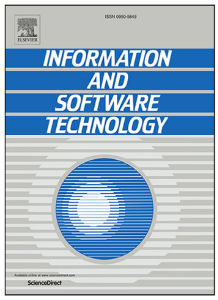
Journal Pre-proof



Open source oriented cross-platform survey

Simeng Yao, Xunhui Zhang, Yang Zhang, Tao Wang

PII: S0950-5849(25)00043-6
DOI: https://doi.org/10.1016/j.infsof.2025.107704
Reference: INFSOF 107704

To appear in: *Information and Software Technology*

Received date : 12 September 2024
Revised date : 26 January 2025
Accepted date : 24 February 2025

Please cite this article as: S. Yao, X. Zhang, Y. Zhang et al., Open source oriented cross-platform survey, *Information and Software Technology* (2025), doi: https://doi.org/10.1016/j.infsof.2025.107704.

Highlights

Open Source Oriented Cross-platform Survey

Simeng Yao,Xunhui Zhang,Yang Zhang,Tao Wang

- We focus on summarizing the current state of development of open source oriented cross-platform research.
- We summarize the datasets, research methods, etc. used in the existing literature.
- We propose 6 future directions for cross-platform research in open source and provide corresponding recommendations for developers, researchers, and service/tool providers.

Open Source Oriented Cross-platform Survey

Simeng Yao, Xunhui Zhang, Yang Zhang and Tao Wang*

[a]College of Computer Science and Technology, National University of Defense Technology, Changsha, 410073, China
[b]State Key Laboratory of Complex & Critical Software Environment, Changsha, 410073, China

ARTICLE INFO

Keywords:
Open Source
Cross-platform
Systematic literature review
GitHub
StackOverflow
Twitter

ABSTRACT

Context: Open-source software development has become a widely adopted approach to software creation. However, developers' activities extend beyond social coding platforms (e.g., GitHub), encompassing social Q&A platforms (e.g., StackOverflow) and social media platforms (e.g., Twitter). Therefore, cross-platform research is essential for a deeper understanding of the nature of software development activities.

Objective: This paper focuses on open-source platforms and systematically summarizes relevant cross-platform research. It aims to assess the current state of cross-platform research and provide insights into the challenges and future developments in this field.

Method: This paper reviews 69 cross-platform research papers related to open-source software from 2013 to 2024, with a focus on several key areas, including platform interconnections, research themes, experimental design methods, challenges and rersearch opportunities.

Results: Through the analysis of 69 papers, we found that cross-platform research primarily involves platforms such as social coding, social Q&A, and social media. Researchers typically rely on information traces, including user personal info, technical info, project/post/bug report metadata, interaction info, to facilitate connections between platforms. Cross-platform research in the open-source domain mainly focuses on problem classification and feature extraction. The predominant research methods include data-driven approaches, qualitative studies, modeling and machine learning, and tool development and implementation. Despite these advancements, common challenges remain, such as subjective evaluation bias in manual data classification, insufficient data source coverage, and inaccurate data recognition. Future research opportunities may focus on increasing the diversity of data sources, improving data recognition accuracy, optimizing data classification methods, and clarifying user skill requirements.

Conclusions: Based on our findings, we propose six future directions for cross-platform research in the open-source domain and provide corresponding recommendations for developers, researchers, and service/tool providers.

1. Introduction

Traditional software development methods often rely on closed teams and internal resources, leading to longer development cycles and innovation constrained by the size and experience of the development team. In contrast, platform-based development models, by opening code repositories, allow developers worldwide to contribute and share code, thereby significantly enhancing development efficiency and innovative capabilities. With the rise of social coding platforms such as GitHub and GitLab, the shift from traditional closed development to platform collaboration has been greatly accelerated [1]. As of 2024, the GitHub platform has more than 100 million registered users and over 420 million repositories[1], forming a vast software development social interaction network that greatly promotes code sharing and collaborative development. Although social coding platforms include issue trackers and support code review, developers often do not limit their activities to these platforms alone due to individual preferences and differences in platform focus [2], but instead engage in information sharing and development collaboration in broader platforms. For example, the social Q&A platforms (e.g., Stack Overflow) provides developers with a platform for sharing knowledge and solving programming problems [3], enabling them to obtain rapid knowledge-sharing and problem-solving services that are difficult to provide through social coding platforms[4], and then apply software development solutions to project repositories. By combining social Q&A platforms with social coding platforms, developers can collaborate efficiently on multiple platforms, sharing knowledge and facilitating the progress in actual code development. Additionally, the social media platform (e.g., Twitter) has become an important platform for disseminating information and learning about new technologies, helping developers stay informed about industry trends [5], and promoting projects within social coding platforms to a broader audience. Subsequently, these projects can attract more software developers to join the project's sustained development and collaboration[6]. Therefore, developers are increasingly relying on these social media and social Q&A platforms to communicate and resolve issues, effectively aiding developers in addressing various challenges in project development[7, 8].

Although significant progress has been made in single-platform studies, such as GitHub collaboration analysis and Stack Overflow knowledge-sharing modeling [9], fundamental limitations persist in addressing the growing demand

*Corresponding author
E-mail addresses:yaosimeng23@nudt.edu.cn(S.Yao), zhangxunhui@nudt.edu.cn(X.Zhang), yangzhang15@nudt.edu.cn(Y.Zhang), taowang2005@nudt.edu.cn (T. Wang)
ORCID(s):

[1] https://github.com/about

for interconnectedness in software development. The necessity of cross-platform research is contingent upon whether the research question involves multi-source data dependencies or the analysis of behavioral heterogeneity. Within the domain of open-source software (OSS), the single-platform perspective exhibits notable shortcomings in the following scenarios:

Data Incompleteness: Developer activities naturally span multiple technical platforms (e.g., code commits on GitHub, answering questions on Stack Overflow, and defect tracking on Bugzilla), forming a technical collaboration ecosystem. Single-platform data captures only partial behaviors, resulting in critical information loss. For instance, Song et al. [2] quantified expertise using an expertise matrix and found that single-platform developer expertise sparsity (SPE) ranged from 0.95 to 0.97, whereas cross-platform joint modeling reduced SPE to 0.9351. Similarly, Hong et al. [10] demonstrated that integrating data from GitHub, Bugzilla, and Stack Overflow could improve patch coverage by 400%.

Systemic Risks: In practice, developers not only reuse code within a single platform but also frequently source problem-solving solutions from platforms like Stack Overflow to support project development [11]. Studies have shown that single-platform code reuse can lead to various issues, including potentially harmful code snippets [12], copyright violations [13], and code modifications [14]. In contrast, cross-platform research, by integrating data from multiple platforms, offers a more comprehensive approach to identifying and addressing these risks.

Behavioral Complexity: Developer behaviors exhibit significant heterogeneity due to differences in platform functionalities. Han et al. [15] highlighted notable variations in knowledge focus across platforms, while Wu et al. [16], through developer interviews, revealed that respondents perceived GitHub's social features as limited and preferred using social media platforms like Twitter for technical interactions.

These challenges fundamentally pertain to the core issues of code quality governance and collaboration efficiency optimization in the field of software engineering. Through a systematic literature review, this study extracts key themes and methodologies in cross-platform research, providing a methodological foundation for constructing a more robust open-source collaboration ecosystem. As cross-platform dependencies in software development continue to intensify, such research holds strategic significance for risk mitigation and efficiency enhancement.

This paper answers the following research questions:

RQ1: How are different platforms connected in cross-platform studies?

The goal of this research question is to explore how various platforms are interlinked in cross-platform studies. This helps researchers identify the relevant connections between platforms and understand the types of information traces used to establish these connections. We found that, in the cross-platform research domain, the primary focus is on the connections between three types of platforms: social coding platforms (e.g., GitHub), social Q&A platforms (e.g., StackOverflow), social media platforms (e.g., Twitter). The connections between these platforms mainly rely on information traces such as user personal info, technical info, project/post/bug report metadata, interaction info.

RQ2: What are the major topics in cross-platform studies?

The goal of this research question is to identify the key issues addressed by cross-platform studies, which in turn reveals the emerging trends in research topics and the motivations behind these studies. We found that cross-platform research primarily focuses on five major themes: problem classification and feature extraction, platform collaboration, code reuse and evolution, user characterization, and cross-platform data optimization. Among these, problem classification and feature extraction is the most prominent research area, with numerous studies exploring how to identify and extract relevant features across multiple platforms to improve the understanding of developer behavior and software development processes.

RQ3: How to design experiments for cross-platform studies?

The goal of this research question is to help researchers quickly understand the datasets and research methods used in related studies. We have collected and organized a total of 40 publicly available datasets, along with some outdated datasets. Regarding research methods, the primary approaches include Data-Driven Methods, Modeling & ML Approaches, Tool Development and Implementation, and Qualitative Studies. The findings highlight a dominance of Data-Driven Methods, a significant trend of method integration, and the rise of intelligent methods exploration.

