Pontifícia Universidade Católica de Minas Gerais
Laboratório de Experimentação de Software
ISSN:

RELATÓRIO

Caracterizando a Atividade de Code Review no GitHub

Characterizing the Code Review Activity on GitHub

Lorrayne Oliveira
Pontifícia Universidade Católica de Minas Gerais, lorrayne.marayze@gmail.com
Pedro Pires Pontifícia Universidade Católica de Minas Gerais, pedro.pires@gmail.com

Resumo. Este trabalho apresenta um estudo empírico sobre a prática de code review em repositórios populares do GitHub. A atividade de revisão de código, fundamental para assegurar a qualidade e manutenibilidade do software, é analisada sob diferentes dimensões, considerando métricas associadas ao tamanho, tempo de análise, descrição e interações de pull requests (PRs). O objetivo central é compreender como essas variáveis influenciam o resultado das revisões (merge ou rejeição) e o número de revisões realizadas. Para isso, foi construído um dataset contendo PRs dos 200 repositórios mais populares do GitHub, avaliando somente aqueles que passaram por processos de revisão humana e tiveram tempo mínimo de análise superior a uma hora. As correlações entre as variáveis foram examinadas por meio de análises estatísticas de Spearman, de modo a identificar padrões significativos no comportamento das revisões de código. Os resultados obtidos fornecem evidências quantitativas sobre fatores que impactam o sucesso das contribuições, oferecendo insights sobre as dinâmicas colaborativas de revisão em projetos open-source.

Abstract. This work presents an empirical study on the practice of code review in popular GitHub repositories. The code review activity, essential for ensuring software quality and maintainability, is analyzed through multiple dimensions, considering metrics related to the size, review time, description, and interactions of pull requests (PRs). The main goal is to understand how these variables influence the review outcomes (merge or rejection) and the number of reviews performed. A dataset was built containing PRs from the 200 most popular GitHub repositories, considering only those with human reviews and analysis time greater than one hour. Correlations among variables were examined using the Spearman statistical test to identify significant patterns in review behavior. The results provide quantitative evidence about factors influencing contribution success, offering insights into collaborative review dynamics in open-source projects.

Palavras-chave: Code Review; Pull Requests; GitHub; Revisão de Código; Engenharia de Software Colaborativa

Keywords: Code Review; Pull Requests; GitHub; Code Quality; Collaborative Software Engineering

1 Introdução

A revisão de código (code review) é uma das práticas mais consolidadas em processos ágeis de desenvolvimento de software. Ela consiste na inspeção do código por revisores antes de sua integração ao repositório principal, com o intuito de detectar defeitos, melhorar a legibilidade e assegurar a aderência aos padrões de qualidade. No contexto de plataformas open-source como o GitHub, essa atividade é operacionalizada por meio de pull requests (PRs), que representam contribuições submetidas por desenvolvedores e avaliadas por colaboradores do projeto.

Neste trabalho, busca-se caracterizar empiricamente o comportamento das atividades de code review em repositórios populares do GitHub. Especificamente, este estudo visa compreender a forma com que os fatores, como o tamanho do PR, o tempo de análise, a extensão da descrição e o volume de interações influenciam o resultado final da revisão (merge ou rejeição) e o número de revisões realizadas.

A partir da análise, pretende-se identificar padrões recorrentes e variáveis críticas que impactam o sucesso de revisões, contribuindo para a melhoria de práticas de colaboração e qualidade de código em ambientes distribuídos de desenvolvimento.

1.1 Hipóteses Informais

Com base nas observações do ecossistema de desenvolvimento open-source, foram elaboradas as seguintes hipóteses informais para orientar a investigação:

IH01: Pull requests menores, com menos arquivos e linhas alteradas, têm maior probabilidade de serem aprovados.

IH02: PRs analisados em tempo moderado (entre 1h e 48h) tendem a ser mais aceitos, enquanto análises muito longas reduzem a taxa de merge.

IH03: PRs com descrições mais completas aumentam a chance de aprovação, por facilitar a compreensão do revisor.

