A Evolução da Cultura de Testes em Projetos Java Maduros: Um Estudo de Caso Múltiplo sobre a Evolução de Estratégias, Ferramentas e Sofisticação

Carlos Henrique Neimar Areas Ferreira[1], **João Victor Temponi Daltro de Castro**[1]

[1]Pontifícia Universidade Católica de Minas Gerais (PUCMG)
Belo Horizonte – MG – Brasil

{carlos.ferreira, joao.castro}@pucminas.br

Abstract. *Software testing is crucial for quality assurance, yet how testing culture evolves in mature, long-lived projects is not fully understood. This paper presents a longitudinal multiple case study analyzing the evolution of testing strategies in mature Java projects, such as Keycloak and Hibernate. We investigate the evolution of test artifact volume (RQ1), the prevalence and distribution of 21 types of test smells (RQ2), and the relationship between testing effort and maintenance effort across releases (RQ3). By mining software repositories and applying automated analysis scripts, we map the trajectory of test sophistication. Our findings provide insights into how testing practices mature, offering a quantitative characterization of the shift in testing culture over time, highlighting patterns in test growth and quality improvement.*

Resumo. *Testes de software são cruciais para a garantia de qualidade, mas a forma como a cultura de testes evolui em projetos maduros e de longa duração não é totalmente compreendida. Este artigo apresenta um estudo de caso múltiplo longitudinal que analisa a evolução das estratégias de teste em projetos Java maduros, como Keycloak. Investigamos a evolução de seus artefatos de teste ao longo das releases, analisando o volume total de métodos e a densidade de métodos por arquivo (RQ1). Além disso, utilizando a ferramenta TsDetect, identificamos a prevalência dos principais Test Smells (RQ2) para avaliar a qualidade interna da suíte de testes. Nossos resultados quantitativos, extraídos diretamente do histórico do repositório, mapeiam a trajetória de amadurecimento das práticas de teste, revelando um crescimento expressivo tanto em volume quanto em densidade, e identificando desafios de qualidade específicos, como a alta incidência de Magic Number Test.*

1. Introdução

A garantia de qualidade de software é um pilar fundamental no desenvolvimento de sistemas complexos, sendo os testes de software a principal ferramenta para assegurar a confiabilidade e robustez das aplicações. Em projetos de código aberto de grande escala e longa duração, a suíte de testes não é apenas um artefato, mas um reflexo da cultura de qualidade da equipe.

Embora a importância dos testes seja inquestionável, pouco se sabe sobre como as práticas, estratégias e a própria cultura de testes evoluem ao longo do tempo. Projetos

”maduros” passam por inúmeros ciclos de release, mudanças de arquitetura e rotação de contribuidores. Compreender como o ecossistema de testes se adapta e amadurece neste cenário é o foco deste trabalho.

A motivação para este estudo reside na necessidade de caracterizar quantitativamente essa evolução. Projetos maduros, como os analisados neste estudo (Keycloak), representam casos valiosos de sucesso e complexidade. Analisar sua trajetória de testes pode revelar padrões de amadurecimento que auxiliam outras equipes a melhorar suas próprias estratégias.

A justificativa para esta pesquisa é a lacuna na literatura sobre estudos longitudinais focados na cultura de testes. Muitos estudos analisam a qualidade de testes em um ponto específico no tempo, mas falham em capturar a dinâmica da sua evolução.

O objetivo geral deste trabalho é caracterizar a evolução da cultura de testes em projetos Java maduros, analisando suas estratégias, ferramentas e nível de sofisticação ao longo do tempo.

As perguntas de pesquisa que guiarão este trabalho são:

- **RQ1:** Como o volume total de artefatos de teste (arquivos e métodos de teste) evolui ao longo das releases de um projeto?
- **RQ2:** Qual é a prevalência de cada um dos 21 Test Smells nas classes de teste?
- **RQ3:** Qual é a relação entre o esforço de teste e o esforço de manutenção dentro de um ciclo de release?

2. Trabalhos Relacionados

Esta seção revisa a literatura sobre evolução de software, test smells e mineração de repositórios aplicada a testes.

2.1. Trabalho Relacionado 1: Evolução de Software

Estudos sobre evolução de software frequentemente analisam métricas de código-fonte, como complexidade ciclomática e acoplamento, ao longo do tempo (e.g., Leis de Lehman). No entanto, a evolução específica dos artefatos de teste é menos explorada. Este trabalho se diferencia ao focar exclusivamente na maturação da suíte de testes, tratando-a como um sistema em evolução.