RQ4: What are the key challenges and research opportunities identified in the existing literature?

This research question aims to identify the key challenges and research opportunities highlighted in the existing literature on cross-platform studies, with the aim of guiding future research directions. Our findings indicate that several common challenges persist across various research themes, including subjective evaluation bias in manual data classification, insufficient data source coverage, and inaccurate data recognition. Moreover, research opportunities emphasize the need to enhance the diversity of data sources, improve data recognition accuracy, optimizing data classification methods, and clarifying user skill requirements.

The main contributions of this study are as follows:

- We conducted a systematic review of 69 papers published between 2013 and 2024, providing valuable guidance for researchers engaged in cross-platform studies.
- We compiled a comprehensive list of 40 publicly available datasets used in cross-platform research, including detailed information such as dataset names, scale, timeframes, access links, application.

- Based on the opportunities and challenges identified in existing cross-platform research, we propose six potential future research directions and provide recommendations for developers, researchers, and service/tool providers.

Every aspect of our research process is available for replication at [17] and [18].

The remainder of this paper is structured as follows: Section 2 provides an overview of the background. Section 3 outlines the study design. In Section 4, we present the results. In Section 5, we discuss the key findings, propose a future agenda for cross-platform studies, and provide practical recommendations. In Section 6, we analyze potential threats to the validity of this survey. And finally we conclude the paper in Section 7.

2. Background

With technology constantly advancing and social interaction methods diversifying, the engagement and interaction strategies of software developers are undergoing significant evolution. To comprehensively analyze this intricate phenomenon, exploring the interaction dynamics between social coding platforms (e.g., GitHub, GitLab), social media platforms like Twitter, and social Q&A platforms like StackOverflow is essential. In this section, we will outline the development of open source platforms, and emphasize the core value and broad significance of cross-platform research.

2.1. The Development of Open Source platforms

Since the emergence of the Git tool in 2005, a multitude of social coding platforms based on Git have emerged, such as GitHub and GitLab. These coding platforms have attracted a large number of developers to participate in open source contributions, supporting the continuous construction of large-scale software projects. To date, GitHub has attracted over 100 million individual developers and 400 million organizations to participate, and has hosted over 420 million repositories [19].

As a novel paradigm, open source software development has many advantages. On one hand, the active engagement of numerous developers fuels rapid software iteration [20]. On the other hand, the extensive platform involvement accelerates the discovery of vulnerabilities, thereby enhancing software quality [21, 22]. Moreover, the adoption of reuse-based software development methodology, coupled with the contributions of platform volunteers, significantly reduces both development and maintenance costs [23].

As the open source ecosystem flourishes, many mechanisms have emerged to facilitate high-efficiency collaboration among developers and to ensure the high-quality, iterative evolution of software. For example, the issue tracking and associated discussion forums, along with the milestone feature, empower contributors of various kinds to articulate their needs, engage in project-related dialogues, and set incremental project goals. The introduction of the pull-based model has allowed developers from the periphery to actively participate in the coding process, with core team members maintaining a quality checkpoint, thereby enhancing the efficacy of collaborative development efforts [24]. On top of this foundation, continuous integration systems have been integrated to ensure contribution quality and streamline the review process through automated testing [25]. Tools such as GitHub Action and bot mechanisms are all part of an automated approach to improving the efficiency and quality of software development [26, 27]. Assignment [28], @mention [29], and linking [30] mechanisms serve to connect developers with software artifacts, enabling them to swiftly identify information pertinent to their interests. platforms like GitHub have opened up access to a vast amount of data through APIs, giving rise to popular datasets such as GHTorrent [31] and GHArchive [2], which have significantly propelled research in the realm of open source development.

While a large number of tools and collaborative frameworks are furnished by social coding platforms to facilitate collaboration, the distinct emphases of different platforms, along with the disparities in user experiences and customary practices, lead to varied levels of participation and contribution across different platforms types. Taking StackOverflow as an example, social coding platforms feature mechanisms such as issues and discussions. However, many developers and software users still prefer to pose questions related to open-source software on StackOverflow. This preference is partly attributed to the fact that a significant number of issues on social coding platforms remain unanswered and are subsequently closed. In contrast, StackOverflow is maintained by dedicated individuals who provide timely responses, ensuring that inquiries are resolved swiftly [4]. Similarly, although social coding platforms allow commenting, they do not ensure the immediacy [32]. This is why many developers prefer to use Discord.

Therefore, it is likely that OSS participants will leave active traces across different types of platforms. The interconnection between platforms, on one hand, can enrich developer information, providing a comprehensive understanding of developers and enabling the construction of personalized services. On the other hand, it allows for the connection of various pieces of information related to open-source software, leading to a thorough understanding of the platform's development.

Next, we will describe the core value of cross-platform research in detail.

2.2. The Core Value of Cross-platform Research

In the current academic and practical landscape, cross-platform research holds significant value and far-reaching implications. Developers are not only active in social coding platforms but also contribute and engage in activities on platforms such as StackOverflow and Twitter. Consequently, cross-platform research not only provides a more comprehensive perspective but also addresses challenges and limitations specific to individual platforms.

[2] https://www.gharchive.org/

For instance, in the analysis of user behavior, cross-platform research offers several benefits. Firstly, cross-platform research contributes to addressing the issue of data sparsity. In a single platform, limited data may hinder the acquisition of comprehensive and accurate information. For instance, as highlighted in the work of Zhao et al.[33], user activity data within a single platform is often limited, resulting in the sparsity of relationship networks. By integrating data from multiple platforms, researchers can obtain richer and more comprehensive user behavior and interaction information, thereby enhancing the accuracy and reliability of studies. Secondly, cross-platform research helps overcome individual and platform preference issues. Each platform possesses unique cultures, rules, and interaction patterns, potentially leading developers to exhibit different skills and behaviors in one platform compared to others. For example, Song et al.[2] points out that developers may demonstrate limited skills in a particular platform due to personal preferences or specific platform rules. Through cross-platform research, a more holistic understanding of developers' actual skills and potential can be gained, mitigating biases resulting from reliance on a single platform. Additionally, cross-platform research contributes to addressing the cold start problem. New users may lack sufficient historical data and interaction records in a specific platform, rendering traditional recommendation and suggestion systems ineffective in providing accurate information. For instance, Yuan et al.[34] extensively explores the impact of the cold start problem in recommendation systems.

By integrating data and information from multiple platforms, more accurate and personalized recommendations can be provided for new users, enhancing user experience and satisfaction.

2.3. Summary

The rapid development of the open-source ecosystem has attracted a significant number of developers. However, due to varying focuses, differences in user experience, and usage habits, other platforms, including social Q&A platforms and social media platforms, are also widely utilized by OSS participants, generating a wealth of behavioral data.

Cross-platform research not only offers a more comprehensive and accurate perspective on software development but also addresses critical issues such as data sparsity, individual and platform preferences, and the cold start problem in developer engagement. As software developers continue to participate and contribute across multiple platforms, the significance and value of such research become increasingly evident.

This study aims to conduct a comprehensive cross-platform exploration of the open-source ecosystem. By providing a systematic review of existing research, it shows the interaction patterns and knowledge-sharing mechanisms of developers across diverse platforms. Furthermore, it explores potential future research opportunities and challenges, offering the academic platform and relevant researchers a nuanced and in-depth perspective. This approach enables them to more accurately capture and respond to the complex and increasingly diversified software development ecosystem.

3. Study Design

In this section, we followed the Systematic Literature Review (SLR) methodology proposed by Kitchenham et al. [35] and Petersen et al. [36] to construct the research framework (as shown in Figure 1). The entire process is divided into two main phases: Study Identification and Study Selection. In the study identification phase, we first defined the research questions (Step 1). Next, we selected major databases, such as IEEE Xplore, ACM Digital Library, and Scopus, as the primary sources for literature retrieval (Step 2). Subsequently, we designed an initial search string, which was refined through pilot searches to determine the final search string (Step 3). Using this search string, we retrieved papers from the selected databases and removed duplicates during the process (Step 4). In the study selection phase, we applied predefined selection criteria to conduct a detailed review of the initially retrieved papers (Step 5), focusing on the titles, abstracts, and conclusions to exclude irrelevant or low-quality studies. At this stage, 161 papers were retained and further subjected to a quality assessment (Step 6). Finally, 69 high-quality papers were selected. Data from these papers were extracted and aligned with the research questions, forming the basis for subsequent in-depth analysis (Step 7).

3.1. Research questions

The primary objective of this study is to analyze the current state of cross-platform research in the context of open source.To achieve this goal, we first define cross-platform as follows: Cross-platform refers to developer activities that span heterogeneous technical infrastructures (e.g., GitHub for code collaboration and Stack Overflow for knowledge sharing) and meet the following two criteria:

Functional Heterogeneity: Platforms must serve distinct technical roles.

