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

USO DE HISTÓRICOS DE CONTROLE DE VERSÃO NA EVOLUÇÃO E MELHORIA DO CÓDIGO-FONTE

Pablo D. G. FERNANDES[1]; Paulo C. dos SANTOS[2]

RESUMO

Este artigo, uma Revisão Bibliográfica, analisa estudos acadêmicos que exploram o uso de históricos de Sistemas de Controle de Versão (VCS) para prever, sugerir e analisar alterações no código, visando otimizar a manutenção e a qualidade do software. A metodologia adotou uma revisão da literatura com publicações entre 2012 e 2024. Os resultados indicam que, embora a análise de históricos de VCS seja fundamental para compreender a evolução de projetos de software e detectar padrões de falhas, a eficácia dessas análises é frequentemente limitada por uma falta de dados mais completos no commit. Conclui-se que o histórico de VCS é uma ferramenta estratégica, mas seu potencial só pode ser plenamente explorado com o desenvolvimento de técnicas mais robustas para lidar com a granularidade dos dados.

Palavras-chave: Controle de Versão; Evolução de Software; Mineração de Repositórios; Padrões de Mudança; Qualidade de Código.

1. INTRODUÇÃO

O desenvolvimento de software é um processo iterativo, marcado por constantes modificações no código-fonte para corrigir defeitos, adicionar funcionalidades e aprimorar a qualidade geral. Nesse cenário dinâmico, os Sistemas de Controle de Versão (VCS) são ferramentas indispensáveis, pois registram um histórico detalhado de cada alteração, viabilizando a colaboração e a rastreabilidade (PALESTINO, 2020). O repositório gerado por um VCS constitui uma rica fonte de dados para a área de Mineração de Repositórios de Software (MSR), que busca extrair insights sobre padrões de desenvolvimento, causas de erros e boas práticas de engenharia (NEGARA et al., 2012). Apesar do grande volume de dados disponível, a sua utilização estratégica ainda enfrenta desafios, principalmente relacionados à qualidade e granularidade das informações registradas. Assim, este trabalho busca responder à seguinte questão: de que forma a literatura recente aborda o uso de históricos de VCS para a melhoria do código-fonte e quais os principais desafios apontados?

2. FUNDAMENTAÇÃO TEÓRICA

[1]Discente do Bacharelado em Ciências da Computação, IFSULDEMINAS – *Campus* Muzambinho. E-mail: pablo.fernandes@alunos.ifsuldeminas.edu.br
[2]Docente do Bacharelado em Ciências da Computação, IFSULDEMINAS – *Campus* Muzambinho.
E-mail: paulo.santos@muz.ifsuldeminas.edu.br

Os sistemas de controle de versão evoluíram de ferramentas centralizadas, como o Subversion, para modelos distribuídos, como o Git, que hoje domina o mercado por sua flexibilidade e suporte a fluxos de trabalho complexos (FREITAS, 2010). Essa evolução tecnológica ampliou o volume e a acessibilidade dos dados históricos, impulsionando pesquisas sobre a evolução do software. A criação de repositórios globais, como o Software Heritage, que arquiva publicamente o histórico de milhões de projetos, representa um marco para a pesquisa na área, permitindo análises em larga escala antes inviáveis (PIETRI et al., 2020).

Contudo, a qualidade desses dados é um ponto de atenção crítico. Negara et al. (2012) apontam uma limitação fundamental: a incompletude semântica dos commits. Os desenvolvedores frequentemente agrupam múltiplas alterações lógicas (correções de bugs, refatorações, novas funcionalidades) em um único commit, dificultando a análise automatizada e a correlação precisa entre uma mudança e seu propósito. Essa prática compromete análises precisas, pois dificulta a associação entre causas e efeitos no código. Por exemplo, ao agrupar a correção de um bug e a introdução de uma nova funcionalidade em um único commit, torna-se desafiador treinar modelos capazes de prever falhas ou sugerir refatorações automáticas (SLIWERSKI; ZIMMERMANN; ZELLER, 2005).

Para superar tais desafios, abordagens baseadas em aprendizado de máquina têm ganhado destaque. Cayres, Lima e Garcia (2021), em uma revisão sistemática, mapearam técnicas que aplicam algoritmos para analisar padrões em históricos de commits com o objetivo de prever falhas, identificar módulos críticos e sugerir refatorações. Entre essas técnicas destacam-se a classificação supervisionada, o agrupamento de alterações similares e o uso de redes neurais recorrentes. Ferramentas como ChangeDistiller, Historage, Code2Vec e DeepCommit são exemplos de soluções que vêm sendo exploradas com sucesso nesse contexto.

2.1 Commits Atômicos e sua Importância

Commits atômicos referem-se à prática de registrar apenas uma alteração lógica por commit, como a correção de um erro específico ou a adição de uma funcionalidade. Essa granularidade facilita a rastreabilidade, tornando mais precisa a identificação da origem de problemas e a aplicação de técnicas automatizadas. Sua ausência compromete a confiabilidade de modelos preditivos e dificulta a reprodutibilidade de análises, como observam Tufano et al. (2017).