IH04: Um número moderado de interações (comentários e participantes) está positivamente associado à aceitação do PR, enquanto muitas interações podem indicar divergências.

IH05: PRs menores demandam menos ciclos de revisão, reduzindo o esforço necessário do revisor.

IH06: O tempo de análise está positivamente correlacionado com o número de revisões. Revisões mais longas tendem a envolver mais ciclos de feedback.

IH07: Descrições detalhadas diminuem o número de revisões necessárias, pois fornecem contexto adequado desde o início.

IH08: PRs com mais interações (comentários, revisores e discussões) tendem a passar por mais revisões até sua aprovação final.

1.2 Objetivos

O objetivo principal deste trabalho é caracterizar empiricamente a atividade de *code review* em repositórios populares do GitHub, analisando fatores que influenciam o resultado e a dinâmica das revisões de código. A partir de métricas quan-

titativas extraídas de *pull requests* (PRs), busca-se responder às seguintes questões de pesquisa:

- **RQ01:** Qual a relação entre o tamanho dos *pull requests* e o feedback final das revisões?
- **RQ02:** Qual a relação entre o tempo de análise dos *pull requests* e o feedback final das revisões?
- **RQ03:** Qual a relação entre a descrição dos *pull requests* e o feedback final das revisões?
- **RQ04:** Qual a relação entre as interações nos *pull requests* e o feedback final das revisões?
- **RQ05:** Qual a relação entre o tamanho dos *pull requests* e o número de revisões realizadas?
- **RQ06:** Qual a relação entre o tempo de análise dos *pull requests* e o número de revisões realizadas?
- **RQ07:** Qual a relação entre a descrição dos *pull requests* e o número de revisões realizadas?
- **RQ08:** Qual a relação entre as interações nos *pull requests* e o número de revisões realizadas?

Como objetivo secundário, pretende-se validar ou refutar as hipóteses informais elaboradas, contribuindo para o entendimento dos fatores que afetam o sucesso e a eficiência do processo de revisão de código em projetos *open-source*.

2 Metodologia

2.1 Coleta de Dados

A coleta de dados foi realizada através da API GraphQL do GitHub, utilizando um script em Python desenvolvido especificamente para esta análise. O processo de coleta contemplou informações relacionadas às atividades de revisão de código, englobando tanto características dos *pull requests* (PRs) quanto atributos do processo de *code review*. As etapas seguiram o seguinte fluxo:

Autenticação: Utilização de token de acesso pessoal do GitHub para autenticar as requisições à API e permitir a coleta de grandes volumes de dados.

Consulta e Paginação: Implementação de consultas em GraphQL para coletar dados dos 200 repositórios mais populares do GitHub, considerando o número de estrelas como critério de popularidade. Foi utilizado um mecanismo de paginação para garantir a coleta completa de todos os 200 PRs por repositório.

Filtragem dos PRs: Seleção apenas dos PRs com status *MERGED* ou *CLOSED*, que possuíam ao menos uma revisão (*review count* > 0) e tempo total de análise superior a uma hora, a fim de descartar revisões automáticas realizadas por *bots* ou pipelines de CI/CD.

Extração de Métricas de Revisão: Para cada PR foram coletadas informações sobre tamanho (arquivos e linhas modificadas), tempo de análise (diferença entre *createdAt* e *closedAt*), descrição (tamanho do texto em *markdown*), interações (número de comentários e participantes) e número total de revisões.

Tratamento de Erros: Implementação de mecanismos de controle de taxa (*rate limiting*) e de repetição automática de requisições em caso de falhas, assegurando consistência e integridade dos dados coletados.

2.2 Métricas Coletadas

Para cada *pull request* analisado, foram coletadas métricas relacionadas tanto ao processo de revisão quanto às características estruturais do PR. As métricas foram agrupadas em quatro dimensões principais, conforme descrito a seguir:

Tamanho: número de arquivos modificados e total de linhas adicionadas/removidas. **Tempo de Análise:** intervalo de tempo entre a criação e o fechamento do PR. **Descrição:** número de caracteres no corpo da descrição do PR, em formato *markdown*. **Interações:** número total de participantes e comentários na discussão. **Revisões:** quantidade de revisões formais registradas no campo *review count*. **Status Final:** estado final do PR, podendo ser *MERGED* (aceito) ou *CLOSED* (rejeitado).

