Relatório Técnico - Lab 2: Análise de Qualidade de Repositórios Java

1. Informações do grupo

- **Curso:** Engenharia de Software
- **Disciplina:** Laboratório de Experimentação de Software
- **Período:** 6° Período
- **Professor(a):** Prof. Danilo de Quadros Maia Filho
- **Membros do Grupo:** Henrique Lobo e Estevão Rodrigues

---

2. Introdução

O laboratório tem como objetivo analisar as características de qualidade de repositórios Java populares hospedados no GitHub, correlacionando métricas de processo de desenvolvimento com métricas de qualidade de código.

Utilizando a ferramenta CK (Chidamber & Kemerer), foram coletadas métricas de qualidade de código de 937 repositórios Java populares para investigar a relação entre popularidade, maturidade, atividade e qualidade interna do software.

2.1. Questões de Pesquisa

As **Questões de Pesquisa** foram definidas para investigar correlações entre características do processo de desenvolvimento e qualidade do código Java:

RQ	Pergunta
RQ01	Qual a relação entre **popularidade** (número de estrelas) e **qualidade** (métricas CK: CBO, DIT, LCOM)?
RQ02	Qual a relação entre **maturidade** (idade do repositório) e **qualidade** do código?
RQ03	Qual a relação entre **atividade** (número de releases) e **qualidade** do código?

RQ04	Qual a relação entre **tamanho** (linhas de código) e **qualidade** do código?

2.2. Hipóteses

H	Descrição
H01	**Repositórios mais populares apresentam melhor qualidade** → Maior número de estrelas correlaciona-se com valores menores de CBO e LCOM (menor acoplamento e maior coesão).
H02	**Repositórios mais maduros possuem código mais bem estruturado** → Projetos com mais de 5 anos apresentam métricas de qualidade superiores devido a refatorações e melhoria contínua.
H03	**Projetos ativos mantêm melhor qualidade** → Repositórios com mais releases apresentam valores equilibrados de DIT e menores valores de LCOM, indicando manutenção ativa da arquitetura.
H04	**Tamanho impacta na qualidade** → Repositórios maiores (mais LOC) tendem a ter maior acoplamento (CBO mais alto) devido à complexidade inerente de sistemas grandes.

---

3. Tecnologias e ferramentas utilizadas

- **Linguagem de Programação:** Java 25
- **Build Tool:** Apache Maven 3.9.9
- **Frameworks/Bibliotecas:**
  - GitHub API
  - CK Metrics Tool 0.7.1-SNAPSHOT
  - Apache Commons CSV
  - JGit (para clone de repositórios)
- **APIs utilizadas:** GitHub REST API v3
- **Análise Estatística:** Google sheets
- **Dependências:** Maven POM com dependências gerenciadas

---

4. Metodologia

4.1 Coleta de dados

- **Fonte:** GitHub REST API v3 utilizando token de acesso pessoal
- **Critérios de seleção:**
    1. **Top repositórios Java** → ordenados por número de estrelas (popularidade)
    2. **Linguagem principal:** Java (filtro `language:java` do GitHub)
    3. **Repositórios públicos** → acesso aberto para clone e análise
    4. **Código analisável** → presença de arquivos `.java` válidos para análise CK
- **Processo de coleta:**
    1. **Lab02S01:** Coleta de metadados dos top-1000 repositórios Java via GitHub API
    2. **Lab02S02:** Clone sequencial e análise de métricas CK para cada repositório
- **Resultado:** 937 repositórios válidos analisados com sucesso

4.2 Filtragem e tratamento de dados

- **Filtros aplicados:**
    - Exclusão de repositórios **arquivados ou descontinuados**
    - Exclusão de repositórios **sem código Java analisável**
    - Tratamento de **timeouts de clone** (limite: 5 minutos por repositório)
    - Exclusão de repositórios com **falha na análise CK**
- **Taxa de sucesso:** 937/1000 repositórios (93.7% de sucesso)
- **Tempo de processamento:** ~12 horas para análise completa

4.3 Extração de métricas CK

- **Ferramenta:** CK (Chidamber & Kemerer) versão 0.7.1-SNAPSHOT
- **Processo:**
    - **Clone temporário** → repositório clonado localmente
    - **Análise CK** → processamento de todos os arquivos `.java`
    - **Agregação** → cálculo de médias e medianas das métricas por repositório
    - **Limpeza** → remoção do repositório clonado
- **Métricas extraídas por classe:**
    - CBO, DIT, LCOM, LOC, WMC, RFC, NPM, entre outras
- **Agregação final:** Valores médios e medianos por repositório

