Caracterizando a atividade de code review em repositórios populares do GitHub

Carlos Henrique Neimar, João Victor Temponi Daltro de Castro

1 Introdução

1.1 O paradigma e os desafios do code review em projetos de código aberto

A prática de code review tornou-se um pilar nos processos de desenvolvimento ágeis. Em sua essência, ela consiste na interação entre desenvolvedores e revisores com o objetivo de inspecionar o código produzido antes de sua integração à base principal do projeto. Este processo é fundamental para garantir a qualidade do código que é integrado, além de ser uma estratégia eficaz para evitar a introdução de novos defeitos no software.

No ecossistema de software de código aberto (open source), especialmente nos projetos hospedados na plataforma GitHub, as atividades de code review são estruturadas em torno da avaliação de contribuições submetidas por meio de Pull Requests (PRs). Para que uma nova contribuição de código seja incorporada à branch principal, é necessário que um colaborador do projeto avalie e discuta a solicitação, que ao final pode ser aprovada (merged) ou rejeitada (closed). Frequentemente, ferramentas de verificação estática e de integração contínua realizam uma análise preliminar para assegurar a conformidade com estilos de programação e padrões definidos pela organização. É neste cenário que se insere o presente estudo, cujo objetivo é analisar a atividade de code review para identificar variáveis que influenciam na aceitação ou rejeição de um PR.

1.2 Quantificando o Processo de Code Review com Métricas de Pull Request

Para avaliar objetivamente o processo de revisão, este estudo utiliza um conjunto de métricas que quantificam dimensões distintas de um Pull Request:

- **Tamanho**: Mede a magnitude da contribuição. É quantificado pelo número de arquivos modificados e pelo total de linhas de código adicionadas e removidas.

- **Tempo de Análise**: Representa o tempo que um PR permaneceu em discussão. É

definido como o intervalo entre a data de criação do PR e a data de sua última atividade (seja o merge ou o fechamento).

- **Descrição**: Avalia o esforço do contribuidor em contextualizar sua mudança. É mensurado pelo número de caracteres presentes no corpo da descrição do PR.

- **Interações**: Captura o nível de engajamento e discussão gerado pelo PR. É medido pelo número de participantes envolvidos na discussão e pelo número total de comentários.

1.3 Questões de Pesquisa e Hipóteses

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

A. Relação com o Feedback Final das Revisões (Status do PR):

- QP01: Qual a relação entre o tamanho dos PRs e o feedback final das revisões?
  - H01: O tamanho de um PR não possui correlação estatisticamente significativa com seu status final (merged/closed).
  - H11: O tamanho de um PR possui correlação estatisticamente significativa com seu status final.

- QP02: Qual a relação entre o tempo de análise dos PRs e o feedback final das revisões?
  - H02: O tempo de análise de um PR não possui correlação estatisticamente significativa com seu status final.
  - H12: O tempo de análise de um PR possui correlação estatisticamente significativa com seu status final.

- QP03: Qual a relação entre a descrição dos PRs e o feedback final das revisões?
  - H03: A qualidade da descrição de um PR não possui correlação estatisticamente significativa com seu status final.
  - H13: A qualidade da descrição de um PR possui correlação estatisticamente significativa com seu status final.

- QP04: Qual a relação entre as interações nos PRs e o feedback final das revisões?
    - H04: O nível de interação em um PR não possui correlação estatisticamente significativa com seu status final.
    - H14: O nível de interação em um PR possui correlação estatisticamente significativa com seu status final.

B. Relação com o Número de Revisões:

- QP05: Qual a relação entre o tamanho dos PRs e o número de revisões realizadas?
    - H05: O tamanho de um PR não possui correlação estatisticamente significativa com o número de revisões que recebe.
    - H15: O tamanho de um PR possui correlação estatisticamente significativa com o número de revisões que recebe.

- QP06: Qual a relação entre o tempo de análise dos PRs e o número de revisões realizadas?
    - H06: O tempo de análise de um PR não possui correlação estatisticamente significativa com o número de revisões que recebe.
    - H16: O tempo de análise de um PR possui correlação estatisticamente significativa com o número de revisões que recebe.

- QP07: Qual a relação entre a descrição dos PRs e o número de revisões realizadas?
    - H07: A qualidade da descrição de um PR não possui correlação estatisticamente significativa com o número de revisões que recebe?
    - H17: A qualidade da descrição de um PR possui correlação estatisticamente significativa com o número de revisões que recebe.