Essas métricas permitem analisar as correlações entre características dos PRs e o resultado das revisões, respondendo às questões de pesquisa propostas no trabalho.

2.3 Exportação e Análise

Os dados processados foram exportados para um arquivo JSON (`repositorios_github_dados.json`) com o objetivo de facilitar as análises estatísticas e permitir a replicação dos resultados. A análise foi conduzida a partir de duas abordagens complementares: medidas de tendência central e análise de correlação.

A etapa de tendência central envolveu o uso de medidas como a mediana e a média para resumir a distribuição dos valores obtidos em cada métrica (tamanho, tempo de análise, descrição, interações e revisões). Esse procedimento possibilita observar o comportamento geral dos pull requests (PRs) sem a influência de valores extremos.

Na segunda etapa, foi aplicada a **correlação de Pearson**, a fim de identificar o grau e a direção das relações lineares entre as variáveis analisadas. A escolha do coeficiente de Pearson, em detrimento de outras técnicas como a correlação de Spearman, deve-se a duas razões principais:

- Natureza contínua e quantitativa das variáveis: As métricas utilizadas (número de linhas alteradas, tempo de análise em horas, tamanho das descrições, número de comentários e revisões) são essencialmente contínuas e quantitativas, o que torna o coeficiente de Pearson mais apropriado, já que ele mede relações lineares em dados intervalares ou racionais.
- Análise de força linear: Diferentemente do coeficiente de Spearman, que captura relações monotônicas (lineares ou não), o objetivo neste estudo é quantificar especificamente o grau de associação linear entre variáveis, permitindo avaliar se aumentos em uma métrica (por exemplo, tamanho do PR) estão diretamente associados a aumentos ou reduções em outra (como tempo de análise ou número de revisões).

2.4 Limitações

Este estudo apresenta algumas limitações que devem ser consideradas para uma interpretação adequada dos resultados.

Primeiramente, devido a restrições de processamento computacional, memória e tempo de execução das consultas à API do GitHub, não foram coletados todos os pull requests de cada um dos 200 repositórios analisados. Em

vez disso, foram amostrados 200 PRs por repositório, totalizando 40.000 registros no conjunto de dados. Esse recorte, pode não refletir integralmente o comportamento completo de cada projeto, especialmente em repositórios com volume muito maior de contribuições.

Outra limitação refere-se à ausência de informações qualitativas sobre o contexto dos PRs. O tempo total de análise, por exemplo, pode incluir períodos de inatividade do revisor ou do autor, não refletindo necessariamente o esforço real de revisão necessário. De forma semelhante, o tamanho da descrição foi avaliado apenas em número de caracteres, sem análise semântica da clareza, completude ou relevância do conteúdo.

3 Resultados

3.1 RQ01: Qual a relação entre o tamanho dos PRs e o feedback final das revisões?

No gráfico abaixo, é possível observar que as categorias de tamanho "muito pequeno" e "pequeno" concentram o maior número de pull requests aceitos, enquanto conforme o tamanho aumenta, a quantidade de merges diminui gradualmente.

Os dados mostram que nas faixas de 0 a 10 e de 11 a 100 linhas modificadas, o número de pull requests aprovados é significativamente maior do que o número de fechados. Isso indica que alterações menores são mais comuns e têm mais chance de serem aceitas. Já nos pull requests médios (101 a 500 linhas) e grandes (501 a 1000 linhas), o número de merges cai consideravelmente, mostrando que revisões mais extensas são menos prováveis de serem aprovadas. Para os pull requests muito grandes, acima de 1000 linhas modificadas, o número de merges continua baixo, o que reforça essa tendência.