Data Linkability: Behaviors must be traceable across platforms via explicit methods (e.g., user ID matching) or implicit methods (e.g., semantic alignment).

Based on this definition, we propose the following research question(s):

RQ1: How are different platforms connected in cross-platform studies?

RQ2: What are the major topics in cross-platform studies?

RQ3: How to design experiments for cross-platform studies?

RQ4: What are the key challenges and research opportunities identified in the existing literature?

3.2. Database selection

We selected IEEE Xplore, ACM Digital Library, and Scopus as the primary databases for this systematic literature review (SLR), as these databases are widely used in the field

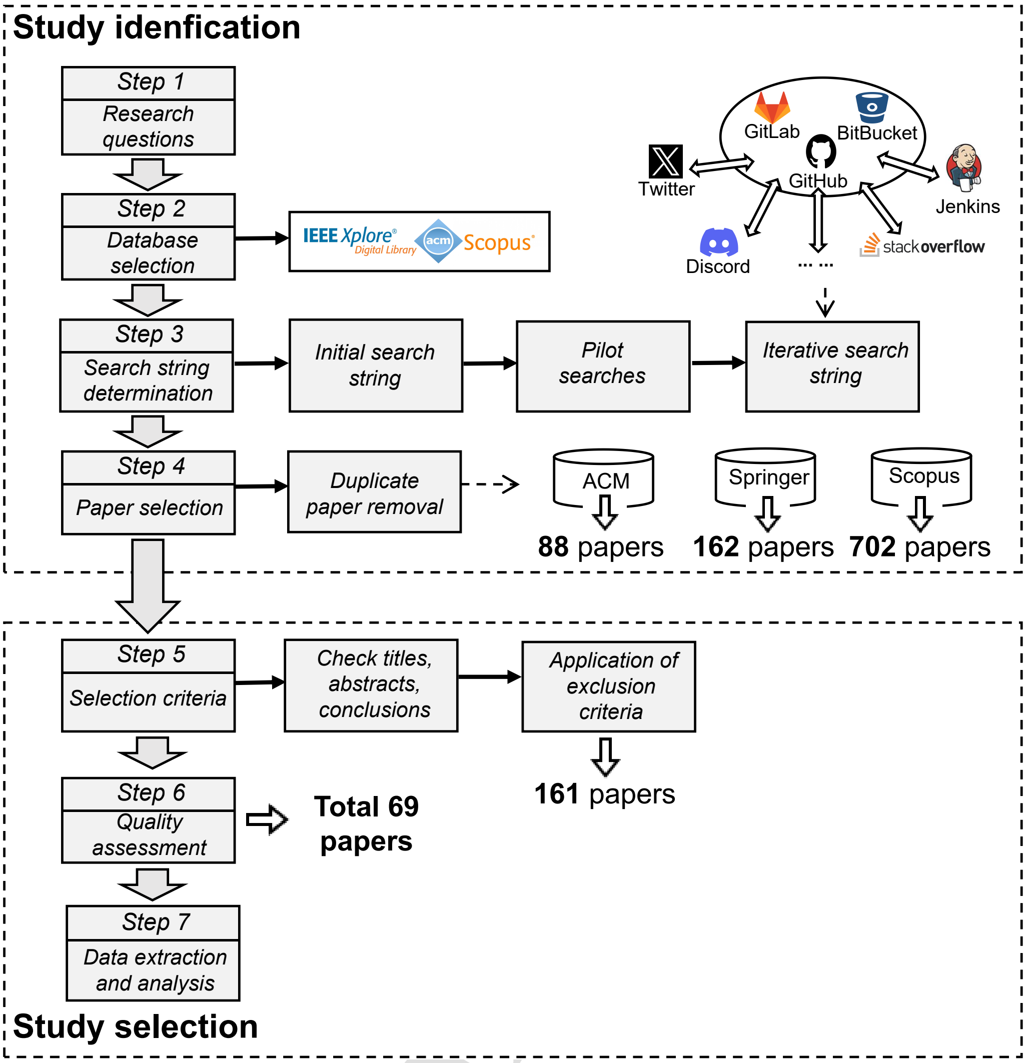


Figure 1: Research Framework

of computer science [37, 38, 39, 40]. During the literature retrieval process, we constructed search strings based on the search rules of each database, with a focus on the titles and abstracts of the papers. This is because we believe that relevant keywords are more likely to appear in these sections [41, 42], whereas full-text searches may generate a large amount of irrelevant noise data. For example, terms related to cross-platform research may be mentioned in contexts unrelated to our core research questions, such as "security of software packages across different operating systems"[43] and "cross-platform engines"[44].Additionally, we decided not to include Google Scholar and SpringerLink as search engines for this review. First, Google Scholar includes a significant number of technical reports and academic papers that have not undergone peer review, raising concerns about their quality and potentially compromising the reliability of the literature selection process [45]. According to statistics, 50-60% of articles in Google Scholar's database originate from online sources that lack peer review [46]. Second, although SpringerLink's advanced search functionality supports filtering by keywords, authors, and publications, it does not allow for limiting searches to titles and abstracts [47]. In our pilot search [48] using preliminary search strings (see 3.3), we found that SpringerLink yielded almost no articles directly relevant to the research topic. To ensure efficiency and accuracy in the literature selection process, we ultimately decided to exclude SpringerLink and Google Scholar.

3.3. Search string determination

To ensure that the search strings effectively retrieve literature relevant to the research objectives, we first conducted pilot searches.

Based on our experience, we first extracted cross-platform related keywords from several relevant papers, as shown in Table 1. The keywords contain two parts: one pertaining to open source software development, and the other related to cross-platform interactions. For open source software development, we find specific keywords commonly used in this domain, such as "open source software", "OSS"

Table 1
Search Keywords.

Type	Keywords	Summary
cross-platform interactions	"across platforms" [49], "across both platforms" [50], "across communities" [51], "across different communities"[52], "across the two platforms" [53], "across two platforms" [14]	"across * $\mathcal{X}s$"
	"cross-network" [53], "cross-system" [54]	"cross-$\mathcal{X}$"
	"multi-community" [52]	"multi-$\mathcal{X}$"
	"multiple networks" [54]	"multiple $\mathcal{X}s$"
open source software development	"open source software" [55, 56], "OSS" [55, 56], "open source projects" [57]	"open source" OR "OSS"

$\mathcal{X}$ stands for "platform" or synonyms, including community, network and system

and "open source projects". We summarized the general expression "open source" OR "OSS". For cross-platform interactions, we find four types of prefixes, namely "across", "cross", "multi" and "multiple". The keywords formed by these prefixes are shown in Table 1. For each type, we summarized the general expressions used for searching, i.e., "across * $\mathcal{X}s$", "cross-$\mathcal{X}$", "multi-$\mathcal{X}$" and "multiple $\mathcal{X}s$", where $\mathcal{X}$ stands for "platform" or synonyms.

In processing keywords such as "cross-network" and "multi-community", we removed the hyphens, as search engines treat them as special characters and ignore them. This approach aligns with the search guidelines of the major databases, including ACM[3], IEEE Xplore[4], and Elsevier[5]. Subsequently, we combined the keywords within each type using **OR** logic operators and linked different types using **AND** logic operators, resulting in the following search string:

> **Initial Search String.** (("across * communities" OR "across * platforms" OR "across * networks" OR "across * systems") OR ("cross community" OR "cross platform" OR "cross network" OR "cross system") OR ("multi community" OR "multi platform" OR "multi network" OR "multi system") OR ("multiple communities" OR "multiple platforms" OR "multiple networks" OR "multiple systems")) AND ("open source" OR "OSS")

During the pilot search process, we found that many papers on cross-platform research did not explicitly use keywords such as 'cross-platform' in their titles and abstracts. Instead, these papers often mentioned specific platform names, such as "GitHub" and "StackOverflow" [58]. This phenomenon made the process of identifying relevant papers through keyword searches more complex, as we were unable to directly identify which platforms were the primary focus of the research. To address the issues encountered during the retrieval process, we first utilized the initial search string identified earlier and conducted the search. Subsequently, we applied Named Entity Recognition (NER) to the titles and abstracts of the retrieved papers to extract the key platform names mentioned. Based on these platform names, we further optimized the search string to enhance the accuracy of selecting relevant studies. The pilot search results also indicated that very few relevant studies had been published prior to 2013 (<2%) and that their content was unrelated to our research topic. Therefore, we restricted the time frame of this review to the period from 2013 to 2024.

