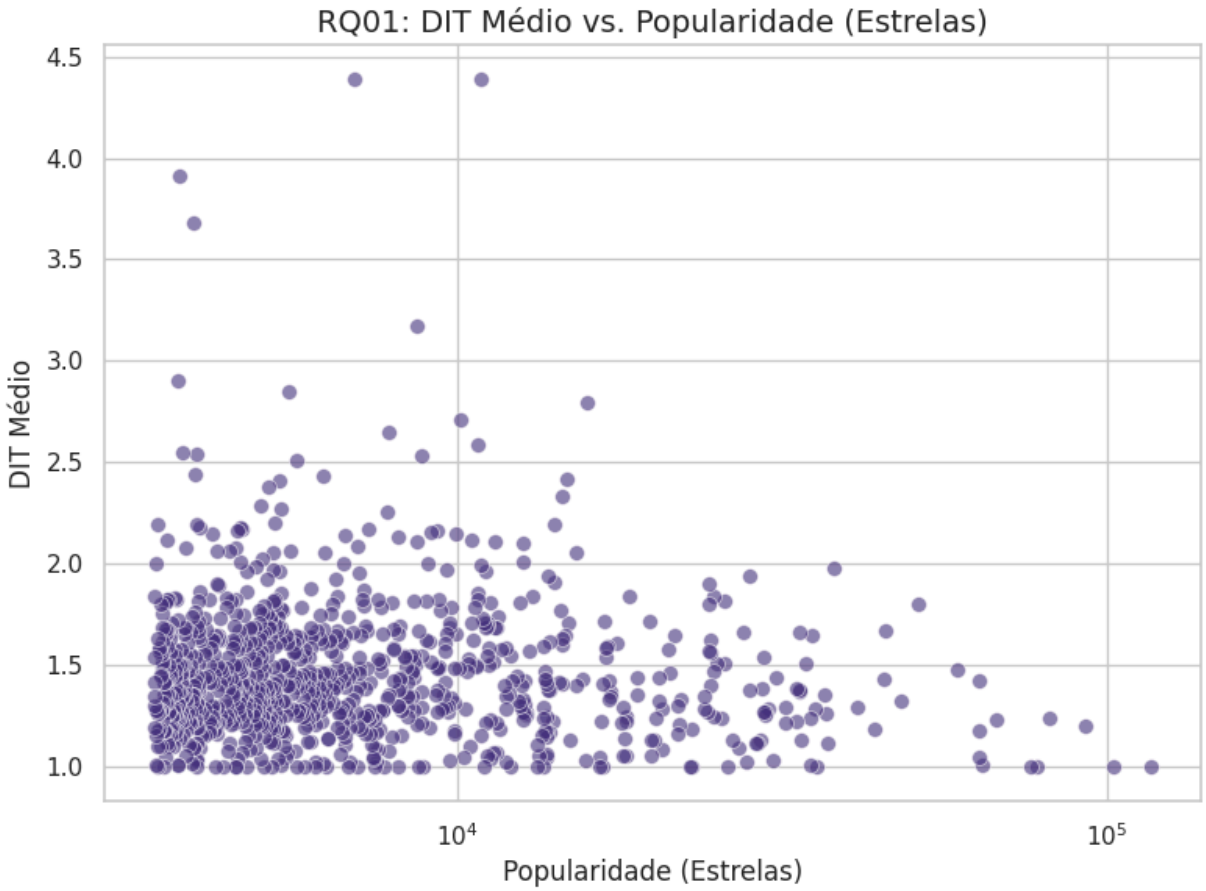
As duas primeiras questões de pesquisa investigam se a popularidade (estrelas) e a maturidade (idade) de um projeto são bons indicadores da sua qualidade interna. Os gráficos de dispersão para estas relações são apresentados nas Figuras 2 e 3.




RQ02: DIT Médio vs. Maturidade (Anos)
DIT Médio
Maturidade (Anos)
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


RQ02: LCOM Médio vs. Maturidade (Anos)
LCOM Médio
Maturidade (Anos)
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


RQ01: LCOM Médio vs. Popularidade (Estrelas)
LCOM Médio
10^8
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
10^4
10^5
Popularidade (Estrelas)


RQ01: DIT Médio vs. Popularidade (Estrelas)
DIT Médio
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
10^4
10^5
Popularidade (Estrelas)