- QP08: Qual a relação entre as interações nos PRs e o número de revisões realizadas?
    - H08: O nível de interação em um PR não possui correlação estatisticamente significativa com o número de revisões que recebe.
    - H18: O nível de interação em um PR possui correlação estatisticamente significativa com o número de revisões que recebe.

2 Metodologia

2.1 Pesquisa e Amostra

O dataset para este estudo será composto por Pull Requests extraídos de repositórios de

software no GitHub que atendem aos seguintes critérios de seleção:

- **Popularidade**: Serão considerados os 200 repositórios mais populares da plataforma, com base no número de estrelas.
- **Volume de Atividade**: Os repositórios selecionados devem possuir um histórico de, no mínimo, 100 Pull Requests com status MERGED ou CLOSED.

Além dos critérios para os repositórios, os Pull Requests individuais foram filtrados para garantir que passaram por um processo de revisão humano e significativo:

- **Status**: Apenas PRs com status MERGED ou CLOSED serão incluídos na análise.
- **Existência de Revisão**: Cada PR deve ter recebido pelo menos uma revisão.
- **Filtro de Automação**: Para remover PRs possivelmente revisados por bots ou ferramentas automáticas, a análise considerará apenas aqueles cujo tempo entre a criação e o fechamento (ou merge) seja maior que uma hora.

2.2 Coleta e Pré-processamento de Dados

A coleta de dados será realizada de forma automatizada através da criação de um *script*. Este *script* utilizará a API do GitHub para:

1. Identificar a lista de repositórios que se enquadram nos critérios de popularidade e volume.
2. Iterar sobre os PRs de cada repositório, aplicando os filtros de status, número de revisões e tempo de análise.
3. Para cada PR válido, extrair as métricas definidas na seção 1.2: tamanho (arquivos, linhas adicionadas/removidas), tempo de análise, descrição (número de caracteres) e interações (participantes e comentários).

Os dados coletados serão consolidados em um *dataset* único, que servirá de base para a fase de análise estatística.

2.3 Método de Análise Estatística

A análise dos dados será conduzida em duas etapas principais:

- **Estatística Descritiva**: Para caracterizar a amostra, os dados serão sumarizados utilizando a **mediana** como medida de tendência central para cada métrica coletada. Esta abordagem foi escolhida para mitigar o efeito de valores extremos (*outliers*), comuns em dados de repositórios de software.
- **Análise Correlacional**: Para investigar as questões de pesquisa, será utilizado um teste de correlação estatística, como o **coeficiente de correlação de Spearman ou de Pearson**. A escolha do teste será justificada com base na distribuição dos dados (normal ou não-normal). Todas as hipóteses serão avaliadas considerando um nível de

significância (α) de 0.05.

2.4 Desafios da Coleta e Análise de Dados

O processo de extração e preparação dos dados apresentou desafios significativos que exigiram ajustes metodológicos:

- **Curadoria da Amostra**: A implementação inicial do script de coleta continha um erro de lógica que resultou em uma amostra menor que a planejada. O script analisava uma lista de 200 repositórios populares e, em seguida, selecionava apenas os que eram válidos dentro desse grupo, o que resultava em aproximadamente 180 repositórios. A lógica foi ajustada para garantir a busca contínua até que 200 repositórios que atendessem a todos os critérios fossem efetivamente coletados.
- **Custo Computacional e Eficiência da API**: A primeira versão do script utilizava a API REST do GitHub. Essa abordagem se mostrou ineficiente, resultando em um tempo de execução excessivamente longo devido ao alto volume de requisições necessárias. Para otimizar o processo, a implementação foi migrada para a API GraphQL, permitindo a construção de consultas mais específicas que retornam apenas os dados necessários em uma única requisição, reduzindo drasticamente o tempo de coleta.
- **Gestão do Volume de Dados**: A estratégia inicial consistia em salvar todos os dados brutos coletados e aplicar os filtros posteriormente. No entanto, essa abordagem gerou arquivos de dados de tamanho massivo, tornando a manipulação e a filtragem computacionalmente custosas e lentas. O processo foi refinado para que os filtros fossem aplicados em tempo de execução, diretamente durante a coleta, garantindo que apenas os dados relevantes fossem salvos no dataset final, otimizando o armazenamento e o processamento subsequente.

3 Resultados e Análise

4 Discussão

5 Conclusão