We completed the initial literature search on December 15, 2024, focusing primarily on the fields of computer science and software engineering. A total of 1,102 papers were retrieved, with 119 from the ACM Digital Library, 231 from IEEE Xplore, and 752 from Scopus. To identify key information related to platforms, we utilized a pre-trained model (en_core_web_sm) to perform Named Entity Recognition (NER) on the titles and abstracts of the collected papers. This process resulted in the extraction of 11,279 entities. After a manual review of the extracted results, we identified 19 commonly mentioned platform-related entities, including: GitHub, GitLab, BitBucket, Stack Overflow, Quora, HackerNews, Reddit, Jenkins, Gitter, Telegram, WhatsApp, Facebook, Instagram, YouTube, Twitter, Slack, Discord, DEV, and LinkedIn. These platform names appeared frequently in the reviewed literature and are closely aligned with the focus of our research. Given that the primary goal of this study is to explore open source oriented cross-platform research, we specifically focused on platforms associated with open-source project hosting. Among the 19 identified platforms, GitHub, GitLab, and BitBucket are the primary platforms currently used for hosting open-source projects. Based on this objective, we optimized our search strategy, as shown below. We used the **AND** operator to combine these open-source project hosting platforms (e.g., GitHub) with other frequently mentioned platforms (e.g., Stack Overflow) in the literature. At the same time, the **OR** operator was used to link different platforms, thereby expanding the search scope. The specific search strings used in each database are detailed in our open-source project [17].

[3] https://dl.acm.org/search/advanced
[4] https://ieeexplore.ieee.org/Xplorehelp/searching-ieee-xplore/search-tips
[5] https://service.elsevier.com/app/answers/detail/a_id/34325/

Iterative Search String. ("GitHub" AND ("Stack Overflow" OR "Quora" OR "HackerNews" OR "Reddit" OR "Jenkins" OR "Gitter" OR "Telegram" OR "WhatsApp" OR "Facebook" OR "Instagram" OR "YouTube" OR "Twitter" OR "Slack" OR "Discord" OR "DEV" OR "LinkedIn"))
OR
("GitLab" AND ("Stack Overflow" OR "Quora" OR "HackerNews" OR "Reddit" OR "Jenkins" OR "Gitter" OR "Telegram" OR "WhatsApp" OR "Facebook" OR "Instagram" OR "YouTube" OR "Twitter" OR "Slack" OR "Discord" OR "DEV" OR "LinkedIn"))
OR
("BitBucket" AND ("Stack Overflow" OR "Quora" OR "HackerNews" OR "Reddit" OR "Jenkins" OR "Gitter" OR "Telegram" OR "WhatsApp" OR "Facebook" OR "Instagram" OR "YouTube" OR "Twitter" OR "Slack" OR "Discord" OR "DEV" OR "LinkedIn"))

Table 2
Papers Screening Statistics.

Database	Preliminary Screening Count	Count After Deduplication
ACM	183	88
IEEE Xplore	273	162
Scopus	962	702
Total Preliminary Screened Papers	1,418	
Total Count After Deduplication	952	

3.4. Paper selection

We completed the second round of paper retrieval on December 16, 2024, using iterative search strings to focus on papers in the fields of computer science and software engineering published between 2013 and the current cut-off date. It is worth noting that the time range is based on the indexing dates of the databases, which means the results may include a small number of papers that have been archived in advance but not yet officially published. At this stage, we initially screened a total of 1,418 papers from the databases, as detailed in Table 2. Subsequently, we imported the BibTeX files exported from the ACM Digital Library, IEEE Xplore, and Scopus into the Parsifal platform[6], which provides built-in functionality for duplicate removal. After removing duplicates, 952 papers were retained, including 88 from the ACM Digital Library, 162 from IEEE Xplore, and 702 from Scopus.

3.5. Selection Criteria

Next, we applied a set of exclusion criteria to screen all studies collected through the search strategy. These criteria took into account various aspects of the papers, including their language, research completeness, and relevance. The specific exclusion criteria (EC) are detailed in Table 3.

In the process of screening papers, we used a portion of the sample data to assess the consistency between different authors in paper evaluation. With a 95% confidence interval and a 5% margin of error[59], we determined a sample size of 274 papers from a total of 952 papers. The sample size was calculated using the Sample Size Calculator [7]. These 274 papers were independently screened by the first two authors. Upon completion of the screening, we calculated the Kappa coefficient to evaluate the consistency between the two authors. The Kappa coefficient was 0.887, indicating the "almost perfect" level (0.81-1) of agreement[60]. The authors held a meeting to discuss the different results and reached a consensus on the final decision. The subsequent paper screening was uniformly handled by the first author. Ultimately, we identified 161 papers that met our study criteria.

[6] https://parsif.al/
[7] https://www.surveymonkey.com/mp/sample-size-calculator/

3.6. Quality Assessment

Next, we adopted the quality assessment criteria established by Yang et al. [61, 62] and made appropriate adjustments based on the specific research questions of this study, ensuring that each paper effectively addresses our research objectives. The detailed Quality Assessment Criteria (QAC) are presented in the table below. Each paper was evaluated based on five quality assessment questions, with answers categorized into three levels: "Yes" (1 point), indicating full compliance with the criteria; "Partial" (0.5 points), indicating partial compliance; and "No" (0 points), indicating non-compliance. Only studies with a total score of 4 points or higher were included in the analysis. The quality assessment was conducted independently by the first author, with final review and verification by the second author.

When developing the quality assessment criteria, we prioritized the ranking of publication venues to ensure the quality of the selected papers. Specifically, for conference papers, we referred to the CORE ranking [63], which is a system specifically designed to evaluate the impact of academic conferences [47]. It is important to note that the CORE ranking does not provide rankings for journals [64]. Therefore, for journal papers, we utilized the SJR (Scimago

Table 3
Exclusion Criteria.

Criteria
1. The paper was not written in English.
2. The paper is a summary of a conference/workshop or is a short paper (fewer than 4 pages), and is not a complete research study.
3. The paper is a duplicate of previously included research (due to name case differences, minor modifications, etc.).
4. The paper is unrelated to open-source projects.
5. The paper primarily focuses on a single platform (e.g., GitHub) or analyzes platforms in isolation without considering cross-platform.

Table 4
Quality Assessment Questions

No.	**Quality Assessment Question**	**Levels**
QA1	Is the study published in a high-reputation venue?	Yes / No / Partial
QA2	Does the study propose a clear motivation for cross-platform research?	Yes / No / Partial
QA3	Does the study clearly propose the connections way between platforms?	Yes / No / Partial
QA4	Does the study clearly design and describe experimental setups, including datasets and methods used ?	Yes / No / Partial
QA5	Does the study effectively discuss future research opportunities, challenges, and limitations?	Yes / No / Partial

Journal Rank) as the ranking criterion [65]. During the selection process, only journal papers categorized as SJR Q1 [66] and conference papers ranked as A or A* [47] were included to ensure the high quality of the research.

Among the 161 papers initially screened, 58 conference papers were rated as A or A*, and 35 journal papers were categorized as Q1. Subsequently, we conducted a detailed review of each paper to evaluate whether it explicitly articulated the motivation for cross-platform research, clearly described the relationships or connections between platforms, and provided a well-defined and detailed experimental design, including datasets and methodologies used. Additionally, we examined whether the studies mentioned their limitations, challenges, and potential directions for future research. Through this process, a total of 69 papers (42.9% of the total) achieved a score of 3.5 or higher. For the lower-quality studies that were excluded, analysis revealed that the primary issue was the failure to explicitly document the relationships or connections between platforms (QA3).

3.7. Data extraction and analysis

We strictly followed the systematic literature review methodology proposed by Kitchenham et al. [67] and conducted a structured data extraction based on the normative framework of this approach for addressing four research questions. Specifically, for the first research question (RQ1), we extracted the primary platforms and the connection mechanisms from the abstract (which, according to the guidelines, should clearly state the research topic) and the data collection section (which requires a detailed record of the methodological implementation). For the second research question (RQ2), we identified the core research themes by examining both the abstract and the research questions section (which, as per the guidelines, should clearly define the research objectives). In addressing the third research question (RQ3), we extracted dataset metadata and research design methods from the data collection and methodology sections (which are required to describe technical parameters). Finally, for the fourth research question (RQ4), we focused on the discussion, validity threats, and conclusion sections (which, as per the guidelines, should systematically summarize research limitations and future directions in the field) to analyze potential opportunities and challenges, establishing connections between the findings and broader implications for future research. At the same time, we validated this method by performing full-text coding on a randomly selected 20% of the papers, confirming that the implicit limitations in the "Results" section had been explicitly addressed in the "Discussion" section. Therefore, independent coding of the results is considered redundant. Additionally, we validated this approach by performing full-text coding on a randomly selected 20% of the papers, confirming that implicit limitations in the "results" section were explicitly mentioned in the "discussion" or "conclusion and future work" section . Therefore, independent coding of the results was deemed redundant.