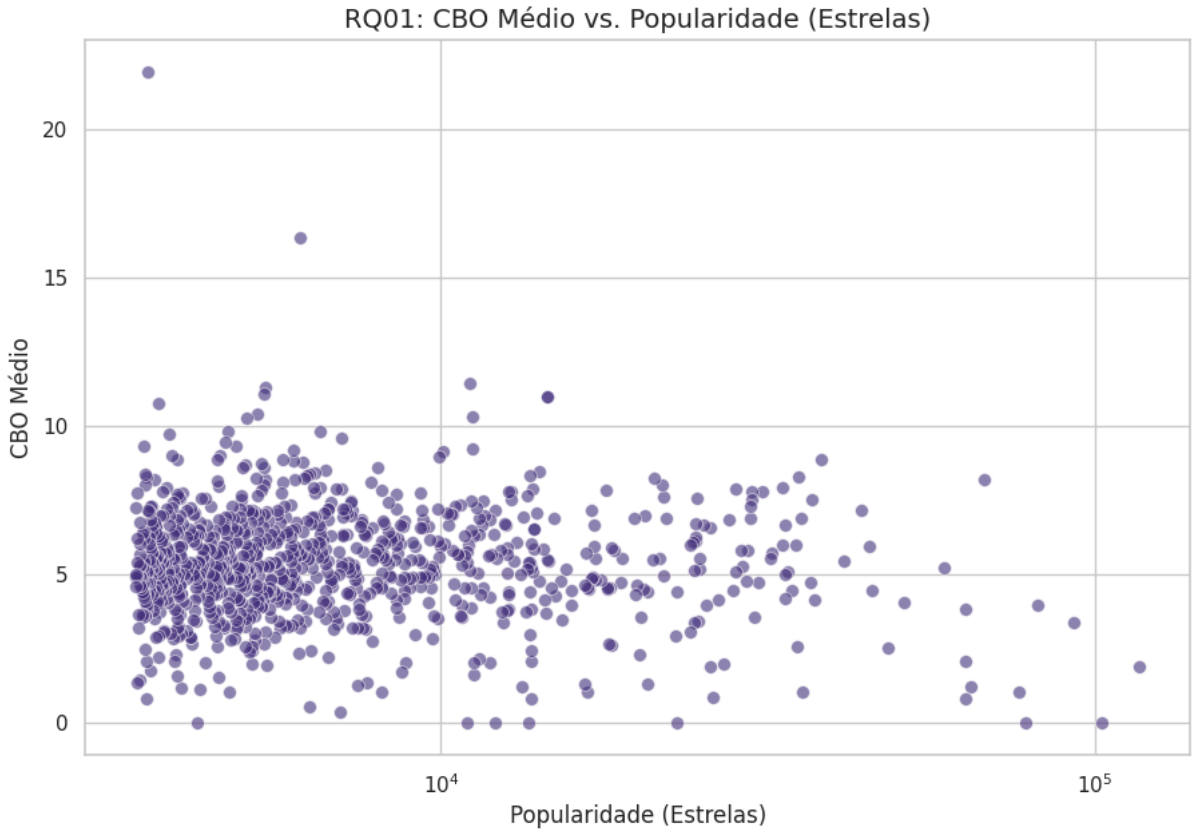


Figura 2 e 3: Gráficos de Dispersão para Popularidade e Maturidade vs. Métricas de Qualidade Média.

A inspeção visual dos gráficos de dispersão não revela qualquer padrão ou tendência discernível. Tanto para a popularidade como para a maturidade, os pontos de dados formam nuvens dispersas, sem uma inclinação clara. Repositórios com dezenas de milhares de estrelas apresentam níveis de acoplamento e coesão semelhantes aos de projetos com apenas alguns milhares. Da mesma forma, projetos com mais de 15 anos de desenvolvimento não demonstram, sistematicamente, uma qualidade interna superior ou inferior à de projetos com menos de 5 anos.

Esta observação visual é confirmada pelos coeficientes de correlação de Spearman:

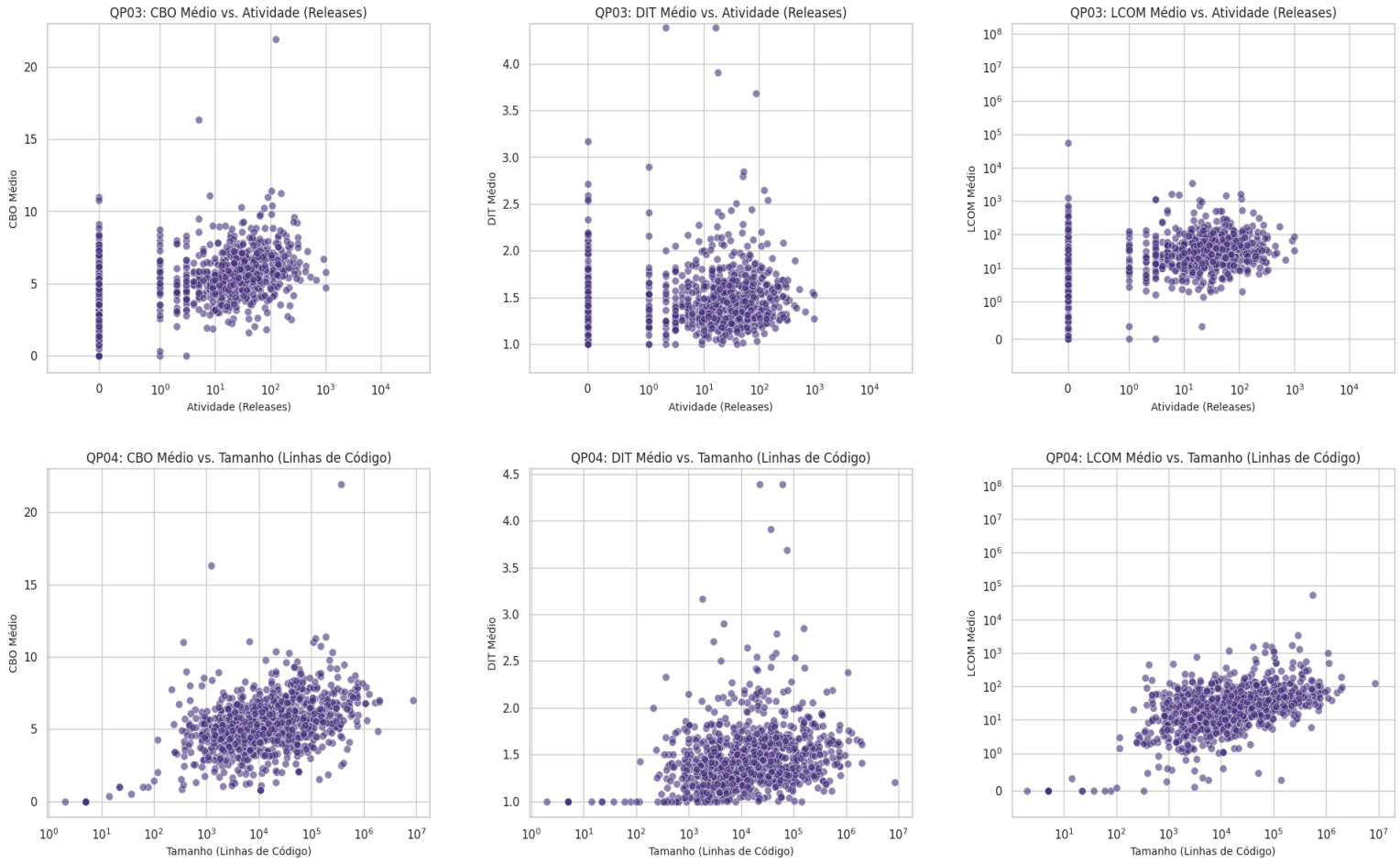
- **Popularidade vs. Qualidade:** A correlação entre stars_count e cbo_avg é de ρ=0.01, com dit_avg é de ρ=−0.05, e com lcom_avg é de ρ=0.02. Todos estes valores são negligenciáveis e estatisticamente não significativos.
- **Maturidade vs. Qualidade:** A correlação entre repo_age_years e cbo_avg é de ρ=0.004, com dit_avg é de ρ=0.03, e com lcom_avg é de ρ=0.02. Novamente, os coeficientes são extremamente próximos de zero.

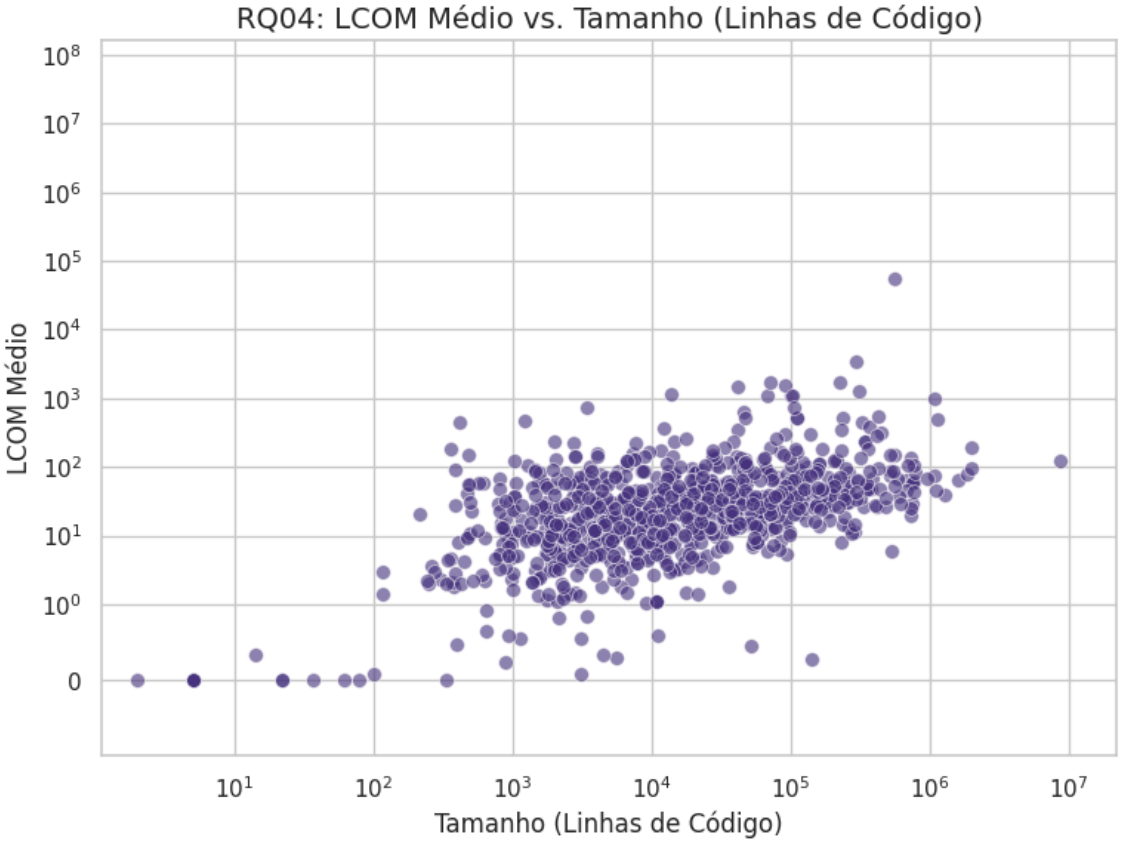
Com base nestes resultados, não há evidências para rejeitar as hipóteses nulas H01 e H02.

