Revista Eletrônica de Iniciação Científica em Computação, 2025, 23:1
ISSN: 3085-8461 • doi: 10.5753/reic.2025.6850
This work is licensed under a Creative Commons Attribution 4.0 International License.

Artigo de Pesquisa/Research Paper

Um Mapeamento Sistemático sobre Técnicas de Classificação e Categorização de Bugs em Software

A Systematic Mapping of Bug Classification and Categorization Techniques in Software

Igor Ramsés Temóteo dos Santos [Instituto Federal de Ciência e Tecnologia do Ceará | *igor.ramses@darmlabs.ifce.edu.br*]
Laís Carvalho Coutinho [Instituto Federal de Ciência e Tecnologia do Ceará | *lais.carvalho07@aluno.ifce.edu.br*]
Samuel Sousa Teles [Instituto Federal de Ciência e Tecnologia do Ceará | *samuel.teles@darmlabs.ifce.edu.br*]
Rubens Abraão da Silva Sousa [Instituto Federal de Ciência e Tecnologia do Ceará | *rubens.silva@darmlabs.ifce.edu.br*]
Manoel Lopes Filho [Instituto Federal de Ciência e Tecnologia do Ceará | *manoel.filho@darmlabs.ifce.edu.br*]
Ronaldo Tadeu Pontes Milfont [Instituto Federal de Ciência e Tecnologia do Ceará | *ronaldo.milfont@ifce.edu.br*]
Alex Lacerda Ramos [Instituto Federal de Ciência e Tecnologia do Ceará | *alex.lacerda@ifce.edu.br*]

Instituto Federal de Ciência e Tecnologia do Ceará, Rodovia BR-020, km 303, s/nº – Jubaia, Canindé – CE, 62700-000, Brasil

Resumo. Este estudo é um mapeamento sistemático da literatura sobre as abordagens de classificação e categorização de bugs em software, analisando 33 artigos finais publicados entre 2019 e 2024. O protocolo foi conduzido seguindo o modelo PRISMA, utilizando a ferramenta Parsifal para triagem, e a busca foi realizada nas bases ISI Web of Science, IEEE Xplore, Scopus e Engineering Village. A síntese dos dados foi realizada através de análise qualitativa e extração padronizada, focando em técnicas, tipos de dados, contextos de aplicação e critérios de categorização. A pesquisa revelou a predominância de algoritmos supervisionados (como Naive Bayes e Support Vector Machine) e a dependência de dados textuais não estruturados de repositórios open-source como Mozilla e Eclipse. A exclusividade de dados open-source restringe a compreensão do ciclo de vida dos bugs em contextos industriais. Consequentemente, identifica-se a necessidade de pesquisas futuras que explorem ambientes corporativos e a integração de modelos híbridos (dados estruturados e não-estruturados) para melhor reflexão do ciclo de vida dos bugs.

Abstract. This study is a systematic mapping of the literature on approaches to classifying and categorizing bugs in software, analyzing 33 final articles published between 2019 and 2024. The protocol was conducted following the PRISMA model, using the Parsifal tool for screening, and the search was performed in the ISI Web of Science, IEEE Xplore, Scopus, and Engineering Village databases. Data synthesis was performed through qualitative analysis and standardized extraction, focusing on technologies, data types, application contexts, and categorization criteria. The research revealed the predominance of supervised algorithms (such as Naive Bayes and Support Vector Machine) and the dependence on unstructured textual data from open-source repositories such as Mozilla and Eclipse. The exclusivity of open-source data restricts understanding of the bug life cycle in industrial contexts. Consequently, there is a need for future research exploring corporate environments and the integration of hybrid models (structured and unstructured data) to better reflect the bug life cycle.

Palavras-chave: Classificação de bugs, Categorização de bugs, Mapeamento sistemático, Software

Keywords: Bug classification, Bug categorization, Systematic mapping, Software

Recebido/Received: 11 November 2025 • **Aceito/Accepted:** 26 December 2025 • **Publicado/Published:** 31 December 2025

1 Introdução

O ciclo de desenvolvimento do software abrange etapas como análise de requisitos, design, implementação, testes e manutenção, que descrevem o processo de construção de um sistema de software. A eficiência desse processo e a contínua satisfação das necessidades dos usuários são intrinsecamente ligadas à garantia de qualidade e confiabilidade do software [qahtan yas *et al*., 2023]. Apesar da adoção de rigorosas práticas de garantia de qualidade, a ocorrência de bugs persiste no ciclo de desenvolvimento, impulsionada pela crescente complexidade dos sistemas e pela natureza propensa a erros da atividade humana de codificação. Consequentemente, esses erros podem comprometer funcionalidades e impactar negativamente a confiança dos usuários [Meher *et al*., 2024]. Assim, a adoção de estratégias que minimizem riscos e assegurem a qualidade dos sistemas entregues é de fundamental importância.

Nesse contexto, abordagens baseadas na classificação e categorização de bugs têm sido aplicadas, utilizando recursos de aprendizado de máquina e processamento de linguagem natural (PNL). Essas técnicas contribuem tanto para uma detecção mais precisa das falhas quanto para a evolução contínua dos sistemas [Laiq *et al*., 2025]. A categorização de bugs pode assumir diferentes cenários, a depender do propósito. Entre as formas mais recorrentes, é destacada a categorização por nível de severidade, tais como crítica, alta, média e baixa. Outra forma utilizada para a categorização é por tipo de falha, tais como erros funcionais, erros de desempenho, defeitos na interface e falhas de segurança.