2.2. Trabalho Relacionado 2: Detecção de Test Smells

Test Smells são indicadores de baixa qualidade em código de teste. Trabalhos anteriores focaram em catalogar (e.g., Van Deursen et al.) e construir ferramentas para detectar esses smells. Nosso estudo utiliza essa base conceitual, mas aplica a detecção de forma longitudinal para investigar se projetos maduros conseguem refatorar e reduzir a incidência desses smells ao longo do tempo.

2.3. Trabalho Relacionado 3: Mineração de Repositórios de Software (MSR)

A MSR é amplamente utilizada para extrair conhecimento de artefatos de desenvolvimento. Estudos prévios usaram MSR para prever defeitos ou analisar esforço. Este trabalho aplica técnicas de MSR (análise de commits e versionamento) para quantificar o esforço dedicado a testes versus manutenção corretiva, buscando correlações entre essas atividades.

2.4. Trabalho Relacionado 4: Maturidade de Testes

Modelos de maturidade, como o OSMM (Open Software Maturity Model), o grau de maturidade de projetos de software a partir de evidências objetivas de práticas e qualidade. O modelo propõe uma análise baseada em dimensões como processos, qualidade e evolução, servindo como referência para entender como projetos amadurecem ao longo do tempo

3. Metodologia

Para responder às perguntas de pesquisa, adotamos uma metodologia de estudo de caso múltiplo longitudinal. Selecionamos projetos Java maduros e de código aberto com um histórico substancial de releases. A coleta de dados é automatizada por meio de scripts Python que interagem com os repositórios GitHub, extraindo métricas em snapshots definidos (tags de release).

3.1. RQ1: Evolução do Volume de Artefatos de Teste

Para responder a esta pergunta, analisamos o crescimento da suíte de testes. Nossos scripts iteraram sobre as tags de release do projeto, calculando o Total de Métodos de Teste e a Média de Métodos por Arquivo em cada ponto no tempo. Os dados foram plotados para análise de tendência.

Utilizamos o script `run_rq1_volume.py` para percorrer o histórico de releases, contando o número total de arquivos de teste identificados por convenção de nomenclatura e o número de métodos de teste, identificados por anotações.

3.1.1. Hipóteses

Hipótese Nula (H0): Não há mudança estatisticamente significativa no volume total de métodos de teste ao longo das releases.

Hipótese Alternativa (H1): O volume total de métodos de teste apresenta uma tendência de crescimento estatisticamente significativa e positiva ao longo das releases.

3.2. RQ2: Prevalência de Test Smells

Para avaliar a qualidade interna dos testes, investigamos a prevalência de 21 tipos de Test Smells (e.g., Assertion Roulette, EmptyTest, IgnoredTest). Utilizamos a ferramenta TsDetect para realizar uma análise estática na última versão do código-fonte. Esta ferramenta identificou e quantificou a ocorrência de 21 tipos de Test Smells, permitindo-nos classificar os problemas de qualidade mais prevalentes.

Utilizamos os scripts `run_rq2_smells.py` e `run_rq2_analysis.py`, que integram uma ferramenta de detecção de smells (baseada em análise estática) e agregam os resultados por release.

3.2.1. Hipóteses

Hipótese Nula (H0): A prevalência dos Test Smells é distribuída de forma uniforme, sem que categorias específicas de ”smells” se destaquem significativamente.

Hipótese Alternativa (H1): A distribuição de Test Smells é desigual, com tipos específicos dominando em volume, indicando desafios de qualidade pontuais e sistêmicos.

3.3. RQ3: Relação entre Esforço de Teste e Manutenção

Para esta RQ, investigamos a relação entre o esforço dedicado a escrever/atualizar testes e o esforço dedicado a corrigir defeitos. A análise da RQ3 correlaciona commits de teste com commits de correção de bugs, está em fase de coleta de dados e será incluída por completo em uma versão futura deste relatório.

Utilizamos o script `RQ4.py` para minerar as mensagens de commit entre duas releases, classificando-os (baseado em palavras-chave como 'fix', 'bug', 'test', 'refactor') em commits de teste e commits de correção.

3.3.1. Hipóteses

Hipótese Nula (H0): Não há correlação estatisticamente significativa entre o volume de commits de teste e o volume de commits de correção de bugs dentro de um ciclo de release.

Hipótese Alternativa (H1): Existe uma correlação estatisticamente significativa (positiva ou negativa) entre o esforço de teste e o esforço de manutenção.

4. Resultados

4.1. Resultados da RQ1: Evolução do Volume de Artefatos de Teste

A análise do gráfico de evolução do volume de testes demonstra uma tendência de crescimento positiva, clara e sustentada ao longo do tempo. O número total de métodos de teste iniciou em valores baixos nas primeiras releases (índices 0-50) e cresceu exponencialmente, ultrapassando a marca de 11.000 métodos de teste nas releases mais recentes.