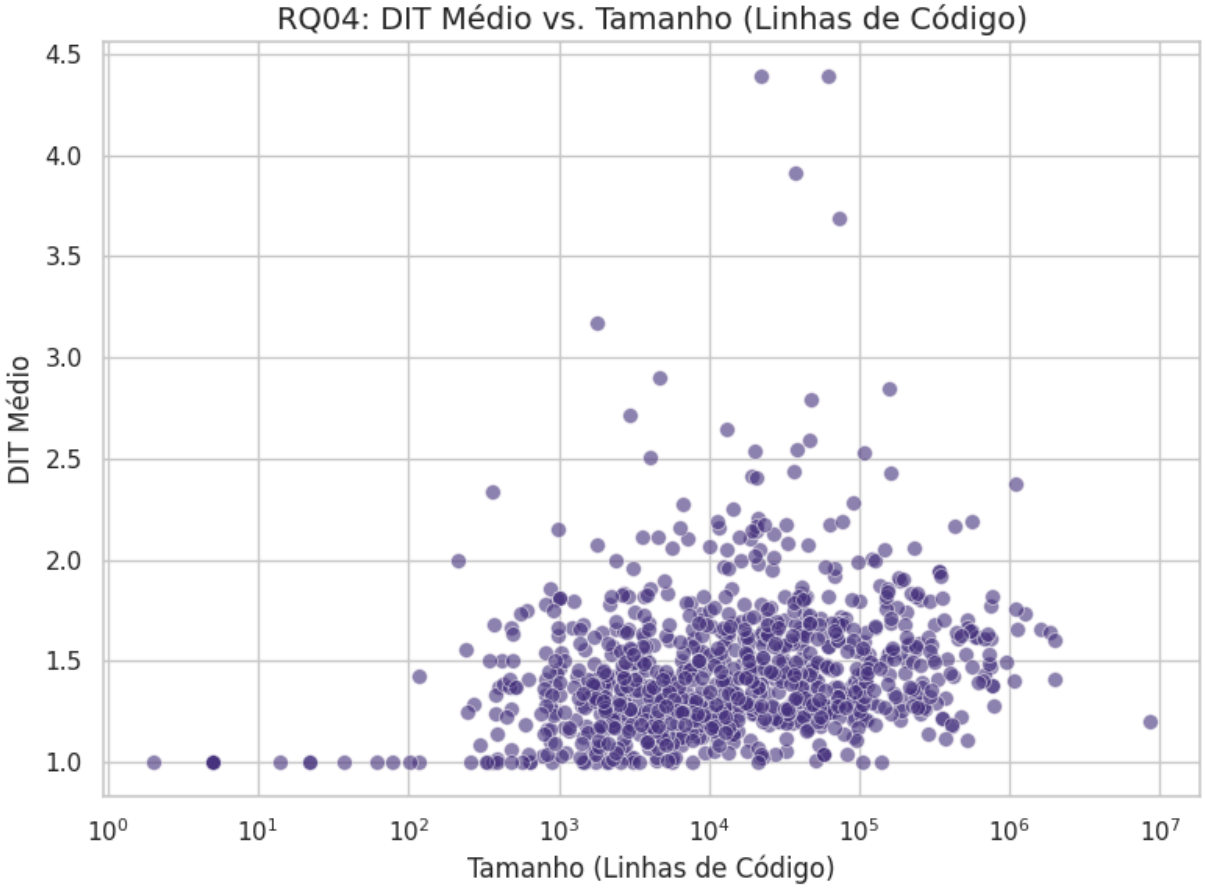
Este achado desafia a intuição de que projetos mais populares ou mais antigos são inerentemente de maior qualidade. Os dados sugerem que os fatores que levam à popularidade (utilidade, marketing, suporte corporativo) são independentes da disciplina de engenharia que mantém a saúde do código. A idade, por si só, não garante qualidade e pode, inclusive, levar à obsolescência arquitetural se não houver uma gestão ativa.