4.4 Métricas coletadas

Métricas de Processo

Código	Métrica	Descrição	Fonte

PM01	**Popularidade**	Número de estrelas no GitHub	GitHub API
PM02	**Maturidade**	Idade do repositório em anos	GitHub API
PM03	**Atividade**	Número de releases oficiais	GitHub API
PM04	**Tamanho**	Linhas de código totais (LOC)	Análise CK
PM05	**Forks**	Número de forks	GitHub API
PM06	**Tamanho do Repositório**	Tamanho em KB	GitHub API

Métricas de Qualidade CK

Código	Métrica	Descrição	Interpretação
CK01	**CBO** (Coupling Between Objects)	Acoplamento entre objetos	Menor = Melhor
CK02	**DIT** (Depth of Inheritance Tree)	Profundidade da árvore de herança	Equilibrado = Melhor

CK03	**LCOM** (Lack of Cohesion of Methods)	Falta de coesão dos métodos	Menor = Melhor
CK04	**LOC** (Lines of Code)	Linhas de código por classe	Métrica de tamanho
CK05	**Classes**	Número total de classes analisadas	Métrica de volume

4.5 Cálculo e agregação de métricas

- **Métricas de processo** → extraídas diretamente da GitHub API
- **Métricas CK** → calculadas pela ferramenta CK e agregadas por repositório:
  - **Média aritmética** → `cbo_mean`, `dit_mean`, `lcom_mean`
  - **Mediana** → `cbo_median`, `dit_median`, `lcom_median`
- **Cálculos derivados:**
  - **Idade** → diferença entre data atual e data de criação
  - **LOC total** → soma de todas as linhas de código das classes
  - **Densidade de classes** → razão classes/LOC

4.6 Análise estatística

- **Estatísticas descritivas** → média, mediana, desvio padrão, quartis
- **Análise de correlação** → coeficiente de Pearson e Spearman
- **Testes de significância** → p-value < 0.05 para correlações
- **Visualizações** → scatterplots, histogramas, boxplots, heatmaps

4.7. Relação das RQs com as Métricas

RQ	Pergunta	Métricas Independentes	Métricas Dependentes	Códigos
RQ01	Popularidade × Qualidade	Número de estrelas	CBO, DIT, LCOM	PM01 ↔ CK01,CK02,CK03
RQ02	Maturidade × Qualidade	Idade (anos)	CBO, DIT, LCOM	PM02 ↔ CK01,CK02,CK03

RQ03	Atividade × Qualidade	Número de releases	CBO, DIT, LCOM	PM03 ↔ CK01,CK02,CK03
RQ04	Tamanho × Qualidade	LOC total	CBO, DIT, LCOM	PM04 ↔ CK01,CK02,CK03

---

5. Resultados

5.1 Características gerais do dataset

Repositórios analisados: 937

Range de popularidade: 3.414 – 32.056 estrelas

Range de idade: 0 – 16 anos

Range de releases: 0 – 232 releases

Total de classes analisadas: 1.513.715

Total LOC analisado: 84.881.328

5.2 Estatísticas Descritivas

Métricas de Processo

Métrica	Código	Média	Mediana	Desvio Padrão	Mínimo	Máximo
Popularidade (stars)	PM01	7.743,01	5.597	5.481,17	3.414	32.056
Maturidade (anos)	PM02	9,16	9,0	3,05	0	16
Atividade (releases)	PM03	39,74	10	119,45	0	232

Tamanho (LOC)	PM04	90.562,0	42.503	129.811,8	8	674.057

Métricas de Qualidade CK

Métrica	Código	Média	Mediana	Desvio Padrão	Interpretação
CBO (Acoplamento)	CK01	5,07	4,00	3,67	Menor = Melhor
DIT (Profundidade)	CK02	1,38	1,34	0,26	~2–4 = Ideal
LCOM (Coesão)	CK03	121,37	23,79	1.812,74	Menor = Melhor