Contudo, com a diversidade crescente de métodos, conjuntos de dados e métricas de avaliação, torna-se desafiador

comparar os resultados obtidos nas pesquisas e avaliar de forma padronizada suas contribuições [Harzevili *et al*., 2025]. A tensão operacional reside no fato de que a alta fragmentação metodológica impede que a comunidade estabeleça qual combinação de técnicas, tipos de dados e contextos de aplicação oferece a melhor capacidade de generalização e replicabilidade dos resultados em engenharia de software. Portanto, organizar e sintetizar o conhecimento disponível é essencial para fornecer uma base mais clara e estruturada para novos estudos na área [Ordoñez-Pacheco *et al*., 2021].

Diante desse contexto, o presente estudo realiza uma análise do estado da arte sobre as técnicas aplicadas à classificação e categorização de bugs em software, por meio de um Mapeamento Sistemático da Literatura. A partir da análise dos estudos primários selecionados, busca-se identificar tendências tecnológicas, lacunas existentes na pesquisa, técnicas, algoritmos, tipos de dados e contextos recorrentes, oferecendo uma visão consolidada e estruturada das soluções propostas e validadas na literatura [Harzevili *et al*., 2025]. Desta forma, espera-se contribuir de forma significativa para o avanço da área e para a melhoria contínua da qualidade do software e manutenção de software.

O objetivo deste mapeamento sistemático é identificar e analisar as principais técnicas, conjuntos de dados e contexto de aplicação abordados nos estudos voltados à classificação e categorização de bugs. Para alcançar esse objetivo, foram selecionados e analisados 33 artigos primários publicados entre 2019 e 2024, que aplicam diferentes métodos computacionais. A revisão busca oferecer uma visão consolidada das abordagens existentes, destacando contribuições, limitações e trabalhos futuros. Além disso, identificar técnicas emergentes, datasets mais utilizados e métricas mais utilizadas para avaliação de desempenho, assim contribuindo com o fortalecimento do conhecimento na área de Engenharia de Software.

Embora o estudo se concentre na identificação e análise de técnicas, tipos de dados, contextos de aplicação e critérios de classificação, optou-se pela condução de um Mapeamento Sistemático (Systematic Mapping Study) visando fornecer uma visão panorâmica e quantitativa sobre as frequências de técnicas e tipos de dados utilizados na área.

O recorte temporal de 2019 a 2024 foi definido com base em dois fatores: a rápida evolução das técnicas de aprendizado de máquina e processamento de linguagem natural, que têm impactado diretamente as abordagens de classificação e categorização de bugs, e a necessidade de fornecer uma visão atualizada sobre as práticas e técnicas emergentes.

Assim, restringir o escopo aos últimos cinco anos permite concentrar a análise em métodos e contextos mais relevantes ao estado da arte atual. O ano de 2025 não foi incluído integralmente no intervalo de busca sistemática, pois o ano corrente não havia sido concluído no momento da execução do protocolo, visando garantir a reprodutibilidade fiel dos resultados anuais completos.

Este estudo está organizado em cinco seções, sendo elas: 1 - Introdução: apresenta o contexto e o objetivo deste trabalho; 2 - Trabalhos Relacionados: discute estudos prévios semelhantes, destacando contribuições, avanços e lacunas que contextualizam e reforçam a relevância deste estudo; 3 - Método: descreve o protocolo utilizado na condução da Mapeamento Sistemático, incluindo os critérios de inclusão e exclusão, além das questões de pesquisa que orientaram a análise; 4 - Resultados: apresenta os achados da revisão, com ênfase nas tendências e lacunas identificadas; e 5 - Conclusão: traz as considerações finais, destacando as implicações dos resultados e possíveis direções para pesquisas futuras.

2 Trabalhos Relacionados

Priyadarshini [2025] afirma que defeitos de software prejudicam a qualidade do software e representam uma séria ameaça à sua confiabilidade. Este estudo enfatiza a importância de técnicas para tarefas de classificação de defeitos em módulos defeituosos, para garantir a eficiência do processo de desenvolvimento de software. Entretanto, a variabilidade entre os datasets e sua influência na capacidade de generalização dos modelos não são abordadas, não permitindo uma visão ampla sobre a sua real aplicação.

Nesse sentido, Pachouly *et al*. [2022] realizaram uma revisão sistemática sobre a predição de defeitos utilizando técnicas de Inteligência Artificial (IA), analisando aspectos como conjuntos de dados mais utilizados, métodos de validação de dados, abordagens e ferramentas aplicadas. O artigo chama atenção sobre a limitação de diversidade de datasets utilizados, ressaltando a escassez de práticas padronizadas de validação de dados, além de falta de transparência com relação à origem e estrutura de dados nos trabalhos da área, comprovando a necessidade de uma análise detalhada sobre a natureza dos dados utilizados em atividades de classificação e categorização de bugs.

Ainda mais, Gunalan *et al*. [2022] exploraram o uso das descrições textuais, mas agora voltado à categorização de bugs, a partir do uso de técnicas de PLN. Contudo, o estudo limita-se ao não generalizar sua aplicação, visto que foi validado apenas em um conjunto de dados, além de não ser claro acerca da estrutura dos dados ou da disponibilidade pública da base. Nesse contexto, uma análise sobre os tipos de dados lidados a partir das descrições textuais citadas ainda é pouco explorada.

Por outro lado, Afric *et al*. [2023] abordaram o impacto de diferentes classificadores (como RoBERTa e FastText) em contextos variados, a partir de uma análise sobre a influência dos rótulos incorretos na qualidade das bases de dados utilizadas em classificações de falhas. Porém, o estudo não explora sua aplicação em tarefas de categorização específicas, além de não ter apresentado informações sobre a disponibilidade dos dados utilizados. Diante disso, é necessário explorar a estrutura e a natureza dos dados envolvidos em classificações e categorizações.