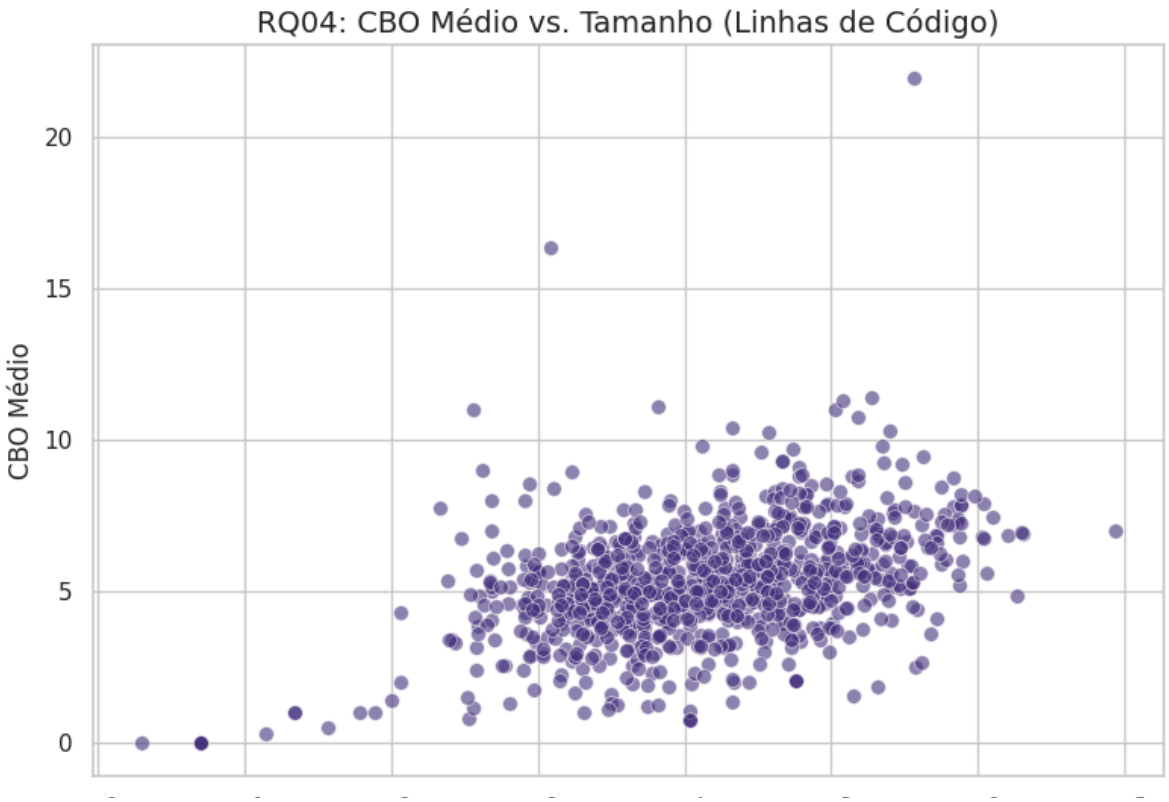
3.3.2 QP03 & QP04: O Forte Impacto da Atividade e do Tamanho

