Relatório Técnico - Lab 4: Impacto da Frequência de Releases na Qualidade e Segurança de Código

1. Informações

Pontifícia Universidade Católica de Minas Gerais
Curso: Engenharia de Software
Disciplina: Laboratório de Experimentação de Software
Professor: Danilo Quadros
Alunos: Henrique Lobo, Estevão Rodrigues

---

2. Introdução

A velocidade de entrega de software é um fator crítico para a competitividade, sendo o uso de ciclos de *release* rápidos (Rapid Release Cycle - RRC) uma prática comum em projetos de código aberto (OSS). No entanto, a aceleração da entrega pode, potencialmente, comprometer a qualidade, a segurança e a manutenibilidade do código.

Neste laboratório, o objetivo é investigar a relação entre a frequência do ciclo de *release* e diversas métricas de qualidade de código. Para isso, 200 repositórios do GitHub foram classificados em dois grupos distintos, permitindo uma análise comparativa das distribuições de métricas entre os ciclos rápido e lento.

2.1. Questões de Pesquisa (Research Questions - RQs)

As questões de pesquisa (RQs) guiaram a análise comparativa entre os grupos Rapid Release Cycle (RRC) e Slow Release (SR):

RQ	Pergunta	Métricas analizadas
RQ1	O RRC torna as *releases* mais vulneráveis?	vulnerabilities, security_hotspots,
RQ2	Erros são mais comuns em *releases* de RRC?	bugs

RQ3	O retrabalho é maior em sistemas que utilizam RRC do que em sistemas que utilizam *releases* lentas?	technical_debt_days, duplicated_lines_density

---

3. Metodologia

3.1. Caracterização do Dataset

O estudo utilizou um conjunto de repositórios de código aberto do GitHub, previamente classificados em dois grupos com base no intervalo de tempo entre as *releases*:

- **Rapid Release Cycle (RRC):** Repositórios com ciclos de *release* curtos, definidos por um intervalo entre **5 e 35 dias**.
- **Slow Release (SR):** Repositórios com ciclos de *release* longos, definidos por um intervalo **superior a 60 dias**.

As métricas de qualidade e segurança de código foram obtidas por meio de ferramentas de Análise Estática de Código (como SonarQube, e incluem contagens de falhas e indicadores de qualidade (ex: bugs, vulnerabilities, technical_debt_days).

3.2. Técnica de Análise

Para comparar as distribuições das métricas numéricas entre os grupos RRC e SR, foi utilizada a técnica de **Análise por Boxplot (Diagrama de Caixa)**.

O Boxplot sumariza os dados através de cinco medidas estatísticas (mínimo, primeiro quartil (Q1), mediana, terceiro quartil (Q3) e máximo), permitindo a comparação visual de:

1. **Tendência Central:** Pela posição da mediana (linha no centro da caixa).
2. **Dispersão/Variabilidade:** Pelo tamanho da caixa (Intervalo entre quartis).
3. **Presença de Outliers:** Pelos pontos externos

As métricas categóricas de *rating* (security_rating e reliability_rating) seriam idealmente comparadas através de gráficos de barras de contagem ou proporção.

---

4. Análise e Discussão dos Resultados

A análise dos Boxplots revelou um padrão consistente de forma negativa: o RRC apresentou distribuições significativamente piores (medianas mais altas e maior variabilidade) em todas as métricas de falhas, segurança e qualidade de código.

RQ1: O RRC torna as releases mais vulneráveis?

Métrica	RRC (Rapid) vs. SR (Slow)	Resultado
vulnerabilities	Mediana do RRC é aproximadamente 6x maior que a do SR.	**Maior Risco**
security_hotspots	Mediana do RRC é mais de 5x maior que a do SR.	**Maior Risco**

Discussão: A tendência central e a dispersão (variabilidade) para **vulnerabilities** e **security_hotspots** são consistentemente e significativamente maiores no grupo RRC. Isso sugere uma forte correlação entre a velocidade do ciclo de *release* e o **aumento da vulnerabilidade e do risco de segurança**. O tempo reduzido entre *releases* pode estar limitando a profundidade das revisões de segurança.

RQ2: Erros são mais comuns em releases de RRC?

Métrica	RRC (Rapid) vs. SR (Slow)	Resultado
bugs	Mediana do RRC é mais de 5x maior que a do SR.	**Mais Erros**

Discussão: O Boxplot para **bugs** demonstra que o RRC está fortemente associado a uma contagem de erros muito superior (mediana de aproximadamente 135) em comparação com o SR (mediana de aproximadamente 25). A dispersão mais alta no RRC também indica que a contagem de bugs varia muito mais entre esses projetos. Embora os Boxplots para code_smells não tenham sido detalhados, o padrão observado em bugs sugere uma **inferioridade geral na confiabilidade do código** produzido em ciclos rápidos.

RQ3: O retrabalho é maior em sistemas que utilizam RRC, do que em sistemas que utilizam releases lentas?

Métrica	RRC (Rapid) vs. SR (Slow)	Resultado
technical_debt_days	Mediana do RRC (aprox. 120 dias) é 6x maior que a do SR (aprox. 20 dias).	**Maior Retrabalho**

duplicated_lines_density	A dispersão e mediana do RRC são muito superiores (a caixa do RRC abrange um intervalo muito maior de duplicação).	**Maior Retrabalho**

Discussão: Ambas as métricas de manutenibilidade e potencial de retrabalho. O RRC exibe uma acumulação de **Dívida Técnica** drasticamente maior. Uma DT maior implica em mais tempo futuro gasto em correções e refatorações (retrabalho). A alta densidade de linhas duplicadas no RRC reforça este cenário, pois a duplicação de código aumenta o esforço de manutenção O Rapid Release Cycle está ligado a uma qualidade de código significativamente inferior e maior potencial de retrabalho.

---

5. Conclusão

O objetivo deste trabalho foi analisar o impacto da frequência do ciclo de *release* na segurança, confiabilidade e qualidade do código em repositórios abertos do GitHub, comparando os grupos Rapid Release Cycle (RRC) e Slow Release (SR).

Os resultados da análise foram conclusivos e consistentes em todas as RQs:

1. **Segurança (RQ1):** O RRC demonstrou ser mais vulnerável (mediana 6x maior em vulnerabilities) e com mais pontos críticos de segurança.
2. **Confiabilidade (RQ2):** O RRC apresentou uma incidência de bugs substancialmente maior (mediana 5x maior).
3. **Manutenibilidade/Retrabalho (RQ3):** O RRC acumula uma Dívida Técnica maior e possui maior duplicação de código, indicando um custo futuro de manutenção e retrabalho mais elevado.

O achado central é que a **velocidade na entrega, representada pelo Rapid Release Cycle, está fortemente correlacionada com uma degradação generalizada nas métricas de qualidade e segurança do código**. Isso sugere que o tempo de ciclo reduzido pode não ser suficiente para acomodar processos robustos de garantia de qualidade (como testes aprofundados e revisões completas), resultando em produtos com maior risco e custo de manutenção a longo prazo.