Da mesma forma, Singh *et al*. [2024] propuseram um modelo de classificação de defeitos a partir de descrições textuais de falhas. Embora, também não discutiram sobre a estrutura dos dados não estruturados, além da aplicabilidade em outros contextos, visto que foram restringidos a apenas uma base de dados. Sendo assim, essas lacunas revelam a necessidade de investigar sobre os tipos de dados e domínios explorados no estado da arte.

Por outro lado, Laiq *et al*. [2025] realizaram um mapeamento sistemático sobre técnicas automáticas para categorização de bugs, destacando o uso de aprendizado de máquina

e NLP. Eles evidenciam que a maioria dos estudos usa dados de repositórios de código aberto, com pouca validação em ambientes industriais, além de negligenciar aspectos como escalabilidade e generalização. Porém, a diversidade e heterogeneidade dos dados - a qual pode afetar a eficácia dos métodos - não foi observada. Assim, é necessária uma análise sobre a variação entre os domínios e contextos dos projetos utilizados em tarefas de categorização de defeitos.

Por fim, Zheng *et al*. [2024] propôs estratégias de análise e critérios de categorização de defeitos, a fim de explorar o processo de análise de defeitos. Este estudo discute a sua aplicação na prática, propondo métodos e processos básicos para a prevenção de defeitos de software. Contudo, o estudo não fornece detalhes sobre os dados utilizados, como sua origem ou estrutura.

Em síntese, os estudos revisados evidenciam avanços no uso de técnicas de aprendizado de máquina e PLN, porém mantêm limitações quanto à diversidade de dados e contextos de aplicação. Assim, este mapeamento sistemático busca preencher essa lacuna, oferecendo uma visão consolidada sobre as técnicas, bases e classificações utilizadas, bem como apontando novas oportunidades de pesquisa em ambientes corporativos e ágeis.

3 Método

3.1 Planejamento da Revisão

A revisão foi organizada em três fases: planejamento, execução e resultados. O processo foi conduzido com apoio da ferramenta Parsifal, que ajudou no gerenciamento das questões de pesquisa, definição dos critérios de inclusão e exclusão, avaliação da qualidade dos estudos e extração dos dados. A Figura 1 apresenta o processo de execução realizado, seguido de avaliação entre os avaliadores para garantir replicabilidade, consenso e seleção do estudo com base nos critérios.

A escolha pelo Mapeamento Sistemático justifica-se pela intenção de, além de mapear, analisar criticamente as evidências disponíveis, identificar padrões, lacunas e propor direções para pesquisas futuras, o que requer maior rigor na seleção, avaliação e síntese dos estudos.

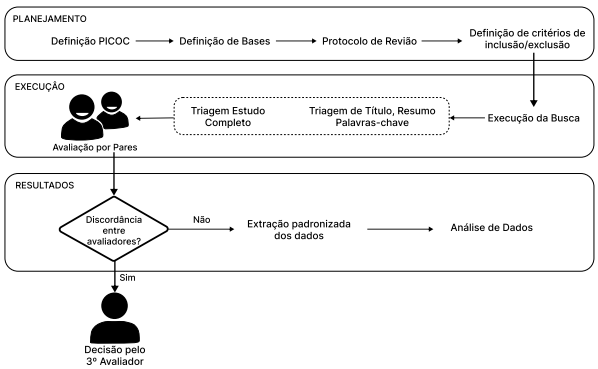

Figura 1. Processo de condução do Mapeamento Sistemático da Literatura.

A metodologia foi colocada conforme o modelo PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses), assim garantindo um processo fácil de replicar e confiável, compreendendo: identificação, triagem e inclusão dos artigos. Iniciando com a busca em bases de dados que resultaram em um grande volume de artigos, que passaram pela remoção de duplicados. Na etapa de triagem, os títulos, resumos e palavras-chave foram avaliados conforme os critérios de inclusão e exclusão. Os artigos selecionados para leitura completa passaram pela análise de qualidade e extração de dados, conduzida por avaliadores autônomos com a intervenção de um terceiro avaliador em situações de discordância. Esse fluxo é ilustrado conforme a Figura 2.

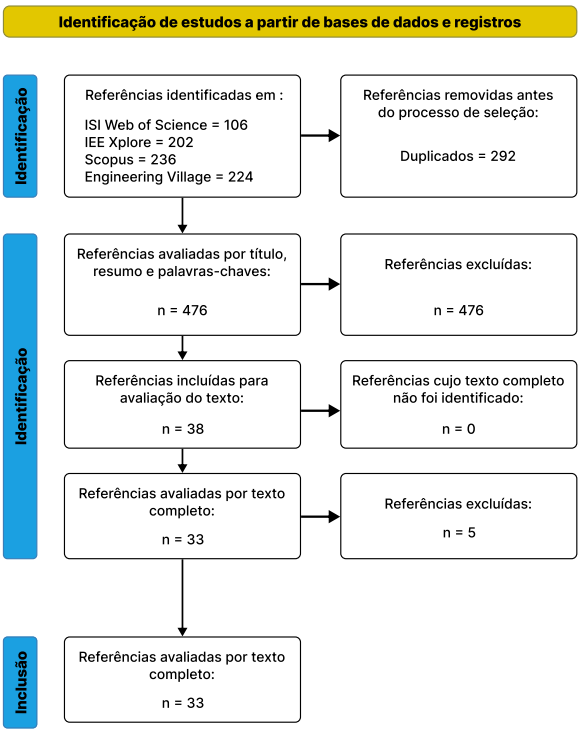

Figura 2. Fluxo PRISMA e resultados durante a etapa

A ferramenta Parsifal foi escolhida devido à sua capacidade de transparência e organização da revisão, assim permitindo registrar o protocolo completo da revisão, incluindo critérios de inclusão e exclusão, questões de pesquisa e formulários de extração de dados. Além disso, a ferramenta oferece integração com o Mendeley e exportação automatizada de relatórios, facilitando o gerenciamento e reduzindo vieses durante o processo de revisão.