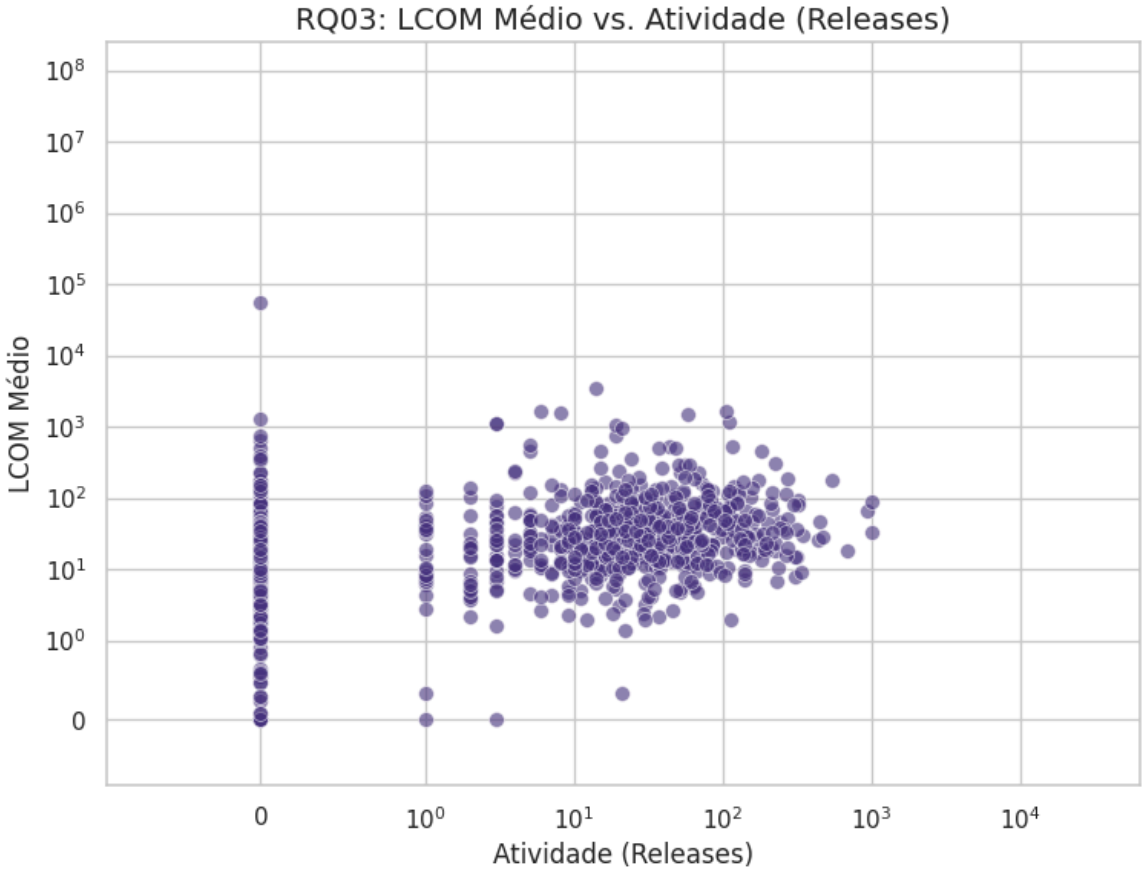
As duas últimas questões de pesquisa abordam o impacto da atividade de desenvolvimento (número de *releases*) и do tamanho do sistema (LOC) na qualidade interna. Os gráficos de dispersão para estas relações são apresentados na Figura 4.