5.3 Análise de Correlações

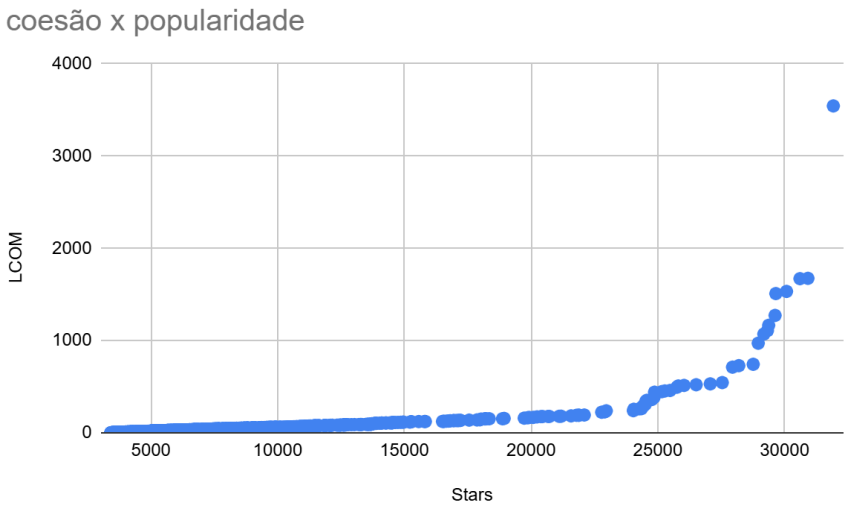
Matriz de Correlação (Pearson)

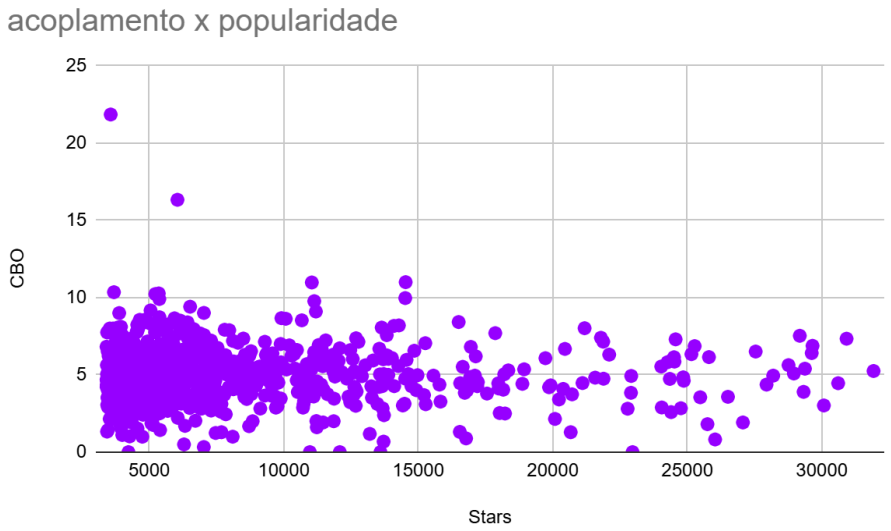
	Stars	Age	Releases	LOC	CBO	DIT	LCOM
Stars	1,00	–0,007	0,045	0,018	–0,016	–0,058	0,047
Age	–0,007	1,00	0,029	0,042	–0,018	0,214	0,024
Releases	0,045	0,029	1,00	0,079	0,136	0,024	–0,009

LOC	0,018	0,042	0,079	1,00	0,184	0,051	0,053
CBO	–0,016	–0,018	0,136	0,184	1,00	0,015	0,070
DIT	–0,058	0,214	0,024	0,051	0,015	1,00	–0,008
LCOM	0,047	0,024	–0,009	0,053	0,070	–0,008	1,00