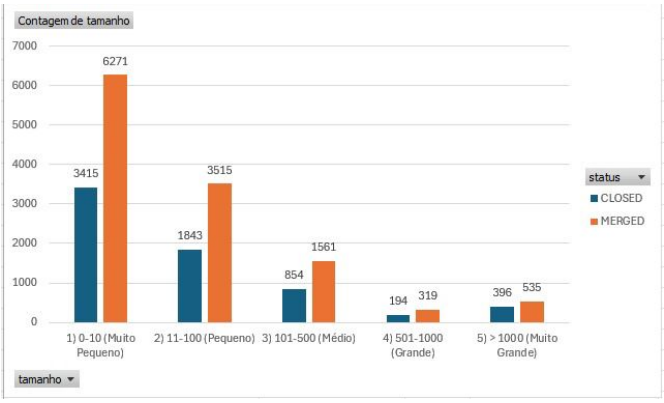

Figura 1. Tamanho dos PRs x Feedback Final

Esses resultados confirmam a hipótese inicial (IH01), que propunha que pull requests menores têm maior probabilidade de aprovação. É possível entender isso porque revisões pequenas são mais rápidas de analisar, têm menor risco de causar conflitos e exigem menos esforço de revisão. À medida que o tamanho do pull request aumenta, a complexidade também cresce, tornando o processo de análise mais demorado e sujeito a rejeições.

Com base nessas observações, podemos concluir que existe uma relação negativa entre o tamanho do pull request e o sucesso da revisão. Ou seja, quanto maior o volume de modificações, menor a chance de o pull request ser aceito.

3.2 RQ02: Qual a relação entre o tempo de análise dos PRs e o feedback final das revisões?

Os pull requests foram agrupados em quatro intervalos de tempo: de 0 a 24 horas (muito rápidos ou triviais), de 24 horas a 1 semana (pico de engajamento), de 1 semana a 1 mês (moderado ou de longa duração) e acima de 1 mês (estagnado ou muito longo).

Os resultados mostram que a grande maioria dos pull requests foi analisada e concluída em até 24 horas, com 8.612 merges e 3.099 fechamentos. Essa faixa de tempo concentra o maior número de aprovações, indicando que revisões rápidas tendem a ser mais bem-sucedidas. No intervalo de 24 horas a 1 semana, o número de merges diminui para 2.247 e os fechamentos para 1.312, o que ainda representa uma boa taxa de aprovação, embora menor que na faixa anterior. Já nas revisões com duração de 1 semana a 1 mês, observa-se um equilíbrio entre merges (791) e fechamentos (812), sugerindo que revisões mais longas não garantem necessariamente melhores resultados. Por fim, os pull requests com duração superior a 1 mês apresentam uma inversão clara, com 1.479 fechamentos contra apenas 551 merges, o que mostra que revisões muito demoradas têm uma alta chance de serem rejeitadas ou abandonadas.

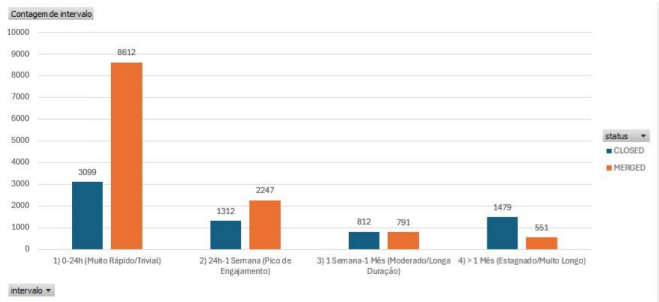

Figura 2. Tempo de Análise x Feedback Final

Esses resultados estão alinhados com a hipótese IH02, que previa que revisões com duração moderada (entre 1 hora e 48 horas) teriam maior probabilidade de aprovação, enquanto tempos muito curtos ou longos reduziriam essa taxa. No gráfico, é possível perceber que a maioria dos merges ocorre em revisões rápidas, mas não excessivamente demoradas. Quando o tempo aumenta para além de uma semana, o número de merges cai progressivamente, o que sugere que revisões longas perdem engajamento dos revisores e dos autores, podendo gerar acúmulo de comentários e atrasos na decisão final.