3.2 Questões de Pesquisa

As questões de pesquisa foram criadas para identificar: as técnicas utilizadas nos últimos cinco anos para a classificação e categorização de bugs, os contextos de aplicabilidade mais frequentes, as bases propostas nos estudos, métricas de avaliação e as classificações ou categorias adotadas de acordo com as técnicas utilizadas.

RQ1: Quais técnicas foram utilizadas nos últimos cinco anos para a classificação e categorização de bugs? **Motivação:** Com a evolução das técnicas utilizadas para classificação e categorização, é fundamental mapear as mais recentes aplicadas nesse problema.

RQ2: Em quais contextos a classificação e a categorização de bugs são mais aplicadas?
Motivação: Entender em quais contextos são aplicadas

ajuda a entender os ambientes onde essas técnicas têm mais impacto.

RQ3: Quais bases de dados foram utilizadas?
Motivação: A diversidade de bases de dados utilizadas impacta diretamente na qualidade e na generalização dos resultados. Tendo ciência das bases mais utilizadas, é permitido avaliar o reuso dos datasets e a confiabilidade deles.

RQ4: Quais são as classificações e/ou categorias empregadas dadas as técnicas utilizadas?
Motivação: Identificar quais tipos de classificações/categorizações estão sendo utilizadas permite identificar padrões e avaliar a adequação nas técnicas que foram utilizadas.

3.3 Estratégia de Busca

A busca foi realizada nas bases Engineering Village, IEEE Digital Library, ISI Web of Science e Scopus. O recorte temporal foi de 2019 a 2024, definido com base na rápida evolução das técnicas de aprendizado de máquina e processamento de linguagem natural. Excepcionalmente, artigos formalmente aceitos para publicação em 2024, mas com data de publicação final em 2025, foram incluídos para garantir a visão mais atualizada do estado da arte.

A string de busca utilizada foi:

(“bug classification"OR “bug categorization"OR “defect classification"OR “defect categorization") AND (“machine learning"OR “deep learning"OR “natural language processing"OR “NLP") AND (software)

O recorte temporal foi de 2019 a 2024, com filtros para artigos revisados por pares, escritos em inglês, com mais de cinco páginas e disponíveis integralmente.

3.4 Critérios de Inclusão e Exclusão

Foram considerados para inclusão apenas estudos que abordassem classificação ou categorização de bugs em software, apresentassem método, dados e métricas de avaliação, publicados no período definido e que atendessem aos requisitos mínimos de qualidade. Foram excluídos trabalhos focados exclusivamente na detecção ou correção de bugs sem atividades de classificação ou categorização, artigos com dados insuficientes para responder às questões de pesquisa e publicações duplicadas ou versões preliminares de estudos já incluídos.

3.5 Seleção e Avaliação da Qualidade

A busca inicial nas bases resultou em 768 referências (ISI Web of Science: 106, IEEE Xplore: 202, Scopus: 236, Engineering Village: 224). Após a remoção de 292 duplicados, 476 artigos foram triados por título, resumo e palavras-chave.

Na etapa de triagem de texto completo, foram identificados 38 artigos para leitura detalhada. Desses, 5 foram excluídos por apresentarem descrição metodológica incompleta ou ausência de dados necessários, resultando em 33 artigos para a síntese final.

A confiabilidade da triagem foi verificada pelo coeficiente Kappa de Cohen, resultando em um valor de k = 0.80, o que representa uma concordância substancial entre os avaliadores. A qualidade dos estudos foi avaliada considerando: revisão por pares, clareza dos objetivos, descrição metodológica, apresentação de resultados, validade das conclusões e acesso aos dados.

3.6 Extração dos Dados

A extração de dados foi realizada por meio de formulário padronizado na plataforma Parsifal, reunindo informações como: autores, veículo de publicação, técnicas utilizadas, tipo de dados, contexto de aplicação e métricas de avaliação. Dois revisores atuaram de forma independente e, em caso de divergências, um terceiro avaliador decidiu, assegurando a consistência dos dados.

A síntese dos dados foi conduzida por análise descritiva quantitativa (contagem de frequência para algoritmos, bases e contextos) e análise temática narrativa. Esta abordagem mista foi fundamental para atingir o objetivo do mapeamento sistemático, permitindo não apenas a enumeração das tendências, mas também a síntese crítica das evidências, a identificação de padrões emergentes e a discussão das implicações das lacunas identificadas.

4 Resultados

Nesta seção, são apresentados os resultados das perguntas de pesquisa obtidos a partir da análise dos artigos selecionados. As Tabelas 1 e 2 apresentam a distribuição das técnicas utilizadas para classificar e categorizar bugs nos estudos analisados.