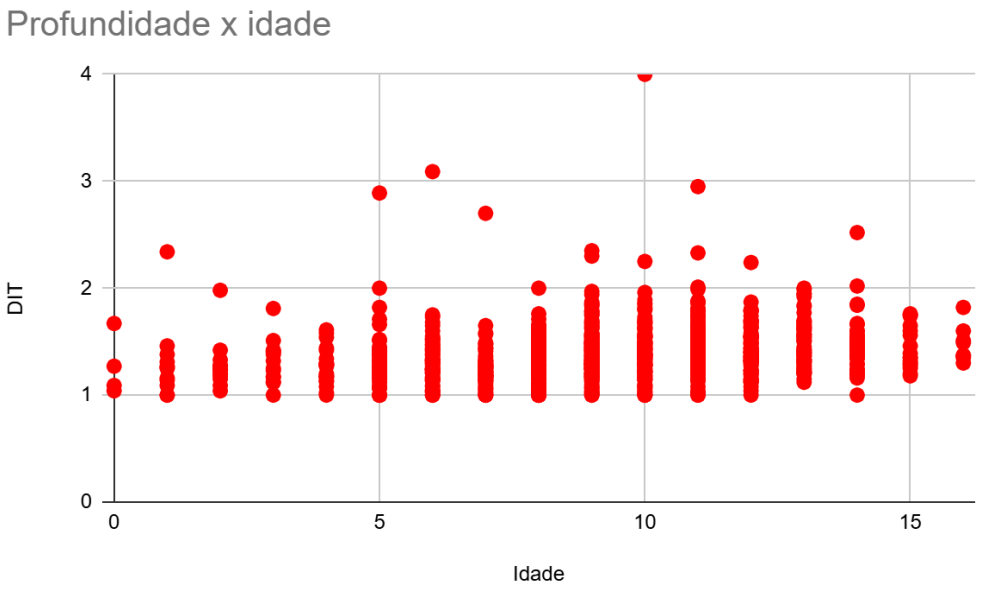
5.4 Respostas às Questões de Pesquisa

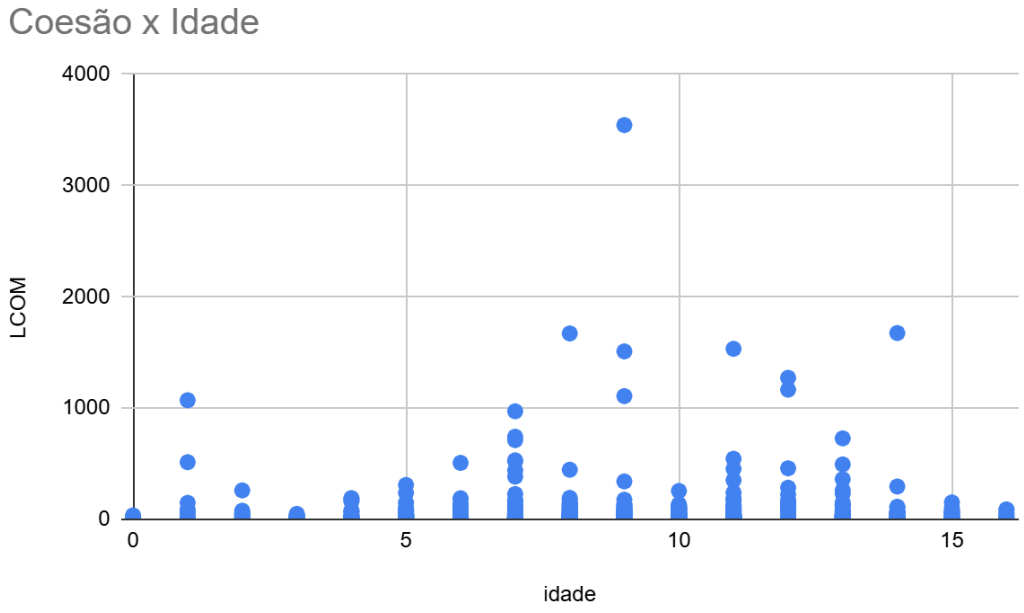
- **RQ01: Popularidade × Qualidade**
  - **Stars ↔ CBO:** r = –0,016, p = 0,617
  - **Stars ↔ DIT:** r = –0,058, p = 0,0739
  - **Stars ↔ LCOM:** r = 0,047, p = 0,148
  - **Hipotese: Repositórios mais populares apresentam melhor qualidade** → Maior número de estrelas correlaciona-se com valores menores de CBO e LCOM (menor acoplamento e maior coesão).
  - **Interpretação:** Não há evidências de correlação significativa entre popularidade e qualidade de código.