Revisões rápidas e objetivas apresentam melhores resultados, enquanto revisões prolongadas tendem a ser rejeitadas com mais frequência. Essa relação linear entre o aumento do tempo e a diminuição dos merges justifica o uso do coeficiente de correlação de Pearson, já que ele permite quantificar o grau de dependência linear entre o tempo de análise e o sucesso do pull request.

3.3 RQ03: Qual a relação entre a descrição dos PRs e o feedback final das revisões?

A RQ03 busca compreender como o tamanho da descrição dos pull requests se relaciona com o resultado final das revisões. As descrições no gráfico foram agrupadas em quatro faixas: curtas (0 a 5.000 caracteres), moderadas (5.001 a 15.000), longas (15.001 a 30.000) e muito longas (acima de 30.000 caracteres).

O gráfico mostra que a maioria dos pull requests apresenta descrições curtas, com 12.150 merges e 6.628 fechamentos. Esse comportamento indica que a maior parte das revisões aprovadas está concentrada em PRs que possuem descrições curtas.

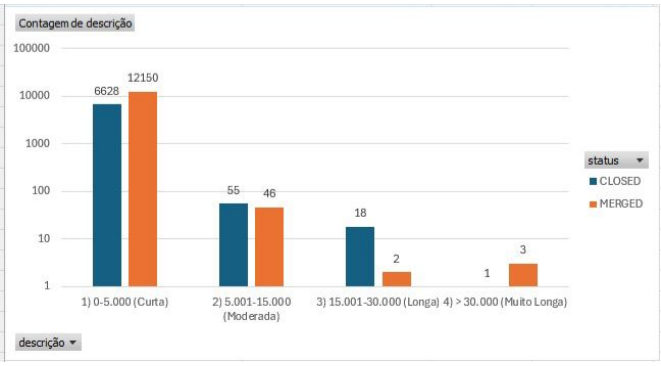

Figura 3. Tempo de Análise x Feedback Final

Nas categorias de descrições moderadas, há 55 fechamentos e 46 merges, sugerindo uma pequena redução na taxa de aprovação, mas sem uma diferença expressiva em relação às curtas. Já nas descrições longas e muito longas, o número de ocorrências é bastante reduzido, o que limita a análise quantitativa e dificulta tirar conclusões estatísticas robustas. Ainda assim, observa-se que pull requests com descrições muito extensas são menos comuns e, em alguns casos, menos aceitos.

Os resultados confirmam parcialmente a hipótese IH03, que previa que descrições mais completas e detalhadas favoreceriam a aprovação do pull request por facilitar a compreensão do revisor. No entanto, os dados mostram que a maioria dos PRs aceitos possui descrições curtas, o que pode estar relacionado ao fato de muitos deles envolverem pequenas alterações, que não exigem explicações extensas. Assim, a clareza e a objetividade da descrição parecem exercer um papel mais relevante do que o comprimento do texto em si.

4 Conclusão

Referências

Ait Houaich, Y. *et al*. (2015). Measuring the maturity of open source software. *ResearchGate*. Preprint.

Badreddin, O., Keshta, I., and Bekhet, S. (2014). A study on maturity model of open source software community to estimate the quality of products. *Procedia Computer Science*, 35:1282–1291. DOI: 10.1016/j.procs.2014.08.233.

Borges, H., Hora, A., and Valente, M. T. (2016a). Predicting the popularity of github repositories. In *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 1–10. ACM. DOI: 10.1145/2972958.2972966.

Borges, H., Hora, A., and Valente, M. T. (2016b). Understanding the factors that impact the popularity of github repositories. In *Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 334–344. IEEE. DOI: 10.1109/SANER.2016.31.

Ferreira, K. R., Bigonha, R. S., Bigonha, M. A., Mendes, L. M. G., and Almeida, H. (2012). A tool for source code analysis of java programs. In *Proceedings of the 2012 Brazilian Symposium on Software Engineering*, pages 1–10.

FINOS (2024). The open source maturity model (osmm). Financial Industry Open Source - Open Source Readiness.

GitHub (2024). Graphql api reference. Documentação oficial da API GraphQL do GitHub.

Hostinger (2025). As linguagens de programação mais usadas em 2025. Acesso em: 27 ago. 2025.

Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2 edition.