We used the open card sorting method as described by Zimmermann et al.[68]. To start, we created descriptive cards based on the content extracted from each research question. Then, we classified the cards based on their similarity in content, creating new categories if no similar ones were found. This entire process was independently carried out by the first two authors and was assessed for consistency using the Kappa coefficient (0.823), which indicated a high level of agreement. Any disagreements found during the

Table 5
Data extraction for Research Questions.

RQ	Data Extraction
RQ1	Abstract, Data collection(connection ways)
RQ2	Abstract, Research questions
RQ3	Data collection(datasets names, descriptions, access links), methodology
RQ4	Discussion, threats to validity, conclusion and future work

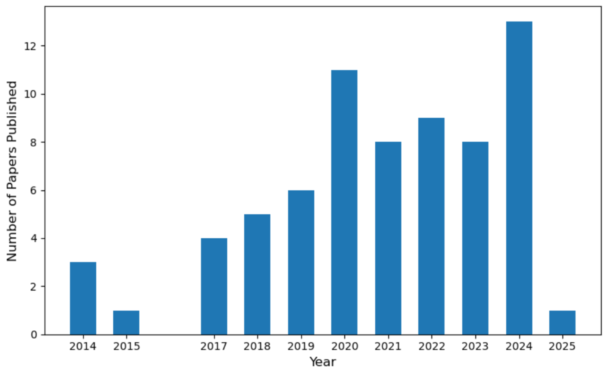


Figure 2: Number of Papers Published per Year

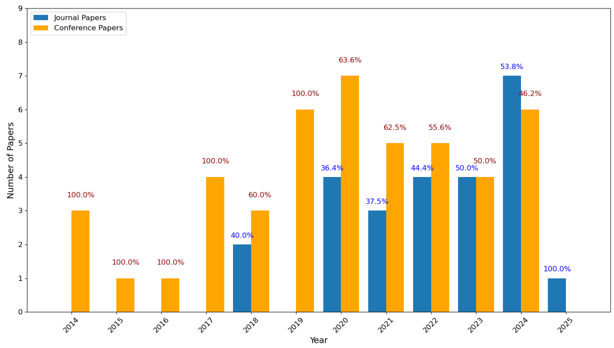


Figure 3: Distribution of Journal and Conference Papers by Year

assessment were resolved through discussion to reach a consensus [68, 69]. Finally, all authors reached an agreement on the categories during a collective meeting. These categories form the foundation for our subsequent comprehensive analysis of the research questions.

4. Results

In this section, we first conducted a general analysis of the number of papers published annually, their publication locations, and other related factors. Subsequently, we performed a detailed analysis based on the research questions (RQ1 to RQ4).

4.1. General Analysis

Figure 2 shows the number of papers published annually from 2014 to 2025. As shown in the figure, the period from 2014 to 2016 represents the initial stage of the research, with relatively few publications. Starting in 2017, the number of publications began to increase significantly, reaching its first peak in 2020. Between 2021 and 2023, the number of publications stabilized, indicating that the research in this field had entered a relatively mature stage. In 2024, the number of publications reached a historical high, while the data for 2025 remains incomplete as the year has only just begun. These findings indicate that cross-platform research has become increasingly popular over the past decade and has gradually matured into a significant research area.

This study collected a total of 69 papers published in various conferences and journals, including 25 journal papers, accounting for 36.2%. As shown in Figure 3, the distribution of journal and conference papers by year is presented. Based on the analysis of the figure, during the initial phase of the research field (2014-2017), conference papers overwhelmingly dominated, while journal papers were almost absent. This indicates that the field was still in its early developmental stage, with researchers favoring conferences as the primary platform for disseminating findings quickly. Since 2018, the number of journal papers has gradually increased, reaching 40%, signaling that the research field was gaining broader recognition within the academic community. By 2020, the proportion of journal papers reached 36.4%. During the period 2021-2024, the proportion of journal papers further stabilized at 37.5%-53.8%. This trend demonstrates a shift in research focus from rapid dissemination through conference presentations to formal publication in journals. It also reflects a deepening of research content and the progression of the field toward greater systematization and maturity.

Figures 4 and 5 present the main publication venues for the collected journal and conference papers. The research findings are primarily concentrated in high-impact journals such as ESE and TSE, as well as in top-tier conferences like ICSE and MSR. This indicates that the research efforts are highly focused on the field of software engineering. By analyzing the characteristics of articles published in these journals and conferences, it is evident that researchers have conducted in-depth explorations on topics such as methodological innovation, tool development, and the quality of open-source software development.

Table 6
Platform Types and Names

Platform Type	Platform Name
Social coding platforms [70]	Github [70]; GitLab [71]
Social Q&A platforms [72]	StackOverflow [72]; Stack Exchange [73]
Social media platforms [74]	DEV [75]; Gitter [76]; Twitter [77]; Reddit [77]; Hacker News, Forrst [16]; Reactiflux, Facebook, Slack, IRC (Internet Relay Chat), Mailing List [78]; Discord [79]; Google+ [74]; Security forums: Garage4Hackers, Offensive Platform, RaidForums, Multiplayer Game Hacking, Hack Forums [74]
issue tracking platforms [10]	Bugzilla [10]
Continuous integration platforms [80]	Jenkins [80]

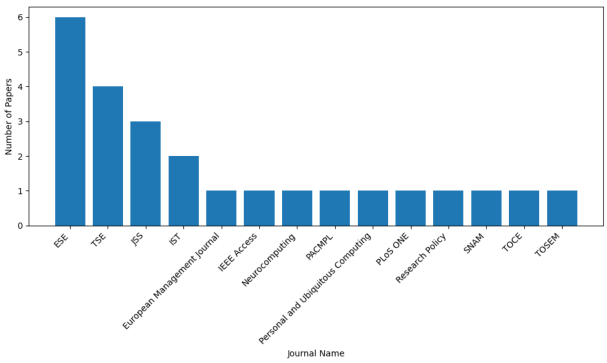


Figure 4: Number of Papers Published in Journals

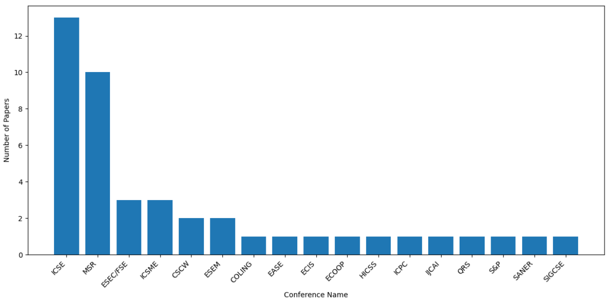


Figure 5: Number of Papers Published in Conferences

4.2. RQ1: How are different platforms connected in cross-platform studies?

Cross-platform research focuses on exploring the interactions and connections between different types of online platforms. By analyzing the ways in which platforms are connected, this type of research provides a fresh perspective for understanding cross-platform collaboration patterns and knowledge dissemination. This section discusses in detail the main types of platforms involved in cross-platform research, as well as the key traces used to establish connections between platforms.

4.2.1. Platform Types and Collected Traces

Table 6 provides an overview of the platform types and specific platform names involved in cross-platform studies. Platforms are categorized based on their core functionality and primary usage scenarios.

The table includes the following categories:

Social coding platforms: These platforms are primarily used for collaborative software development and version control. Their core functionalities include code sharing, collaborative development practices, code review, and project management [70, 71]. Represented by platforms such as GitHub and GitLab, they leverage distributed version control systems (e.g., Git) to support team members in efficiently sharing code and collaborating on development.

Social Q&A platforms: These platforms focus on knowledge sharing through a question-and-answer format. Their core goal is to connect users with questions to experts or members who can provide answers, thereby collaboratively solving complex technical challenges [81]. Represented by platforms such as StackOverflow and Stack Exchange, they not only offer efficient solutions to technical problems but also foster the dissemination of technical knowledge through a platform-driven model [82].

Social media platforms: The core functionality of social media platforms is to support team communication, collaboration, and information sharing, playing a crucial role in distributed open-source software projects. These platforms provide teams with convenient communication channels to facilitate task discussions, problem-solving, and project management [78]. Represented by platforms such as Twitter, Slack, Gitter, and Facebook, they significantly enhance the visibility of open-source projects through their broad audience base and efficient information sharing [83, 6], while also promoting the identification of technical issues and the growth of platforms [74].

Issue tracking platforms: The core functionality of issue tracking platforms is to record, assign, and track the lifecycle of issues, providing transparent process management to enhance project manageability and task traceability. A typical example is Bugzilla [10, 84].

Continuous integration platforms: The core functionality of continuous integration platforms is to support end-to-end management of code integration, testing, and deployment through automation tools, significantly enhancing the efficiency and quality of software development. Represented by platforms such as Jenkins, these tools can automatically pull code from GitHub once it is submitted, build it, and

Table 7
Type of connection between platforms.