3. MATERIAL E MÉTODOS

Este estudo consiste em uma revisão bibliográfica da literatura, executada com o objetivo de

identificar tendências e desafios no uso de históricos de VCS para a melhoria de código-fonte. A busca por artigos foi realizada nas bases de dados Google Scholar e arXiv, cobrindo o período de janeiro de 2012 a junho de 2024. Foram utilizados os seguintes termos de busca e suas combinações em português e inglês: ("histórico de controle de versão" OR "version control history"), ("evolução de software" OR "software evolution"), ("mineração de repositórios de software" OR "mining software repositories") AND ("qualidade de código" OR "code quality").

Os critérios de inclusão foram: (a) artigos publicados em conferências ou periódicos revisados por pares; e (b) trabalhos que apresentassem análise, estudo de caso ou abordagem técnica sobre o uso de dados de VCS. Os critérios de exclusão foram: (a) artigos de opinião ou resumos curtos; (b) trabalhos cuja análise não estivesse centrada no histórico de versões; e (c) teses e dissertações. A partir da seleção, os artigos foram analisados e seus principais achados foram sintetizados na seção seguinte.

4. RESULTADOS E DISCUSSÃO

A análise dos artigos selecionados revela duas vertentes principais: o potencial dos dados de VCS e os desafios metodológicos para sua exploração. Os resultados demonstram um consenso sobre a utilidade dos históricos como ferramenta para entender a saúde e a trajetória de um projeto de software. O trabalho de Pietri et al. (2020) sobre o Software Heritage reforça que análises em larga escala podem revelar padrões de evolução que transcendem projetos individuais, oferecendo insights para a indústria como um todo.

A principal barreira identificada é a qualidade dos dados. O alerta de Negara et al. (2012) é amplamente confirmado por outros estudos, apontando a prática de agrupar mudanças como um obstáculo para a aplicação de técnicas de aprendizado de máquina. Conforme Cayres, Lima e Garcia (2021), a eficácia de modelos preditivos depende diretamente da capacidade de isolar e rotular corretamente as alterações. Essa lacuna revela um descompasso entre o potencial analítico e as práticas de versionamento adotadas.

No âmbito prático, esses achados sugerem que equipes de desenvolvimento devem ser incentivadas a adotar commits mais claros, atômicos e semanticamente ricos. Ferramentas como Commitizen, Conventional Commits, Gitlint e Husky são exemplos de soluções que impõem ou recomendam boas práticas. Além disso, pesquisas recentes como Yamashita et al. (2023) investigam o uso de modelos de linguagem natural (LLMs) para classificar e interpretar mensagens de commit com baixa qualidade textual, e sistemas de suporte inteligentes integrados a IDEs para fornecer feedback em tempo real.

5. CONCLUSÃO

Esta revisão da literatura reafirma a importância estratégica dos históricos de controle de versão como fonte de dados para a melhoria contínua do código-fonte. Resgatando a questão norteadora, a literatura aborda o tema principalmente pela aplicação de técnicas de MSR e aprendizado de máquina para identificar padrões e prever falhas. O principal desafio identificado é a baixa granularidade dos dados registrados nos commits, que compromete a precisão e a viabilidade de análises automatizadas complexas.

Conclui-se que, para avançar no campo, os esforços futuros devem seguir uma abordagem dupla: por um lado, o desenvolvimento de algoritmos de aprendizado profundo (deep learning) capazes de inferir padrões mesmo em dados ruidosos; por outro, a proposição de ferramentas e integrações ao ambiente de desenvolvimento que incentivem e facilitem a criação de históricos de versão mais ricos e semanticamente precisos.

REFERÊNCIAS

CAYRES, L. U.; LIMA, B. S.; GARCIA, R. E. Learning and Suggesting Source Code Changes from Version History: A Systematic Review. arXiv preprint arXiv:2105.08861, 2021.

FREITAS, D. T. M. Análise Comparativa entre Sistemas de Controle de Versões. 2010. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Federal de Juiz de Fora, Juiz de Fora, 2010.

NEGARA, S. et al. Is It Dangerous to Use Version Control Histories to Study Source Code Evolution? In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, 26., 2012, Beijing. Proceedings... Berlin: Springer, 2012. p. 425-449.

PALESTINO, C. M. C. Estudo de Tecnologias de Controle de Versões de Softwares. 2020. Trabalho de Conclusão de Curso (Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de São Paulo, São Paulo, 2020.

PIETRI, A. et al. The Software Heritage Graph Dataset: Large-scale Analysis of Public Software Development History. In: INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES, 17., 2020, Seul. Proceedings... New York: ACM, 2020. p. 284-295.