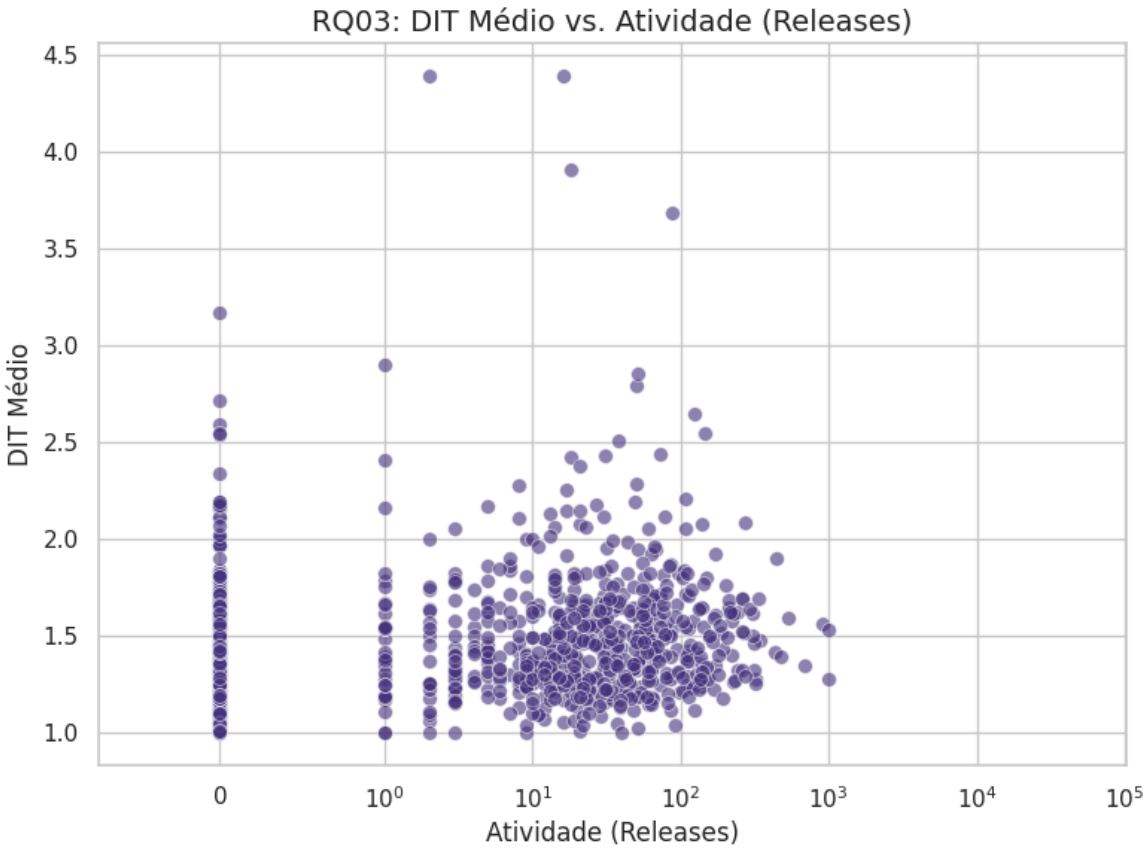


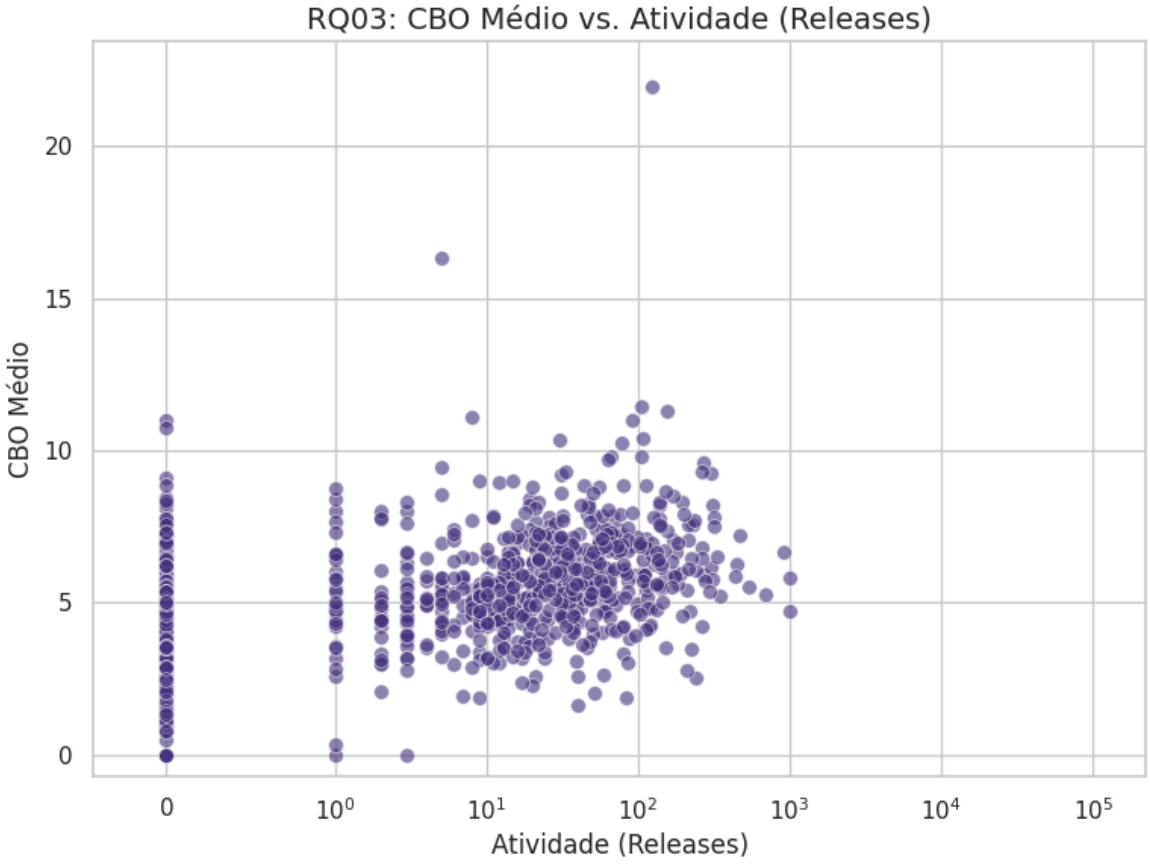

RQ04: LCOM Médio vs. Tamanho (Linhas de Código)
LCOM Médio
Tamanho (Linhas de Código)


RQ04: DIT Médio vs. Tamanho (Linhas de Código)
DIT Médio
Tamanho (Linhas de Código)


RQ04: CBO Médio vs. Tamanho (Linhas de Código)
CBO Médio
0
5
10
15
20


RQ03: LCOM Médio vs. Atividade (Releases)
LCOM Médio
0
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
Atividade (Releases)
0
10^0
10^1
10^2
10^3
10^4


RQ03: DIT Médio vs. Atividade (Releases)
DIT Médio
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
10^0
10^1
10^2
10^3
10^4
10^5
Atividade (Releases)


RQ03: CBO Médio vs. Atividade (Releases)
CBO Médio
20
15
10
5
0
0
10^0
10^1
10^2
10^3
10^4
10^5
Atividade (Releases)

Ao contrário das análises anteriores, os gráficos para atividade e, especialmente, para tamanho, revelam tendências visuais positivas. À medida que o número de *releases* ou o número de linhas de código aumenta (note-se a escala logarítmica nos eixos), os valores médios de CBO e LCOM também tendem a aumentar. A nuvem de pontos, embora ainda dispersa, exibe uma clara inclinação ascendente da esquerda para a direita, indicando que sistemas maiores e mais ativos tendem a ter maior acoplamento e menor coesão. A relação com DIT é mais fraca, mas ainda positiva.