Type	Related Study	Count(%)
Social coding platforms–social Q&A platforms	[86, 87, 88, 89, 71, 90, 91, 92, 13, 93, 94, 95, 96, 97, 98, 99, 100, 73, 70, 101, 102, 103, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 15, 115, 116, 115, 72, 117, 118, 119, 75, 120, 50, 121, 122, 14, 123, 124]	50(72.5%)
Social coding platforms–social media platforms	[16, 78, 77, 125, 126, 127, 128, 129, 79, 130, 83, 131, 132, 6, 133, 74]	16(23.2%)
Social coding platforms–social Q&A platforms–Issue tracking platforms	[10, 134]	2(2.9%)
Social coding platforms–Continuous integration platforms	[80]	1(1.4%)

execute automated testing and deployment tasks to ensure code quality and system stability [85, 80].

Table 7 summarizes the distribution of different types of connections in cross-platform research. Among them, the connection between Social coding platforms and social Q&A platforms constitutes the largest proportion (72.5%), followed by the connection between Social coding platforms and social media platforms, which accounts for 23.2%.

However, the realization of these connections relies on the data traces left by users' activities across different platforms. To further elucidate the types of information generated by different platforms and their roles in cross-platform connections, Table 8 provides a systematic summary of the data traces on major platforms and identifies which traces play a key role in the construction of cross-platform connections.

Research indicates that cross-platform connection methods primarily rely on the following types of information: user personal info, technical info, metadata of projects/posts/bug reports, and interaction info. Among these, some are explicit, while others are implicit. For instance, in user matching, explicit information is typically used to establish user connections through direct identifiers. Examples include email addresses [13, 101, 103, 106, 130], externally shared links between platforms (e.g., URLs)[132], or precise usernames[126, 128], which can directly identifies the same user across different platforms. Implicit information, on the other hand, involves inferring potential user associations by analyzing the similarity between usernames[74, 6]. Techniques such as string similarity calculations or applying edit distance algorithms can be used to deduce the corresponding user identities across platforms. Furthermore, cross-platform research heavily relies on key informational traces such as code snippets (e.g., projects, posts)[90, 94, 97, 103, 111, 115, 119], tags or keywords (e.g., project tags, issue tracking labels, tweets)[90, 91, 93, 98, 105, 77], and external links between platforms [78, 95, 79] to establish connections across platforms.

By leveraging explicit or implicit linkages, it is possible to trace the complete pathway of developers from knowledge sharing (e.g., Stack Overflow) to code implementation (e.g., GitHub). Cross-platform data associations, such as the co-occurrence of GitHub issues and Stack Overflow discussions, can help identify development bottlenecks. Furthermore, linking community interaction data from platforms like Reddit and Twitter with development activities on GitHub or Gitter enables the quantification of social influence on the evolution of open-source projects.

> **Finding 1.** Cross-platform research primarily focuses on two types of connections: social coding platforms–social Q&A platforms (72.5%) and social coding platforms–social media platforms (23.2%). The informational traces that establish these connections are primarily categorized into user personal info, technical info, metadata of projects/posts/bug reports, and interaction info.

4.3. RQ2: What are the major topics in cross-platform studies?

Using the systematic research topic analysis method proposed in [135], we conducted a classification of 69 cross-platform studies. Each category includes the number of related studies, the percentage they represent, and relevant references, as detailed in Table 9. The classification results indicate that the major topics in cross-platform research include problem classification and feature extraction (25 studies, 36.2%), platform collaboration (18 studies, 26.1%), code reuse and evolution (11 studies, 15.9%), user characterization (11 studies, 15.9%), and cross-platform data optimization (4 studies, 5.8%). We further visualized the annual distribution of publications across different research topics (see Figure 6). The results reveal that the topic of problem classification and feature extraction has shown a significant upward trend in recent years. This trend reflects that, with the rapid development of platforms and the continuous growth of data generated, an increasing number of researchers are exploring data from various platforms to gain a comprehensive understanding of complex problems. Furthermore, this highlights the critical importance of data integration and collaborative analysis in cross-platform research. The

Table 8
Key Information Types and Cross-Platform Traces

Platform Types	Information Categories	Key Traces for Cross-Platform Connections	Description
Social coding/Q&A/media platforms	*User Personal Info:* (username, email addresses, external platform links	MD5 hash value of users' email addresses [13, 101, 103, 106, 130]; username[126, 128, 74, 6]; external platform links [132]	Identifies the same user across platforms
Social coding platforms(e.g., GitHub)	*Technical Info:* (commit messages, bug id, code, push and pull requests)	Keywords of commit messages [134, 105, 10]; Bug ID [134, 10]; code [90, 87, 96, 10, 94, 97, 103, 110, 111, 115, 119, 14, 80, 100]; push/pull requests [88, 91, 128, 15, 123]	Records implementation details; can link to bug id; compares code snippets with Q&A platforms
	Project Metadata: (Project name, description, tags, creation/last commit date, language, organization/team, release, branch, wiki, readme)	Keywords of projects' descriptions and names [88, 77, 73, 95, 78, 88]; projects' language[86, 105, 123] projects' tags [90, 91, 93, 98, 105, 95]; readme / wiki and associated URLs [78, 95, 79] and Wiki files [78]	Describes basic attributes
	Interaction Info: (Issues, Stack Overflow links in issues, discussion, comments, forks, stars, contributors, followship)	Issues labels and keywords [88, 91, 128, 15, 123, 107, 71, 92, 113, 115]; discussion keywords [71, 107]; Stack Overflow links in issue [89]	Can reference Q&A discussions
Social Q&A platforms (e.g., StackOverflow)	*Technical Info:* (Code)	Code [88, 90, 94, 97, 103, 111, 115, 119, 122, 14]	Provides examples and technical details
	Post Metadata: (submission ID, title, body, answers, comments, tags, status, release date, change history)	Post tags [86, 87, 71, 90, 91, 92, 93, 98, 73, 105, 108, 114, 15, 115, 72, 120, 123, 88]; Topic [134]; post keywords [95, 96, 104, 107, 110, 110, 113, 117]; change history [10]	Thematic fields/links may point to code repos, issues, or project docs.
	Interaction Info: (Voting types: upvote/downvote, external platform links)	External platform links [70, 129, 116]	Link fields used to cite external resources
Social media platforms (e.g., Twitter)	*Interaction Info:* (Tweets/posts, retweets/shares, quoted tweets, replies/comments, Like/Favorite, link fields, chat logs, etc.)	Twitter: Keywords of tweets, retweets, quoted tweets, replies [125, 77]; links in posts/tweets [125, 75, 83, 6]; issue report links [134, 127]; chatroom project name [127, 121]	Shows user interactions and potential impact
	Post Metadata: (Posts, tweets)	Twitter: Hashtags of tweets, other platforms: post keywords [125, 77]	Marks topic content, can match projects or Q&A posts
Issue tracking platforms (e.g., Bugzilla)	*Bug reports Metadata:* (Bug ID, summary, description, product, component, status)	Bug ID [134, 10]	Describes issue context, linking Q&A or commit message

following sections will provide a detailed discussion of the core aspects of each topic.

4.3.1. Problem Classification and Feature Extraction

Problem classification and feature extraction is a key topic in cross-platform research, as it effectively addresses the limitations of single platforms in providing technical information and examples. For instance, developers often seek solutions to specific issues on Stack Overflow and upload optimized code to code hosting platforms such as GitHub. By integrating information from multiple platforms, researchers can obtain more comprehensive data support, enabling a deeper analysis of the specific problems developers face and potential solutions.

The majority of research is concentrated on software defect repair. Due to the lack of standardized defect benchmarks, evaluating the performance of related techniques becomes exceedingly complex. To address this challenge, researchers have proposed fine-grained defect classification