Conclusão (RQ1): Os dados contradizem H0. A hipótese nula (H0) é rejeitada. Há uma evidência estatística e visual clara que suporta a hipótese alternativa (H1).

4.2. Resultados da RQ2: Prevalência de Test Smells

A análise executada com a ferramenta TsDetect gerou dados quantitativos sobre a prevalência de *smells*. A distribuição é visivelmente desigual.

1. **Magic Number Test:** Apresenta uma dominância massiva, com aproximadamente **90.000 ocorrências**.
2. **Exception Catching Throwing:** Segundo mais prevalente, com $\sim$48.000 ocorrências.
3. **Assertion Roulette:** Em terceiro lugar, com $\sim$44.000 ocorrências.
4. **Unknown Test** e **Duplicate Assert:** Apresentam volumes significativamente menores ($\sim$17.000 e $\sim$12.000, respectivamente).

Conclusão (RQ2): Os dados contradizem H0. **A hipótese nula (H0) é rejeitada.** A distribuição é fortemente enviesada, com o "Magic Number Test" sendo o desafio de qualidade mais proeminente, suportando amplamente a H1.

4.3. Resultados da RQ3: Relação entre Esforço de Teste e Manutenção

Esta RQ não pôde ser avaliada empiricamente nesta versão do relatório.

5. Ameaças à Validade

1. **Validade Externa (Ameaça Principal):** Esta é a limitação mais significativa e *intencional* desta primeira versão. Os resultados são de um **estudo de caso único (N=1)**. Nenhuma conclusão sobre ”projetos maduros” em geral pode ser feita. A trajetória cultural do Keycloak é apenas isso: a trajetória do Keycloak. **O objetivo desta fase não é a generalização**, mas sim a **validação do pipeline metodológico** (nossos scripts de coleta, a integração com o TsDetect e a estrutura de análise). Este pipeline provou ser eficaz e agora está pronto para ser escalado para os 128+ repositórios planejados, onde a validade externa será tratada.
2. **Validade de Construto:** Medimos ”cultura de testes” e ”sofisticação” por meio de *proxies* (métricas de volume, densidade e contagem de *smells*). Esses são construtos complexos que as métricas apenas aproximam. Além disso, nossa análise de qualidade (RQ2) está inteiramente dependente das definições e da precisão da ferramenta **TsDetect**. A ”verdadeira” qualidade do teste pode ser diferente.
3. **Validade Interna:** Para a futura RQ3 (análise de commits), a classificação de ”esforço de teste” vs. ”esforço de manutenção” com base em mensagens de commit é uma fonte conhecida de viés, pois as mensagens podem ser imprecisas ou misturar múltiplos propósitos.

6. Conclusão e Próximos Passos

Este trabalho apresentou um estudo de caso longitudinal piloto, focado no projeto Keycloak, com o objetivo de caracterizar a evolução de sua cultura de testes. Demonstramos com sucesso a aplicação de um pipeline metodológico que combina mineração de repositório (RQ1) e análise estática de *Test Smells* (para RQ2).

Contribuições Atuais (Baseadas no Piloto):

1. **Validação Metodológica:** Provamos que nosso pipeline automatizado é capaz de extrair e analisar métricas longitudinais de volume, densidade e qualidade de testes.
2. **Hipótese de Maturação:** Geramos uma hipótese inicial sobre a maturação da cultura de testes: ela evolui de um foco em *volume* (cobertura) para um foco em *densidade* (profundidade), mas essa transição, se não for gerenciada, gera *dívida técnica massiva* (smells).
3. **Insight Prescritivo (O ”Quando”):** O caso do Keycloak sugere fortemente que intervenções culturais, como a adoção de linters para prevenir *Test Smells* (ex: ”Magic Numbers”), devem ser implementadas *no início do ciclo de vida* do projeto, antes que a dívida se torne impagável.

Trabalhos Futuros (O Estudo Principal):

O sucesso deste estudo piloto abre caminho para o trabalho principal. O próximo passo imediato é aplicar este pipeline validado ao conjunto de dados completo de **mais de 128 repositórios Java maduros**.

Essa análise em larga escala nos permitirá ir além das observações de um único projeto e buscará generalizar os padrões de evolução. Nosso objetivo final é construir um modelo empírico que identifique os estágios de maturidade da cultura de testes e os pontos de inflexão médios. Isso permitirá que equipes de novos projetos usem nosso trabalho para tomar decisões informadas sobre quando implementar práticas de teste específicas, baseando-se em padrões observados em centenas de projetos de sucesso.

References