josif 2025
17ª Jornada Científica e Tecnológica
14º Simpósio de Pós-Graduação
do IFSULDEMINAS
ISSN: 2319-0124
REALIZAÇÃO
INSTITUTO FEDERAL Sul de Minas Gerais
APOIO
CNPq
FAPEMIG

IMPACTOS DO VERSIONAMENTO DE CÓDIGO NO TRABALHO COLABORATIVO EM EQUIPES DE DESENVOLVIMENTO

Luana S. N. de CASTRO[1]; Paulo C. dos SANTOS[2]

RESUMO

Este artigo apresenta uma Revisão Bibliográfica com o objetivo de analisar como o processo de versionamento de código influencia o trabalho colaborativo em equipes de desenvolvimento de software. A pesquisa investiga os principais sistemas de controle de versão, identifica práticas colaborativas associadas e verifica os benefícios e desafios documentados na literatura recente. Por meio da análise de publicações em bases de dados como IEEE Xplore e ACM Digital Library, o estudo sintetiza o conhecimento sobre como essas ferramentas moldam a comunicação, a integração e a qualidade do produto final, apresentando ao final as boas práticas recomendadas.

Palavras-chave:
Controle de Versão; Git; Colaboração em Software; Revisão de Código; Desenvolvimento Colaborativo.

1. INTRODUÇÃO

Na indústria de desenvolvimento de software, a colaboração efetiva é um pilar para o sucesso de qualquer projeto. A complexidade crescente dos sistemas e a distribuição geográfica das equipes (HOSSAIN; BABAR; PAASIVAARA, 2009) tornam imperativo o uso de ferramentas que orquestrem o trabalho coletivo de forma eficiente e segura. Nesse contexto, os Sistemas de Controle de Versão (VCS, do inglês *Version Control Systems*) surgem como tecnologias fundamentais, permitindo que múltiplos desenvolvedores trabalhem simultaneamente no mesmo código-fonte sem que haja sobreposição destrutiva de suas contribuições.

Apesar da ampla adoção de sistemas como o Git, muitas equipes ainda enfrentam desafios na sua utilização eficaz em contextos colaborativos (GOUSIOS et al., 2014a). A mera presença da ferramenta não garante uma colaboração fluida; a integração de suas práticas ao fluxo de trabalho coletivo influencia diretamente a produtividade, a comunicação e a qualidade do código produzido (KALLIAMVAKOU, 2017). Contudo, ainda existem lacunas na compreensão sobre como o processo de versionamento afeta, de forma prática, a dinâmica do trabalho em equipe. Diante disso, este artigo é guiado pela seguinte pergunta de pesquisa: Como o processo de versionamento de código impacta o trabalho colaborativo em equipes de desenvolvimento de software?

O objetivo geral deste trabalho é analisar, por meio de uma revisão bibliográfica, como o processo de versionamento de código influencia o trabalho colaborativo. Para tal, busca-se investigar os principais sistemas utilizados, identificar práticas colaborativas associadas, verificar benefícios e desafios enfrentados pelas equipes e apresentar boas práticas consolidadas na literatura.

[1]Bolsista PIBIC/CNPq, IFSULDEMINAS – *Campus* Muzambinho. E-mail: luana.castro@alunos.ifsuldeminas.edu.br.
[2]Orientador, IFSULDEMINAS – *Campus* Muzambinho. E-mail: paulo.santos@muz.ifsuldeminas.edu.br.

2. FUNDAMENTAÇÃO TEÓRICA

Um Sistema de Controle de Versão é um software que gerencia e rastreia as alterações em arquivos ao longo do tempo. Ele permite que se reverta um arquivo ou um projeto inteiro para um estado anterior, compare mudanças, identifique quem modificou o código e quando, entre outras funcionalidades (CHACON; STRAUB, 2014). A literatura tradicionalmente classifica os VCS em duas categorias principais: centralizados e distribuídos.