Tabela 1. Distribuição das técnicas utilizadas na classificação e categorização de bugs

Rank	Técnica / Modelo	Freq.
1	Naive Bayes	11
2	F-SVM	10
3	MLP (Multi-Layer Perceptron)	9
4	GLM (Generalized Linear Model)	9
5	Random Forest	7
6	Word2Vec	6
7	BERT	5
8	TF-IDF	5
9	Tree-LSTM	5
10	SVM (Support Vector Machine)	5
11	SMOTE	3
12	NLP (Natural Language Processing)	3
13	RDL	3
14	GRCNN	3
15	LSTM	3
16	Gaussian Naive Bayes	2
17	Transformer	2
18	CodeBERT	2
19	DistilBERT	2
20	Multinomial Naive Bayes	2
21	IFSDCR	2
22	CNN (Convolutional Neural Network)	2
23	Fuzzy Linear Regression	2
24	ALBERT	1
25	Polynomial Naive Bayes	1
26	KeyBERT	1
27	Ordinal Regression	1

É notável que grande parte dos artigos analisados emprega algoritmos de Machine Learning (ML) e Deep Learning (DL), mostrando uma tendência de automação na classificação de bugs. O algoritmo Naive Bayes foi aplicado em 11 de 33 artigos (33%), seguido por Fuzzy Support Vector Machine (F-

Tabela 2. Distribuição das técnicas utilizadas na classificação e categorização de bugs. (continuação)

Rank	Técnica / Modelo	Freq.
28	RoBERTa	1
29	Bayesian Classification	1
30	Rule-Based Classification	1
31	Bug Framework	1
32	Feature Vectors	1
33	Multihead-CNN	1
34	Softmax Classifier	1
35	Gradient Boosting	1
36	SDPET	1
37	IFSDTR	1
38	CRF-LSTM	1
39	Term Frequency	1
40	Inverse Document Frequency	1
41	RNN	1
42	ANN	1
43	Decision Tree	1
44	Logistic Regression	1
45	DeepFM	1
46	K-Nearest Neighbor	1
47	Adaboost	1
48	Fuzzy Multi Linear Regression	1
49	Multiple Linear Regression	1
50	Support Vector Regression	1
51	K-Nearest Neighbors	1
52	Regression	1
53	ONN-DP	1

SVM) em 10 artigos (30%) e Support Vector Machine (SVM) em 5 artigos (15%). Já o Multilayer Perceptron (MLP) e o Generalized Linear Model (GLM) apareceram em 9 estudos cada (27%). A persistência de classificadores tradicionais (Naive Bayes, SVM) reflete sua interpretabilidade, baixo custo computacional e desempenho como baseline robusto em tarefas textuais.

Além disso, observou-se o crescimento do uso de modelos pré-treinados, como Bidirectional Encoder Representations from Transformers (BERT) e suas variações CodeBERT, DistilBERT, RoBERTa e ALBERT, os quais foram empregados em 11 artigos (33%), indicando uma adoção crescente de representações semânticas profundas. As técnicas de pré-processamento também foram frequentemente utilizadas, com destaque para TF-IDF em 5 artigos (15%), Word2Vec em 6 artigos (18%) e SMOTE em 3 artigos (9%).

Em resumo, vê-se que a classificação e categorização de bugs têm sido comumente abordadas por meio de algoritmos supervisionados, com grande parte baseada em modelos de ML e DL, acompanhadas de um crescente interesse em técnicas de processamento textual. Essa tendência indica uma busca maior pela automação da classificação e categorização de bugs, além da melhoria da qualidade da análise semântica, o que pode auxiliar na eficiência do processo de qualidade e manutenção de software. A Figura 3, apresenta a distribuição dos tipos de dados que foram analisados na classificação e/ou categorização de bugs dos estudos analisados.

De acordo com a Figura 3, é notável que a maioria dos artigos utiliza dados extraídos de Bug Reports (55%), títulos (36%), resumos (39%) e descrições (30%). Essa forte

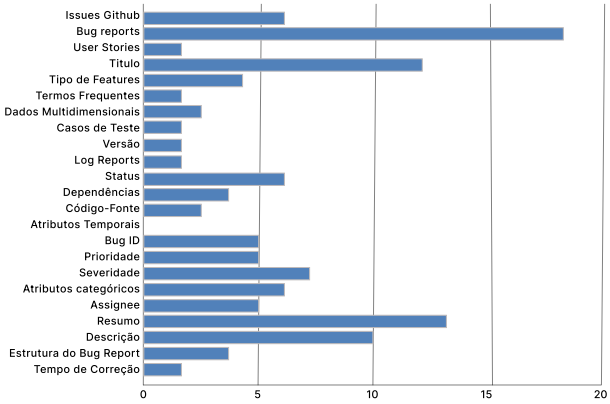

Figura 3. Distribuição dos tipos de dados utilizados na classificação e categorização de bugs.

concentração em dados textuais reforça o uso de técnicas de Processamento de Linguagem Natural (PNL) e aprendizado supervisionado. A análise crítica aponta que a excessiva dependência de dados textuais faz com que o sucesso da classificação esteja intrinsecamente ligado à volatilidade semântica e à qualidade da anotação humana em textos curtos e não padronizados.

Ademais, diversos artigos utilizam dados categóricos, tais como severidade (7 artigos, 21%), prioridade (5 artigos, 15%), status (6 artigos, 18%) e assignee (5 artigos, 15%), que complementam as análises textuais e permitem avaliações mais amplas e detalhadas.

Por outro lado, atributos temporais e tempo de correção são pouco explorados. Este é um achado crítico: negligenciar a dimensão temporal limita severamente a compreensão do ciclo de vida completo dos bugs e do desempenho operacional da triagem. Em ambientes corporativos, o tempo de correção é uma variável-chave para a priorização e alocação de recursos, e sua ausência nos modelos revisados representa uma barreira significativa para a transferência e a validade externa dessas soluções. Esse resultado mostra que, embora o uso de dados textuais seja frequentemente utilizado, há espaço para estudos futuros que envolvam dimensões temporais e contextuais, assim ampliando a visão sobre classificação e categorização de bugs.