Table 9
Research Topics, Subtopics, and Related Studies

Research Topic	Subtopics ("→" represents "support")	Count (%)	Related Study
Problem Classification and Feature Extraction	Software bug fixes[105, 92, 123, 87, 134, 112, 113, 88, 114]	9(13%)	[134, 101, 115, 96, 102, 86, 92, 73, 105, 107, 112, 123, 120, 87, 15, 88, 98, 104, 71, 117, 10, 114, 91, 95, 113]
	API[96, 86, 120]	3(4.3%)	
	AutoML[104]	1(1.4%)	
	GitHub Actions[115]	1(1.4%)	
	GitHub Copilot[107]	1(1.4%)	
	WebAssembly[117]	1(1.4%)	
	Security patches[10]	1(1.4%)	
	Programming language security[91]	1(1.4%)	
	Machine learning management[71]	1(1.4%)	
	Value co-loss[102]	1(1.4%)	
	Architectural decisions[95]	1(1.4%)	
	Open-source project management[101]	1(1.4%)	
	Quantum software engineering[73]	1(1.4%)	
	Deep learning frameworks[15]	1(1.4%)	
	Emerging programming languages[98]	1(1.4%)	
Platform Collaboration	Continuous integration platforms → Social coding platforms[80]	1(1.4%)	[127, 130, 78, 83, 80, 75, 133, 16, 6, 100, 110, 118, 77, 111, 128, 103, 79, 125]
	Cross-Platform collaboration and mutual development[118, 77, 125]	3(4.3%)	
	Social coding platforms → Social Q&A platforms[103]	1(1.4%)	
	Social media platforms → Social coding platforms[127, 130, 78, 83, 75, 133, 16, 6, 128, 79]	10(14.5%)	
	Social Q&A platforms → Social coding platforms[100, 110, 111]	3(4.3%)	
Code Reuse and Evolution	Evolution of reused code snippets[119, 89, 14]	3(4.3%)	[119, 14, 122, 13, 93, 90, 94, 116, 89, 97, 115]
	Reuse behavior for code snippets[122, 90, 116]	3(4.3%)	
	Origin of reused code snippets[93]	1(1.4%)	
	Adaptation of reused code snippets[94, 115]	2(2.9%)	
	Attribution of reused code snippets[13, 97]	2(2.9%)	
User Characterization	User structure analysis[132]	1(1.4%)	[131, 132, 106, 103, 70, 50, 74, 72, 126, 124, 109]
	User identity recognition[70, 74]	2(2.9%)	
	User behavior analysis[131, 109]	2(2.9%)	
	User profiling assessment[106, 103, 50, 72, 126, 124]	6(8.7%)	
Cross-platform Data Optimization	Topic modeling optimization[108]	1(1.4%)	[108, 121, 99, 129]
	Semantic matching for Q&A[129]	1(1.4%)	
	Types of fine-grained information traces[121]	1(1.4%)	
	Title completion and optimization[99]	1(1.4%)	

frameworks and conducted in-depth analyses of repair patterns, providing strong theoretical support for defect repair research. With the widespread application of artificial intelligence (AI) technologies in software systems, understanding the defect characteristics of AI-based systems has become crucial for ensuring software quality and maintainability. For instance, numerous studies focus on deployment challenges associated with deep learning frameworks such as TensorFlow, PyTorch, and Keras[92, 105, 123, 113, 87, 112]. Additionally, researchers have explored critical defect issues in other areas, such as actor-based concurrent development[88] and Android runtime permission management[114]. In terms of repair methods, relevant

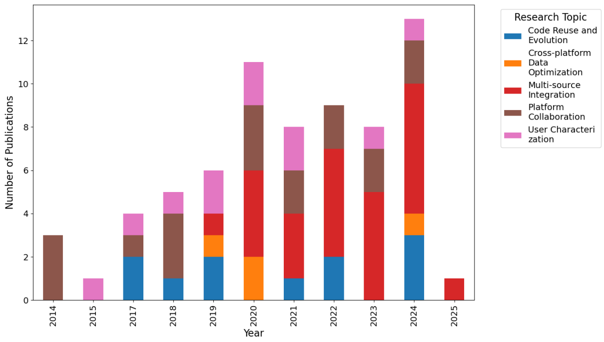


Figure 6: Publications by Year and Research Topic

studies have leveraged historical defect data and utilized automated techniques to generate repair patches, thereby achieving automated defect repair[134].

Use of API. Subsequently, the use of APIs represents the second most studied area. APIs are crucial for enabling developers to access functionalities and third-party libraries and are widely adopted in modern software development[120]. However, the complexity and diversity of software systems[86], as well as ambiguities in API method names[96], pose challenges in API usage. Researchers have leveraged cross-platform data to address these issues, such as helping developers efficiently locate relevant code examples[96], analyzing misuse patterns[86], and exploring challenges associated with specific frameworks like Reactive Programming[120]. These studies highlight the importance of improving API usability and support systems.

Additionally, researchers have focused on key issues across a wide range of fields, from the application of emerging technologies to concerns related to security and reliability. In the application of emerging technologies and tools, such as GitHub Copilot, GitHub Actions, WebAssembly, quantum computing software (QSE), and new programming languages like Swift, Go, and Rust, studies have revealed numerous technical challenges that developers face when using these technologies[115, 107, 117]. In the fields of machine learning and artificial intelligence, research has primarily focused on areas such as automated machine learning[104], deep learning frameworks[15], machine learning asset management[71], and key issues in the development of AI systems[95]. In terms of security and reliability, Croft et al.[91] analyzed the potential security risks associated with different programming languages. In the area of platform building, to understand the key factors that attract high-skilled developers to continuously contribute, relevant studies have examined the management practices of open-source projects and discussed value decomposition within online collaborative networks (OCN) to identify potential influencing factors[101, 102].

In summary, these studies highlight the extensive application of problem classification and feature extraction and provide significant references for understanding the relevant issues.

4.3.2. Platform Collaboration

Platform collaboration emphasizing the interdependence and cooperative development between different platforms. This theme explores how one platform supports the functionality and growth of another, as well as the reciprocal benefits derived from such collaboration.

Platform-Supported Development. Platform-supported development refers to practices that enhance platform efficiency and quality through collaborative interactions. A notable example of this is the support that social media platforms provide to social coding platforms. For instance, the rapid adoption of instant messaging tools such as Gitter and Slack is reflected not only in the significant increase in the number of README files linking to these tools[79], but also in their crucial role in facilitating distributed development collaboration[128]. On one hand, researchers have widely explored the impact of social media on software development, covering areas such as issue management[127], the GitHub Sponsors funding model[83], communication among developers[16], and attracting contributors[6, 128]. On the other hand, studies also focus on how developers utilize social media during collaborative development processes[78, 75, 133, 79]. Furthermore, in recent years, research has begun to explore how to recommend more relevant social media content to developers[130].

Additionally, social Q&A platforms play an important role in supporting the development of social coding platforms. Extensive research has utilized the knowledge from Stack Overflow to supplement and optimize searches on GitHub, significantly improving the quality of recommendations and searches[100, 110, 111]. For example, by extending the content from Stack Overflow to generate high-quality API sequences[100, 110]. On the other hand, studies have found that social coding platforms also provide support to social Q&A platforms. For instance, API usage patterns mined from GitHub projects can be used to detect improper API usage in Stack Overflow posts[103]. Moreover, continuous integration platforms like Jenkins provide strong support to social coding platforms such as GitHub and GitLab by optimizing code testing and deployment processes, not only enhancing the efficiency of code submissions and testing but also enabling real-time feedback and automated grading[80].

Cross-Platform Collaboration and Mutual Development. Another prominent research topic is cross-platform collaboration to achieve mutual development. For instance, in the domain of information dissemination, different platforms complement each other functionally to build an efficient system for diffusion and collaboration. GitHub serves as the starting point of information, providing initial resources and technical support. Twitter extends the reach of the information, covering a broader audience. Reddit offers a platform for in-depth discussions, while Slack enhances the precision of information transfer through efficient team collaboration. This cross-platform collaboration mechanism leverages the complementary functionalities of different platforms to enable the rapid dissemination and sharing of information, significantly improving developers' efficiency and quality in accessing relevant resources[77, 125].

4.3.3. Code Reuse and Evolution.

Reuse behavior of code snippets. Code reuse and evolution focusing on how developers utilize code snippets across platforms to address programming challenges and improve efficiency. Studies have shown that developers tend to reuse larger and non-trivial code blocks, with a strong preference for high-quality Stack Overflow (SO) posts. These posts are often highly rated or frequently bookmarked, and developers commonly refer to multiple related posts. Additionally, files that undergo frequent modifications exhibit significantly higher rates of code reuse compared to other files[122, 90]. Among various languages, JavaScript is reused most frequently, with references to its code in GitHub projects dynamically evolving over time[116].

Evolution of reused code snippets. However, another prominent research area focuses on the evolution of reused code snippets. While code reuse offers convenience, it also raises concerns regarding synchronization, updates, and security. Research by Manes et al.[14] found that code snippets on Stack Overflow (SO) and GitHub typically evolve independently, resulting in many reused SO code snippets in GitHub projects not being updated in a timely manner[119]. Additionally, many reused code snippets contain security vulnerabilities that propagate across multiple projects without being addressed, posing significant risks.

Adaptation of reused code snippets. Other tasks also include research on the adaptation of code snippets, a process that involves multiple complex factors, such as contextual environment, semantic consistency, and functional optimization. Zhang et al.[94] systematically revealed the associations between Stack Overflow (SO) posts and corresponding code snippets in GitHub projects by combining code clone detection, timestamp analysis, and explicit URL references. Their study clarified the dynamic adaptation features of cross-platform code reuse. The research found that when developers modify the same code snippet, they typically follow specific adaptation patterns. In subsequent research[115], four typical context-based adaptation patterns were further refined, including fortification, code wiring, attribute-ization, and parameterization. These patterns reflect the diverse practices of developers in code adaptation. The study also pointed out that most adaptations are corrective in nature, primarily focused at the variable level, and tend to occur within the last 10 lines of a code snippet. These findings not only reveal the adaptation patterns in code reuse but also provide theoretical guidance and practical support for the development of automated adaptation technologies.