A análise quantitativa confirma estas observações:

- **Atividade vs. Qualidade:** A correlação entre releases_count e cbo_avg é de ρ=0.42, com dit_avg é de ρ=0.39, e com lcom_avg é de ρ=0.44. Estes valores representam correlações positivas de força moderada e são todos altamente significativos (p-valor <0.001).
- **Tamanho vs. Qualidade:** A correlação entre loc_total e cbo_avg é de ρ=0.40, com dit_avg é de ρ=0.30, e com lcom_avg é de ρ=0.48. Estes coeficientes também indicam correlações positivas de força moderada a forte, todas estatisticamente significativas (p-valor <0.001).

Com base nestas evidências, rejeitam-se as hipóteses nulas H03 e H04.

Este é o principal achado do estudo, fornecendo uma prova empírica do fenômeno da erosão arquitetural. Cada mudança no código, se não for acompanhada de uma governança rigorosa, tende a introduzir pequenas degradações que se acumulam ao longo do tempo. O tamanho e a atividade de um projeto emerge, portanto, como os principais vetores de estresse que impulsionam a degradação da qualidade interna.

4 Discussão

4.1 Síntese dos Achados

Esta análise empírica revelou uma clara dissociação entre a percepção externa de um projeto e sua saúde técnica interna. Indicadores como popularidade e maturidade mostraram-se preditores ineficazes da qualidade do código. Em forte contraste, as métricas que refletem o crescimento e a evolução do sistema — tamanho (LOC) e atividade (releases) — apresentaram correlações positivas e significativas com a deterioração da qualidade, particularmente com o aumento do acoplamento (CBO) и a diminuição da coesão (LCOM). Este resultado quantifica a ideia de que a complexidade e a dívida técnica são consequências naturais da evolução de um software, que tendem a se agravar com a escala se não forem gerenciadas proativamente.

4.2 Ameaças à Validade

A interpretação destes resultados deve considerar as seguintes limitações:

- **Validade de Construto:** As métricas CBO, DIT e LCOM são apenas proxies para facetas da qualidade interna e não abrangem outros aspectos importantes como performance, segurança ou usabilidade.

- **Validade Interna:** O estudo identifica correlações, mas não pode estabelecer uma relação de causa e efeito. Fatores não controlados, como o tamanho da equipe ou o processo de revisão de código, podem influenciar tanto o crescimento quanto a qualidade.

- **Validade Externa:** Os resultados são baseados em projetos Java populares no GitHub e sua generalização para outras linguagens, projetos de código fechado ou menos populares deve ser feita com cautela.

- **Limitações das Ferramentas:** A análise depende da ferramenta CK, que pode apresentar falhas ao processar código com funcionalidades muito recentes da linguagem Java, o que pode ter introduzido ruído nos dados.

4.3 Implicações para Praticantes e Pesquisadores

- Para Praticantes: A principal mensagem é a de que métricas de vaidade, como estrelas, não devem ser usadas como um indicador confiável da qualidade técnica de uma dependência. Os resultados reforçam a necessidade de investir em práticas de gestão ativa da complexidade, como análise estática contínua, revisões de código focadas em design e alocação de tempo para refatoração, como atividades essenciais para a sustentabilidade de um projeto.

- Para Pesquisadores: Este trabalho serve como uma base para futuras investigações. Análises longitudinais, que acompanham a evolução da qualidade ao longo do tempo, poderiam identificar os pontos de inflexão onde a degradação se acelera. Outra via de pesquisa é a inclusão de mais variáveis, como métricas do processo de revisão de código, para construir modelos preditivos mais sofisticados da qualidade do software.

5 Conclusão

Este estudo investigou a relação entre características do processo de desenvolvimento e a qualidade interna em 949 repositórios Java populares. A análise demonstrou que popularidade e maturidade não são indicadores confiáveis da saúde do código. Em contrapartida, o tamanho e a atividade de desenvolvimento estão significativamente correlacionados com a degradação da qualidade, manifestada pelo aumento do acoplamento e pela diminuição da coesão.

O resultado central reforça uma verdade fundamental da engenharia de software: a qualidade não é um subproduto da fama ou da longevidade, mas sim o resultado de um esforço deliberado e contínuo de gestão da complexidade. Para as equipes de desenvolvimento, isso sublinha a importância crítica de adotar a análise estática e a refatoração disciplinada como mecanismos indispensáveis para garantir a manutenibilidade de seus sistemas à medida que eles crescem e evoluem.