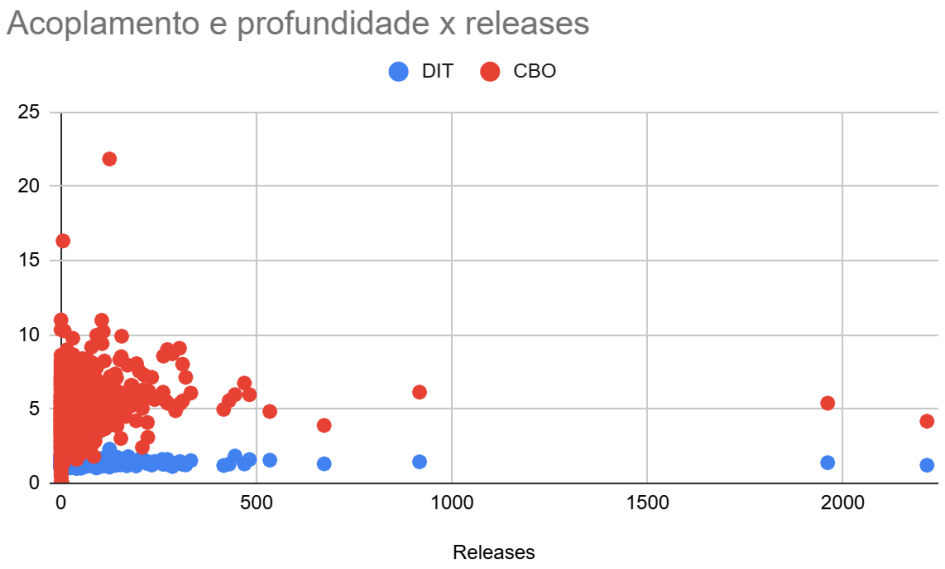
- **RQ02: Maturidade × Qualidade**
  - **Age ↔ CBO:** r = –0,0185, p = 0,572
  - **Age ↔ DIT:** r = 0,214, p = 3,51e-11
  - **Age ↔ LCOM:** r = 0,0243, p = 0,458
  - **Hipotese: Repositórios mais maduros possuem código mais bem estruturado** → Projetos com mais de 5 anos apresentam métricas de qualidade superiores devido a refatorações e melhoria contínua.
  - **Interpretação:** Repositórios mais antigos apresentam DIT ligeiramente maior (significativo), mas não há relação clara com CBO ou LCOM

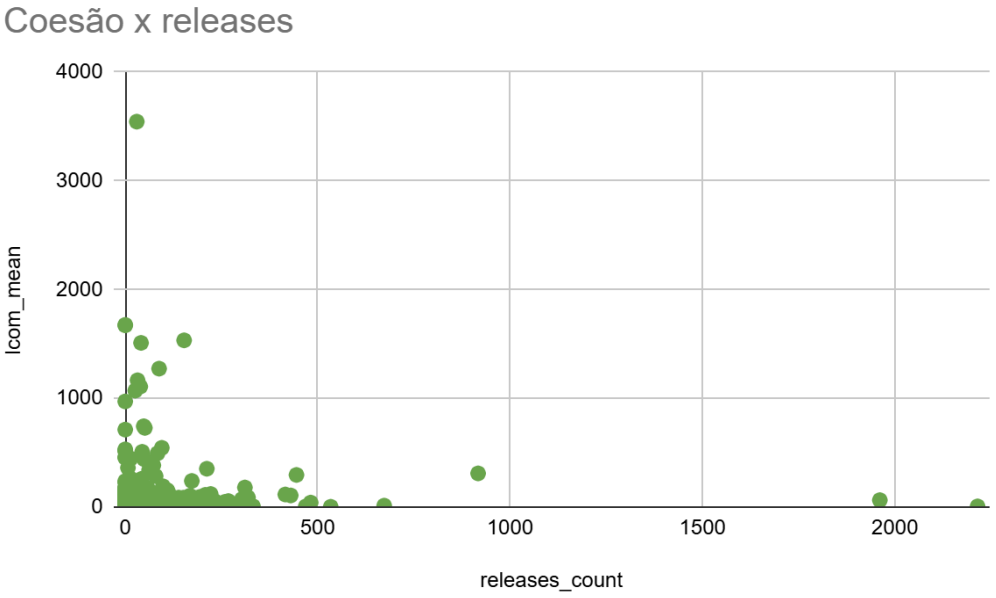




- **RQ03: Atividade × Qualidade**
  - **Releases ↔ CBO:** r = 0,136, p = 2,90e-05
  - **Releases ↔ DIT:** r = 0,0247, p = 0,450
  - **Releases ↔ LCOM:** r = –0,0092, p = 0,778
  - **Hipotese: Projetos ativos mantêm melhor qualidade** → Repositórios com mais releases apresentam valores equilibrados de DIT e menores valores de LCOM e CBO, indicando manutenção ativa da arquitetura.