Nos Sistemas de Controle de Versão Centralizados (CVCS), como o Subversion (SVN), há um único servidor que contém todos os arquivos versionados, e os desenvolvedores realizam *check-outs* de arquivos para trabalhar. A principal desvantagem desse modelo é a existência de um ponto único de falha. Em contrapartida, nos Sistemas de Controle de Versão Distribuídos (DVCS), como Git e Mercurial, cada desenvolvedor possui uma cópia completa do repositório, incluindo todo o seu histórico. Esse modelo facilita o trabalho offline e permite fluxos de trabalho mais flexíveis e robustos, tornando-se o padrão de fato na indústria moderna (CHACON; STRAUB, 2014).

O Git, em particular, popularizou práticas colaborativas que se tornaram centrais no desenvolvimento de software. Conceitos como *branches* (ramificações), *merges* (fusões) e *pull requests* (solicitações de fusão) não são apenas comandos técnicos, mas mecanismos que estruturam a interação social e a revisão por pares (ZAGALSKY et al., 2015). Um *branch* permite o desenvolvimento de uma nova funcionalidade de forma isolada, sem afetar a linha de base do código. Ao concluir o trabalho, o desenvolvedor pode solicitar a integração de suas alterações por meio de um *pull request*, um artefato que serve como um fórum para a revisão de código (*code review*), discussão e aprovação antes da fusão definitiva (GOUSIOS; PINZGER; VAN DEURSEN, 2014b; YANG et al., 2024).

3. MATERIAL E MÉTODOS

Este artigo adota a metodologia de revisão bibliográfica para reunir, analisar e sintetizar publicações relevantes sobre o processo de versionamento de código e sua relação com o trabalho colaborativo. A pesquisa foi conduzida em bases de dados de alto impacto acadêmico e científico, incluindo IEEE Xplore, ACM Digital Library, Scopus, Google Scholar e SciELO.

A seleção dos trabalhos foi guiada pela busca de combinações das seguintes palavras-chave e seus correspondentes em inglês: “versionamento de código”, “controle de versão”, “Git”, “colaboração em software”, “trabalho em equipe” e “desenvolvimento colaborativo”. Para garantir a atualidade e relevância da análise, foram aplicados os seguintes critérios de inclusão: (i) publicações no período entre 2013 e 2024; e (ii) trabalhos com foco explícito em ferramentas e práticas colaborativas de versionamento. Foram excluídos artigos com abordagem puramente técnica sobre configurações de ferramentas sem relação com colaboração, bem como trabalhos

duplicados ou com abordagem superficial sobre o tema.

4. RESULTADOS E DISCUSSÃO

A análise da literatura recente confirma a hegemonia do Git como o sistema de controle de versão distribuído padrão. Seu impacto na colaboração pode ser analisado sob a ótica dos benefícios, dos desafios e, principalmente, da interação entre eles.

4.1 Benefícios do Versionamento para a Colaboração

Entre os principais benefícios, a literatura destaca a capacidade de promover o desenvolvimento paralelo. O uso de *branches* permite que equipes trabalhem em diferentes funcionalidades ou correções de forma simultânea e segura, reduzindo gargalos e acelerando o ciclo de desenvolvimento (GOUSIOS; PINZGER; VAN DEURSEN, 2014b). Outro benefício central é a rastreabilidade. O histórico detalhado de *commits* (confirmações de alterações) cria um registro imutável de quem alterou o quê, quando e por quê, o que é fundamental para auditorias, depuração de erros e compreensão da evolução do projeto (CHACON; STRAUB, 2014).

O mecanismo de *pull request* é apontado como a mais importante prática colaborativa. Ele formaliza o processo de revisão por pares (*code review*), que, segundo diversos estudos, está diretamente associado à melhoria da qualidade do código, à disseminação de conhecimento entre os membros da equipe e ao fortalecimento de um senso de propriedade coletiva sobre o software (YANG et al., 2024). A discussão que ocorre durante um *pull request* serve como uma forma de comunicação assíncrona documentada, essencial para equipes distribuídas (GOUSIOS et al., 2014a).

