PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
ENGENHARIA DE SOFTWARE - CAMPUS LOURDES
LABORATÓRIO DE EXPERIMENTAÇÃO DE SOFTWARE

Ana Carolina Corrêa, Caio Elias,
Henrique Diniz e Maria Clara Santos

Caracterizando a Atividade de Code Review no GitHub

24 de outubro de 2024
Belo Horizonte

LISTA DE FIGURAS

SUMÁRIO

1. Introdução

No cenário de desenvolvimento de software open-source, o processo de code review desempenha um papel fundamental na garantia da qualidade do código antes de sua integração na base principal. Este estudo foca em entender como as características dos pull requests (PRs) em repositórios populares do GitHub influenciam seu merge ou fechamento, considerando variáveis como tamanho do PR, tempo de análise, qualidade da descrição e nível de interação entre desenvolvedores.

Com base nessas variáveis, busca-se identificar padrões que possam contribuir para um processo de revisão mais eficaz e que levem a uma integração mais segura e otimizada de código nos projetos open-source.

2. Hipóteses

As hipóteses a serem investigadas neste estudo consideram a relação entre as características dos PRs e o feedback final das revisões, assim como o número de revisões realizadas:

- **RQ 01: Qual a relação entre o tamanho dos PRs e o feedback final das revisões?**

PRs maiores, em termos de arquivos, linhas adicionadas e removidas, têm menor chance de aprovação rápida, devido à complexidade aumentada e à necessidade de revisões mais detalhadas.

- **RQ 02: Qual a relação entre o tempo de análise dos PRs e o feedback final das revisões?**

PRs com um tempo de análise mais longo tendem a ter uma maior chance de aprovação, pois o tempo permite uma revisão mais minuciosa e completa.

- **RQ 03: Qual a relação entre a descrição dos PRs e o feedback final das revisões?**

PRs com descrições mais detalhadas apresentam maior probabilidade de aprovação, pois a clareza nas informações facilita a compreensão do objetivo e das alterações propostas.

- **RQ 04: Qual a relação entre as interações nos PRs e o feedback final das revisões?**

PRs com mais interações entre desenvolvedores (comentários, revisões) tendem a ser mais aprovados, pois envolvem mais discussões e melhorias

colaborativas antes do merge.

- **RQ 05: Qual a relação entre o tamanho dos PRs e o número de revisões realizadas?**

PRs maiores tendem a passar por um maior número de revisões, já que mudanças extensas requerem mais verificações para evitar problemas de integração.

- **RQ 06: Qual a relação entre o tempo de análise dos PRs e o número de revisões realizadas?**

PRs com um tempo de análise mais longo podem envolver mais revisões, já que o tempo estendido pode ser reflexo de revisões adicionais para atender aos requisitos.

- **RQ 07: Qual a relação entre a descrição dos PRs e o número de revisões realizadas?**

PRs com descrições mais claras e completas podem demandar menos revisões, pois são mais fáceis de entender e avaliar.

- **RQ 08: Qual a relação entre as interações nos PRs e o número de revisões realizadas?**

PRs com maior número de interações devem ter mais revisões, já que a comunicação ativa entre os colaboradores indica maior envolvimento no processo de revisão.

3. Metodologia

A análise da atividade de code review em repositórios populares do GitHub foi realizada por meio de uma abordagem sistemática que envolveu a coleta e processamento de dados utilizando a API GraphQL. As principais etapas do processo, desde a obtenção dos dados até o processamento final e a análise das métricas, são descritas a seguir.

3.1. Seleção de Repositórios

A primeira etapa consistiu na coleta de repositórios do GitHub. Foi utilizado o GitHub GraphQL API para realizar consultas e obter informações relevantes dos repositórios:

Query GraphQL: Foi feita uma consulta com a seguinte configuração: repositórios com pelo menos 100 PRs fechados ou mesclados, limitando a coleta de 30 por página até chegar a um total de 200 repositórios. Para cada pull request foram coletadas as seguintes informações:

- Nome do repositório;

- Número do PR;
- Quantidade de arquivos;
- Adições e Remoções;
- Horas de duração (data de fechamento/merge - data de criação);
- Tamanho da descrição;
- Número de comentários;
- Número de participantes;

3.1.1 Execução da Query

O código em Python realizou múltiplas requisições até atingir o número de repositórios necessários. O controle de paginação foi feito por meio do cursor fornecido pela API do GitHub.