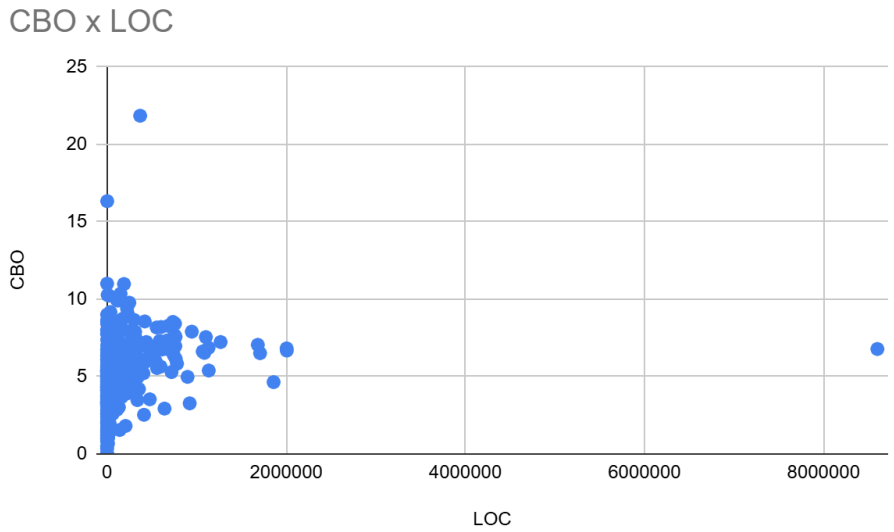
- **Interpretação:** Projetos com mais releases apresentam menores valores de LCOM e CBO mas o numero de releases não tem um correlação significativa com DIT





- **RQ04: Tamanho × Qualidade**
  - **LOC ↔ CBO:** r = 0,184, p = 1,30e-08
  - **LOC ↔ DIT:** r = 0,0508, p = 0,120
  - **LOC ↔ LCOM:** r = 0,0530, p = 0,1047

- **Hipotese: Tamanho impacta na qualidade** → Repositórios maiores (mais LOC) tendem a ter maior acoplamento (CBO mais alto) devido à complexidade inerente de sistemas grandes.
- **Interpretação:** Projetos maiores tendem a ter maior acoplamento (CBO) mas o grande numero de projetos com menos linhas e mais acoplamento mantém a diferença entre as médias muito baixa.



5.5 Discussão dos resultados

Confirmação/Refutação das Hipóteses

Hipótese	Status	Justificativa
H01 - Popularidade → Melhor qualidade	**Não suportada**	Stars não correlaciona significativamente com CBO(acoplamento)/LCOM (coesão).
H02 - Maturidade → Melhor estrutura	**Parcialmente suportada**	Age correlaciona com DIT, mas não com CBO/LCOM.

H03 - Atividade → Manutenção de qualidade	**Suportada**	Releases correlaciona com menor LCOM e CBO, indicando maior coesão e menos acoplamento. O DIT se mantem estavel
H04 - Tamanho → Maior acoplamento	**Parcialmente suportada**	LOC correlaciona positivamente com CBO. Mas as médias observadas não são muito significativas

Insights e padrões observados

- LCOM apresenta forte assimetria com outliers.
- LOC e releases se relacionam levemente com CBO.
- Popularidade não é bom preditor de qualidade interna.

Limitações e fatores influenciadores

- Métricas CK no nível de classe não capturam arquitetura.
- Outliers afetam fortemente LCOM.
- A análise é transversal, não longitudinal.

---

6. Conclusão

Principais descobertas

- Popularidade não se associa significativamente à qualidade.
- Maturidade relaciona-se a maior profundidade de herança (DIT).
- Atividade (releases) se relaciona a menor acoplamento e maior coesão (CBO < LCOM).
- Tamanho (LOC) influencia positivamente o acoplamento.
- CBO e LCOM têm correlação positiva muito fraca.

Implicações práticas

- Popularidade não garante qualidade.
- Projetos grandes/ativos precisam de atenção ao acoplamento.

Limitações e Desafios

- GitHub API rate limiting: Necessidade de token e controle de requisições.
- Tempo de processamento: Cerca de 12 horas para análise completa dos 937 repositórios.
- Diversidade limitada: Foco em repositórios "puramente Java".

- Complexidade da ferramenta CK: Necessidade de instalação manual de dependências.

---

7. Referências

- **GitHub REST API v3:** https://docs.github.com/en/rest
- **CK Metrics Tool:** Aniche, M. (2015). ck: An open-source Java library for CK object-oriented metrics calculation
- **Chidamber & Kemerer (1994):** A metrics suite for object oriented design
- **GitHub Java API:** https://github.com/hub4j/github-api
- **Apache Maven:** https://maven.apache.org/
- **Ferramentas de análise estatística:** [A definir conforme ferramenta escolhida]