4.2 Desafios e Fatores de Sucesso

Apesar das vantagens, a literatura também aponta desafios significativos. O principal deles é a curva de aprendizado. Ferramentas como o Git possuem um modelo conceitual complexo, e a falta de compreensão aprofundada pode levar a erros graves (ZAGALSKY et al., 2015). Outro desafio recorrente são os conflitos de fusão (*merge conflicts*), que ocorrem quando dois desenvolvedores alteram a mesma parte de um arquivo. A resolução pode ser complexa e consumir um tempo considerável, gerando frustração.

O ponto central da discussão, contudo, é que a eficácia do versionamento depende da adoção de um fluxo de trabalho (*workflow*) claro e compartilhado, como o GitFlow ou o GitHub Flow. A literatura aponta que é a padronização de processos que mitiga os desafios: a prática da revisão de código via *pull requests*, por exemplo, funciona como um mecanismo de mentoria que atenua a curva de aprendizado. Da mesma forma, um *workflow* que preconiza *branches* de curta duração e integração contínua minimiza a ocorrência de conflitos de fusão. Portanto, o impacto do versionamento é tanto técnico quanto social e processual.

5. CONCLUSÃO

Esta revisão bibliográfica concluiu que o impacto do versionamento de código no trabalho colaborativo é significativo, mas depende criticamente da combinação da ferramenta com processos bem definidos. Sistemas como o Git fornecem a infraestrutura técnica para o desenvolvimento paralelo e a revisão por pares, mas sua eficácia é condicionada pela superação de desafios como a curva de aprendizado e conflitos de fusão.

O sucesso reside menos na ferramenta em si e mais na adoção de *workflows* padronizados e em uma cultura de comunicação e revisão construtiva. Diante do exposto, recomenda-se que líderes de equipes de desenvolvimento não apenas implementem a ferramenta, mas invistam proativamente na definição de um fluxo de trabalho adaptado ao contexto do time e promovam uma cultura de revisão de código contínua. Como limitação, este trabalho se ateve à literatura; para trabalhos futuros, sugere-se a realização de estudos de caso empíricos que investiguem a aplicação dessas práticas em contextos industriais brasileiros.

REFERÊNCIAS

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. New York: Apress, 2014. Disponível em: https://git-scm.com/book/en/v2. Acesso em: 2 jul. 2025.

GOUSIOS, Georgios et al. Work practices and challenges in pull-based development: The integrator's perspective. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 36., 2014, Hyderabad. **Proceedings...** New York: ACM, 2014a. p. 345-355. Disponível em: https://gousios.org/pub/pullreqs-integrators.pdf. Acesso em: 2 jul. 2025.

GOUSIOS, Georgios; PINZGER, Martin; VAN DEURSEN, Arie. An exploratory study of the pull-based software development model. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 36., 2014, Hyderabad. **Proceedings...** New York: ACM, 2014b. p. 271-282. Disponível em: https://gousios.org/pub/exploration-pullreqs.pdf. Acesso em: 2 jul. 2025.

HOSSAIN, E.; BABAR, M. A.; PAASIVAARA, M. A systematic review of scrum in global software development. In: INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING, 4., 2009, Limerick. **Proceedings...** Limerick: IEEE, 2009. p. 175-184. Disponível em: https://doi.org/10.1109/ICGSE.2009.25. Acesso em: 2 jul. 2025.

KALLIAMVAKOU, Eirini. **Collaboration via Aligned Autonomy for Commercial Software Teams**. 2017. Tese (Doutorado em Engenharia de Software) – University of Victoria, Victoria, 2017. Disponível em: https://dspace.library.uvic.ca/bitstream/handle/1828/8784/Kalliamvakou_Eirini_PhD_2017.pdf. Acesso em: 2 jul. 2025.

YANG, Zezhou et al. A Survey on Modern Code Review: Progresses, Challenges and Opportunities. **arXiv preprint arXiv:2405.18216**, 2024. Disponível em: https://arxiv.org/abs/2405.18216. Acesso em: 2 jul. 2025.

ZAGALSKY, Alexey et al. The emergence of GitHub as a collaborative platform for education. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK & SOCIAL COMPUTING, 18., 2015, Vancouver. **Proceedings...** New York: ACM, 2015. p. 1906-1917. Disponível em: https://doi.org/10.1145/2675133.2675284. Acesso em: 2 jul. 2025.