Esse resultado mostra que, embora os dados textuais sejam recorrentemente empregados, há espaço para estudos futuros que considerem dimensões temporais, assim ampliando a visão sobre classificação e categorização de bugs. De forma geral, observa-se que as técnicas que foram utilizadas nos últimos cinco anos para classificação/categorização de bugs determinam o uso de algoritmos supervisionados com técnicas de PLN aplicadas em dados textuais.

Essa aproximação entre ML, modelos pré-treinados e abordagens de pré-processamento textual mostra uma busca por automação e escalabilidade em atividades de classificação. A predominância de dados provenientes de bug reports evidencia a centralidade das fontes textuais estruturadas e não estruturadas. Apesar da limitada exploração de dados temporais e atributos como tempo de correção, mostram-se oportunidades futuras para ampliação da compreensão do ciclo de vida de bugs. A Tabela 3, apresenta a distribuição dos contextos de aplicação que foram abordados nos estudos

analisados.

Tabela 3. Contextos de aplicação abordados nos estudos

Contexto	Número de Ocorrências
Projetos GitHub	9
Projetos Open-source	25
Projetos Cybersecurity	1
Projetos SAAS	1
Projetos Mobile	1
Projetos de Software	1
Projetos Privados	5
Projetos Java	1
Projetos em fase de testes	8
Projetos Ágeis	1
Projetos JavaScript	1

Conforme a Tabela 3, a maioria dos artigos foi conduzida em contexto de projetos open-source (76%) e projetos hospedados no GitHub (27%), o que decorre principalmente da disponibilidade pública dos dados. Em contrapartida, projetos privados (15%) e projetos ágeis (apenas 3%) têm baixa representatividade. Este cenário reforça a ameaça à validade externa (Seção 4.1), pois a grande dependência de projetos open-source limita a generalização para cenários corporativos, onde a cultura de comunicação, a estrutura do bug report e o viés organizacional são fatores críticos.

Além disso, observa-se a presença de projetos em fase de testes em 8 estudos (24%), evidenciando o interesse na avaliação de técnicas em ambientes não consolidados. Em contrapartida, vê-se que artigos que utilizaram projetos privados somam 5 artigos (15%), apresentando limitações de acesso e restrições. Além de contextos menos explorados, tais como segurança cibernética, SaaS, mobile, Java e JavaScript, com 1 artigo (3%) cada contexto. Vale ressaltar que apenas 1 artigo (3%) citou a aplicação em projetos ágeis, configurando uma possível lacuna na literatura sobre a adaptação dessas técnicas em contextos ágeis.

Em resumo, os resultados abrangem aplicação em projetos abertos e acessíveis, reforçando a importância de repositórios públicos como base de análises científicas. Esse cenário reforça a importância da disponibilização de dados reais e evidencia oportunidades de exploração em ambientes menos estudados, como ambientes corporativos e projetos ágeis. A Tabela 4, apresenta a distribuição das bases de dados utilizadas nos estudos analisados.

Conforme a Tabela 4, Mozilla e Eclipse aparecem com maior frequência (39% cada), o que reforça o alto alinhamento com a ciência aberta. No entanto, essa concentração limita a diversidade de contextos, indicando a necessidade de exploração de novas bases de dados de domínios pouco representados. A predominância de dados não estruturados (84,8%) (Figura 6) é coerente com o foco em PLN, mas a escassez de dados estruturados (15,2%) sugere a oportunidade de modelos híbridos mais robustos que combinem dados textuais com atributos categóricos como prioridade e severidade.

Outras bases utilizadas foram NetBeans, em 6 artigos (18%), e Apache, Bugzilla e Firefox, cada uma utilizada em 3 artigos (9%), reforçando o uso dessas bases em projetos maduros e bem estabelecidos. Outra base utilizada foi o GitHub, presente em 4 estudos (12%) com datasets específicos, enquanto bases privadas aparecem em 5 artigos (15%),

Tabela 4. Bases de dados utilizadas nos estudos

Base	Número de Ocorrências
GitHub dataset	4
Dataset private	5
Nasa Dataset	1
BugJs	1
Kaggle dataset	2
F-Droid	1
Bugzila	3
JBoss	1
OpenFOAM	1
Firefox	3
Mozilla	13
Eclipse	13
CVE	1
Gentoo	1
Gnome	1
Open office	2
NetBeans	6
Apache	3
KDE	1
LibreOffice	1
Linux	1
Thunderbird	1
Seamonkey	1
Boot2Gecko	1
Add-OnSDK	1
Webtools	1
Addons.mozilla.org	1
Camel	1
Derby	1
Wicket4	1

indicando um interesse em cenários corporativos, embora o acesso a essas bases apresente limitações.

Além disso, observou-se menor utilização de fontes como Kaggle, em 2 artigos (6%), e OpenOffice, em 2 artigos (6%), bem como de bases menos consolidadas, como NASA Dataset, CVE, F-Droid, Gentoo, Gnome, Linux, KDE, entre outras, cada uma mencionada em 1 artigo (3%). Esse cenário evidencia uma grande dispersão em repositórios pouco representativos, o que reforça a necessidade de diversificar as fontes de dados utilizadas em futuras pesquisas.

Os resultados mostram que a comunidade acadêmica tem, de forma expressiva, utilizado bases de projetos open-source, dada a acessibilidade e a grande quantidade de dados. Porém, essa concentração limita a diversidade de contextos, mostrando a necessidade de exploração de novas bases de dados de domínios pouco representados. A Figura 4, apresenta a distribuição da disponibilidade dos datasets utilizados nos estudos analisados.