3.1.2 Armazenamento Inicial

Os dados obtidos foram salvos em um arquivo CSV para consulta posterior, evitando a necessidade de repetir as requisições à API em caso de falha.

3.2. Coleta de Dados dos Pull Requests

Para cada PR, foram aplicados filtros específicos:

- Apenas PRs com status "mesclado" (merged) ou "fechado" (closed) foram considerados.
- Somente PRs que passaram por pelo menos uma revisão manual foram incluídos na análise, descartando revisões automáticas.
- O tempo de duração da análise do PR foi considerado apenas se superou uma hora, garantindo que revisões automáticas e rápidas não fossem incluídas na amostra.

As principais métricas coletadas para cada PR incluíram:

- **Número de arquivos alterados**: Reflete o tamanho e o escopo da alteração proposta.
- **Linhas adicionadas e removidas**: Medem a complexidade e o impacto do PR no código base.
- **Duração da análise**: Calculada como o tempo entre a criação do PR e o fechamento ou merge.
- **Tamanho da descrição**: Medido em número de caracteres, avaliando o nível de detalhamento das informações fornecidas no PR.
- **Número de comentários**: Indica o nível de interação e discussão entre os revisores e o autor do PR.
- **Número de participantes**: Considera todos os revisores e comentaristas

envolvidos no processo de revisão, refletindo o grau de colaboração.

3.3. Execução do Script de Coleta

O script de coleta foi projetado para rodar de forma contínua e sequencial, percorrendo todos os repositórios selecionados e iterando sobre os PRs de cada repositório.

3.3.1 Tratamento de limitação de requisições

Para mitigar problemas de limitação de requisições, o script inclui verificações automáticas do limite de taxa da API do GitHub, pausando a execução até que o limite fosse restaurado, evitando falhas ou interrupções no processo de coleta.

3.3.2 Armazenamento de Dados

Os dados coletados foram continuamente armazenados em um arquivo CSV durante a execução do script. A estrutura do script foi configurada para atualizar o arquivo a cada repositório processado, garantindo que a coleta de dados não fosse perdida em caso de interrupção ou falha inesperada.

3.3.3 Tratamento e Limpeza dos Dados

Após a coleta, foi realizada uma etapa de limpeza dos dados para remover duplicatas, tratar possíveis erros de coleta e garantir a consistência das métricas registradas.

PRs que não atendiam aos critérios estabelecidos, como aqueles sem revisões ou com tempo de análise inferior a uma hora, foram filtrados na fase de pré-processamento dos dados.

3.4 Análise das Métricas e Correlações

A análise estatística foi conduzida para identificar correlações entre as características dos PRs e os resultados das revisões (aprovação ou rejeição), bem como o número de revisões realizadas.

Os testes de correlação de Spearman ou Pearson foram utilizados, dependendo da distribuição dos dados e da linearidade das relações observadas. O teste de Spearman foi preferido para relações não lineares, enquanto o teste de Pearson foi utilizado para relações lineares.

As métricas analisadas incluíram:

1. Tamanho do PR (número de arquivos, linhas adicionadas/removidas).
2. Tempo de análise (duração do processo de revisão).
3. Descrição do PR (comprimento e detalhamento da descrição).
4. Interações (número de comentários e participantes).

Os resultados das análises foram utilizados para testar as hipóteses

formuladas, identificando se e como as características dos PRs afetam a probabilidade de aprovação, rejeição e o número de revisões necessárias.

3.5 Armazenamento e Visualização dos Resultados

Os resultados das análises foram armazenados em um arquivo CSV consolidado, permitindo uma fácil manipulação dos dados para criação de visualizações gráficas e sumarização das estatísticas.

Gráficos de dispersão, histogramas e boxplots foram gerados para visualizar as relações entre as variáveis analisadas e facilitar a interpretação dos resultados.

4. Resultados

5. Conclusão