Conforme a Figura 4, observa-se que a grande maioria dos artigos utiliza datasets de disponibilidade pública, presentes em 28 de 33 estudos (84.8%). Essa predominância mostra o forte alinhamento da comunidade científica com a ciência aberta, facilitando a replicação e a melhoria de experimentos, a comparação entre estudos e o avanço do conhecimento na área de classificação e categorização de bugs.

Por outro lado, há 5 artigos (15.2%) que utilizaram datasets privados, geralmente provenientes de contextos corpo-

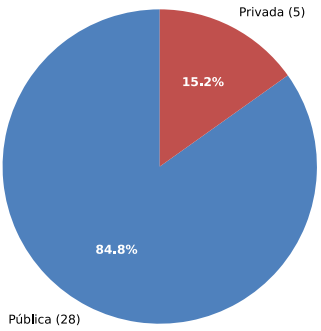

Figura 4. Distribuição da disponibilidade dos datasets analisados.

rativos ou organizações privadas. Embora representem um retrato fiel de cenários reais, sua restrição limita a reprodução e o avanço de experimentos, impossibilitando comparações e validações. Esse cenário indica uma grande abertura da comunidade científica em compartilhar dados, assim evoluindo pesquisas. A Figura 5, apresenta a distribuição do tipo de estrutura utilizada nos datasets dos estudos analisados.

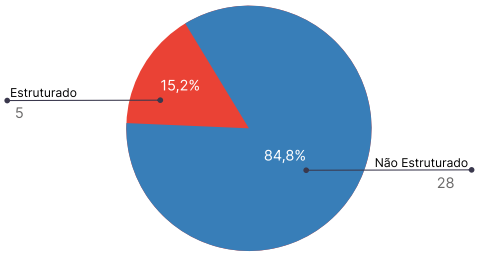

Figura 5. Distribuição do tipo de estrutura dos datasets analisados.

A análise dos artigos revela uma predominância no uso de dados não estruturados, presentes em 84,8% (28 de 33) dos estudos. Esse dado é coerente com a aplicação de técnicas de PLN, que se mostram eficazes na extração de informações textuais para a classificação. Entretanto, apenas 15,2% (5 artigos) utilizaram dados estruturados, como prioridade, severidade e status, que se mostram mais fáceis de manipulação em modelos supervisionados. Esse cenário evidencia uma tendência da área na exploração de dados textuais, ao mesmo tempo que mostra uma oportunidade de combinação de modelos híbridos mais robustos. A Tabela 5, apresenta a distribuição do tipo de classificação e/ou categorização empregada nos estudos analisados.

De acordo com a Tabela 5, observa-se uma grande diversidade de classificações e categorizações de bugs empregadas nos estudos analisados. Dentre as mais recorrentes, destaca-se a classificação multiclasse, presente em 12 de 33 artigos (36%), a qual permite que os modelos atribuam os bugs a duas ou mais categorias, refletindo cenários mais realistas e complexos.

Além disso, há a classificação por tipo de bug em 4 artigos (12%), demonstrando a preocupação em evidenciar as diferentes naturezas dos bugs. Outras abordagens de destaque

Tabela 5. Tipos de Classificação e Categorização de Bugs Identificados

Tipo de Classificação / Categorização	Frequência
Classificação multiclasse	12
Classificação por severidade	10
Classificação de duplicidade de bugs	7
Classificação por tipo de bug	4
Classificação baseada em relato detalhado	3
Classificação por tipo técnico	2
Classificação baseada em risco	2
Classificação binária	1
Classificação por prioridade	1
Classificação por similaridade	1
Classificação balanceada	1
Classificação por testes	1
Classificação multirrótulo	1

são: classificação por severidade, com 10 artigos (30%); por prioridade, com 1 artigo (3%); e por tipo técnico, com 2 artigos (6%), voltadas para a categorização dos bugs conforme a criticidade e/ou o impacto no sistema.

Abordagens menos frequentes incluem a classificação binária, com 1 artigo (3%); baseada em risco, com 2 artigos (6%); por similaridade, com 1 artigo (3%); e multirrótulo, com 1 artigo (3%). Em menor número, têm-se ainda classificações balanceada, por testes, por duplicidade de bugs e categorização baseada em relatos detalhados, todas com 1 artigo (3%).

Esses dados indicam que a grande maioria dos artigos busca uma categorização mais rica e contextualizada, indo além da rotulação binária, o que sugere uma maturidade na área e a possibilidade de refletir contextos reais do desenvolvimento de software. Ao mesmo tempo, a presença de abordagens diversas evidencia que a classificação de bugs é um campo dinâmico e repleto de possibilidades de estudo em aberto.

4.1 Ameaças à Validade

Esta revisão apresenta algumas limitações e ameaças à validade que devem ser consideradas na interpretação dos resultados. Em relação à validade da construção, embora a string de busca tenha sido elaborada de forma ampla, é possível que nem todos os estudos relevantes tenham sido recuperados devido a variações terminológicas na literatura.

Quanto à validade interna, a avaliação de qualidade e a extração dos dados foram conduzidas por dois revisores independentes, com intervenção de um terceiro em casos de divergência, mas ainda assim podem ter ocorrido interpretações subjetivas. No que se refere à validade externa, a predominância de estudos baseados em dados provenientes de projetos open-source limita a generalização das conclusões para contextos corporativos e ágeis.

Por fim, no tocante à validade de conclusão, as análises realizadas dependem das informações fornecidas pelos estudos primários, que podem conter vieses de publicação ou ausência de detalhes metodológicos. Para mitigar tais ameaças, foram aplicados protocolos estruturados, como o PRISMA, e calculado o coeficiente Kappa, visando assegurar maior confiabilidade ao processo de seleção e análise.

5 Conclusão

Os resultados da análise dos 33 artigos selecionados indicam que a classificação e categorização de bugs é um campo explorado na área acadêmica. Há uma predominância de algoritmos de Machine Learning (ML) e Deep Learning (DL), como Naive Bayes, SVM, MLP e modelos pré-treinados, como BERT e suas variantes. Essa tendência reflete a busca por automação e melhoria da eficiência no processo de manutenção de software. A aplicação de técnicas de pré-processamento textual e o foco em dados não estruturados (descrições, resumos e títulos de bugs) reforçam a centralidade do Processamento de Linguagem Natural (PNL).

As fontes de dados predominantes são repositórios open-source, com destaque para Mozilla e Eclipse, evidenciando a facilidade de acesso a informações dessas bases. Contudo, essa concentração limita a diversidade de cenários analisados, destacando a necessidade de explorar fontes privadas e dados corporativos. Embora a utilização de datasets públicos demonstre o alinhamento da comunidade científica com os princípios da ciência aberta, a menor presença de bases privadas aponta para uma lacuna na representação de dados reais, o que restringe a generalização dos resultados para ambientes corporativos.

A respeito de contextos de aplicação, grande parte dos estudos se concentrou em projetos open-source e hospedados no GitHub, tendo uma baixa representatividade em ambientes corporativos, projetos ágeis, segurança cibernética e desenvolvimento mobile, indicando lacunas promissoras para pesquisas futuras. Bem como, a maioria dos artigos foca em dados textuais, com baixa exploração de atributos temporais e tempo de correção de bugs, limitando a compreensão do ciclo de vida de bugs.

No que tange os tipos de classificação e categorização, os artigos demonstram o uso de abordagens como classificação multiclasse, por tipo de bug, severidade e prioridade, mostrando uma tentativa de replicação da complexidade dos cenários reais no desenvolvimento de software. Todavia, este campo mostra-se dinâmico, com abordagens menos frequentes que revelam novas possibilidades de inovação.

Os resultados desta revisão podem orientar empresas de software, gestores de qualidade e pesquisadores na escolha de técnicas de classificação e categorização mais adequadas aos seus contextos. Recomenda-se que pesquisas futuras explorem modelos híbridos que combinem dados estruturados e não estruturados, incorporando atributos temporais e contextuais para ampliar a aplicabilidade prática das técnicas. A integração de modelos baseados em LLMs (Large Language Models) com dados de bugs reais também representa uma direção promissora para aumentar a precisão e a interpretabilidade dos resultados.

Declarações complementares

Financiamento

Esta pesquisa não foi financiada por nenhum órgão governamental e não-governamental.

Contribuições dos autores

Ramsés contribuiu para a concepção deste estudo. Samuel, Laís e Rubens realizaram a investigação, a metodologia, a escrita e a revisão. Manoel e Alex foram responsáveis pela curadoria dos dados, administração de projetos e validação do estudo. Todos os autores leram e aprovaram o manuscrito final.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

Os conjuntos de dados (e/ou softwares) gerados e/ou analisados durante o estudo atual serão feitos mediante solicitação.

Referências

Afric, P., Vukadin, D., Silic, M., and Delac, G. (2023). Empirical study: How issue classification influences software defect prediction. *IEEE Access*, 11:11732–11748. DOI: 10.1109/ACCESS.2023.3242045.

Gunalan, P., Jaiseenu, V., and Madumidha, S. (2022). Natural language processing based bug categorization. In *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 1859–1863. IEEE. DOI: 10.1109/ICAAIC53929.2022.9793205.

Harzevili, N. S., Belle, A. B., Wang, J., Wang, S., Jiang, Z. M. J., and Nagappan, N. (2025). A systematic literature review on automated software vulnerability detection using machine learning. *ACM Computing Surveys*, 57:1–36. DOI: 10.1145/3699711.

Laiq, M., bin Ali, N., Börstler, J., and Engström, E. (2025). A comparative analysis of ml techniques for bug report classification. *Journal of Systems and Software*, 227:112457. DOI: 10.1016/j.jss.2025.112457.

Meher, J. P., Biswas, S., and Mall, R. (2024). Deep learning-based software bug classification. *Information and Software Technology*, 166:107350. DOI: 10.1016/j.infsof.2023.107350.

Ordoñez-Pacheco, R., Cortes-Verdin, K., and Ocharán-Hernández, J. O. (2021). *Best Practices for Software Development: A Systematic Literature Review*, pages 38–55. DOI: 10.1007/978-3-030-63329-5_3.

Pachouly, J., Ahirrao, S., Kotecha, K., Selvachandran, G., and Abraham, A. (2022). A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools. *Engineering Applications of Artificial Intelligence*, 111:104773. DOI: 10.1016/j.engappai.2022.104773.

Priyadarshini, A. (2025). Software defect prediction across multiple datasets using classification techniques. *Journal of Information Systems Engineering and Management*, 10:158–167. DOI: 10.52783/jisem.v10i41s.7807.

qahtan yas, Alazzawi, A., and Rahmatullah, B. (2023). A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions. *Iraqi Journal for Computer Science and Mathematics*, pages 173–190. DOI: 10.52866/ijcsm.2023.04.04.014.

Singh, K., Aggarwal, A., Aneja, R. D., Kumar, R., and Soni, P. (2024). Deep learning models for software defect classification. In *2024 2nd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, pages 1–4. IEEE. DOI: 10.1109/DICCT61038.2024.10532830.

Zheng, L., Xie, Y., Hou, B., and Xie, X. (2024). Software

defect classification and analysis study. In *2024 5th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pages 621–624. IEEE. DOI: 10.1109/ISPDS62779.2024.10667506.