Attribution and Origin of reused code snippets. Moreover, the attribution of reused code snippets is another common research task, as the use of reused code can lead to maintenance and legal issues. Studies have shown that insufficient attribution and a lack of understanding of licensing agreements are prevalent among developers, highlighting the significant legal challenges associated with code reuse[13, 97]. Furthermore, Stack Overflow itself faces issues related to the origin of code, with approximately 70% of JavaScript snippets being sourced from GitHub or other external repositories[93], further emphasizing the complexity and multi-layered impacts of code reuse.

4.3.4. User Characteristics.

User identity recognition. In cross-platform research, user identity recognition serves as the foundation for conducting user characteristics assessment. Related studies focus on achieving accurate cross-platform user matching by integrating multidimensional features [70, 74].

User expertise assessment. Based on user identity recognition, user characteristic evaluation typically focuses on developers' expertise. However, traditional evaluation methods, such as relying on resumes or social recommendations, often fail to fully capture a developer's actual abilities. Research shows that developers' activities on Q&A platforms significantly improve the accuracy of bug assignment [106], and also play an important role in supporting "cold start" users [103]. Additionally, developers' contributions on GitHub are largely driven by personal motivations, while their activities on Stack Overflow are primarily related to career development needs [50]. This difference in driving forces reveals distinct behavior patterns and needs of developers across different platforms, further highlighting the potential value of cross-platform data in skill evaluation. Moreover, studies have found a significant positive correlation between team members' chat contributions and code contributions, which validates the key role of soft skills (such as communication and teamwork abilities) in defining "expertise" [126, 50]. Therefore, the evaluation of expertise should encompass not only technical skills (such as programming languages and tool usage) but also soft skills, as both are integral to a developer's overall competence [50]. Given the vast amounts of information contained in online collaborative platforms, related research has integrated contribution data from social coding platforms and social Q&A platform to construct aggregated views of candidate contributions. Furthermore, tools have been developed to support detailed analysis [124, 72], providing more comprehensive and reliable bases for skill evaluation and personalized recommendations.

User behavior and structure analysis. Analyzing users' cross-platform behavior is another important task. Research indicates that developers' behaviors are driven by multiple factors, influenced both by individual role characteristics and external environmental factors [131, 109]. For example, developers in different roles (such as repository owners, project contributors, and followers) exhibit significant differences in their behavior on Twitter [131]. Additionally, major events can have a profound impact on developers' public contribution behaviors, such as changes in activity levels or adjustments in contribution patterns [109]. Furthermore, studies on user structures have revealed the evolving patterns of leadership structures within online platforms [132].

4.3.5. Cross-platform Data Optimization.

Moreover, the optimization of cross-platform related data has increasingly attracted attention. In the field of software engineering, textual data, such as source code comments, issue descriptions, and Stack Overflow Q&A content, contains rich semantic information, making it a valuable resource for enhancing the quality of cross-platform research. To this end, researchers have employed topic modeling techniques to optimize text analysis[108], refined knowledge-sharing classification systems[121], and uncovered the positive effects of both explicit and implicit knowledge on open-source contributions. Additionally, in addressing title quality and cross-platform issue matching, studies have proposed efficient automatic completion and semantic association methods[99, 129]. These efforts collectively contribute to the advancement of multi-source data mining and the improvement of software collaboration efficiency.

> **Finding 2.** Cross-platform research focuses on five key topics: problem classification and feature extraction(36.2%), platform collaboration(26.1%), code reuse and evolution(15.9%), user characterization(15.9%), and cross-platform data optimization(5.8%). Problem classification and feature extraction improves data coverage and aids in issue identification. Platform collaboration emphasizes interoperability benefits, while code reuse and evolution tackle synchronization and security challenges. User characterization highlights developer behavior patterns and profiling assessment.

4.4. RQ3: How to design experiments for cross-platform studies?

4.4.1. How is the data obtained?

When conducting cross-platform research, experimental design must consider multiple key factors, particularly the selection of datasets and the design of research methods. Choosing appropriate publicly available datasets is fundamental to cross-platform studies. Currently, there are 40 publicly available cross-platform datasets for researchers to use, as shown in Table 10. The table summarizes key information about these datasets, including the research domain, dataset name, scale, time range of data collection, access links, related papers citing these datasets, and the specific research tasks for which they can be used.

When selecting cross-platform datasets, problem classification and feature extraction is a key research direction. Such datasets integrate information from multiple development platforms (e.g., GitHub, Stack Overflow, Gitter) and cover a range of developer activities, including collaboration, technical discussions, API usage, defect fixing, as well as the use of machine learning frameworks (e.g., TensorFlow, PyTorch) and development tools (e.g., GitHub Actions). The data typically exists in the form of issues, posts, commits, and other content, which are the focus of research. Currently, analyses of these datasets primarily focus on software defect fixing and machine learning frameworks, with relevant datasets being relatively abundant.

In the field of platform collaboration research, the focus is on the impact of social media on developer behavior and project development, as well as the flow of information between different platforms. Relevant datasets emphasize social media posts or tweet content, and how such content influences activities on development platforms like GitHub (e.g., issues, pull requests). Within this topic, research not only emphasizes the contextual matching of information across different platforms [130], but also focuses on recommendation evaluation based on global information. For instance, researchers can establish connections between platforms by analyzing URL information or keyword matching in README files [79, 77], without relying heavily on specific contextual details.

In the field of code reuse research, cross-platform datasets primarily focus on code snippets obtained from different platforms, and use code clone analysis to explore the reuse and evolution of code across platforms. A typical dataset in this area is the SOTorrent dataset [11], which is specifically designed to analyze cross-platform code reuse and evolution between Stack Overflow and GitHub. This dataset provides version histories of code blocks, revealing how technical discussions on Stack Overflow influence code implementations on GitHub, and exploring the evolution, reuse, and adaptation processes of code snippets.

Through an in-depth analysis of user characterization evaluation and cross-platform data optimization, we can reveal how interactions between different platforms influence developer behavior and the optimization of platform content. Data collection primarily focuses on identifying the same user across different platforms [131], and mining data related to cross-platform data optimization, particularly common types of information such as titles and their contextual information, to optimize the flow of information between platforms [99, 129].

Furthermore, the research field of general-purpose datasets also has broad applications. Datasets such as StackOverflow Data Dump, GHTorrent [136, 31], and gharchive provide rich public data that supports a variety of research tasks. For instance, the StackOverflow Data Dump offers quarterly updates, including questions, answers, tags, votes, and badges, making it a core data source for question-answering analysis. GHTorrent provides over 900GB of raw data and 10GB of metadata, covering multiple dimensions of data on GitHub, such as issues, commits, and pull requests (PRs). Gharchive collects event records from GitHub, encompassing 236 million event records from 2017 to 2020, with updates occurring every hour. It is particularly worth noting that GHTorrent is more suited for providing historical records of individual projects and developer activity logs.

However, during the process of collecting and organizing data, we identified several outdated datasets, whose access links are no longer valid and cannot be used for subsequent research. For example, since June 2019, shell access to the

GHTorrent service has been discontinued [8]. In addition, some older dataset links have become inactive, such as those referenced in [110] and [103]. Furthermore, some datasets have yet to be made publicly available, such as those cited in [124], [75], [126], [108], [87], and [121]. Additionally, some researchers have conducted problem analysis and studies through interviews, as seen in [102], [133], and [50]. Regarding data collection methods, datasets in cross-platform research are typically gathered through platform APIs or web scraping techniques.

4.4.2. What kind of research methods are used in the related studies?

Based on a systematic review of existing research, the research methods in cross-platform software engineering can be categorized into four main types: data-driven methods (including data mining, code clone detection, semantic/text analysis, time series analysis, etc.), qualitative studies (such as interviews and surveys & questionnaires), modeling & ml approaches(including machine learning and large language models), and tool development and implementation (such as tool prototyping, deployment & user evaluation).

Data-Driven Methods. Data analytics methods focus on how to systematically collect, preprocess, and analyze large-scale data from various platforms such as GitHub, Stack Overflow (SO), and Twitter.

Data Mining. Researchers employ methods such as API scraping [134, 120], tag-based retrieval [115, 107, 123, 88, 117, 114, 95], heuristic approaches [73, 91], and multi-method hybrid search [112, 87, 15, 98, 104, 71, 10]. For example, Li et al. [73] extracted QSE-related questions and reports from Stack Exchange and GitHub through heuristic search, and applied the LDA model to identify the challenges faced by developers. Zhang et al. [115] collected 6,590 Stack Overflow questions and 315 GitHub issues via tag-based retrieval and manual annotation, using metrics such as *avgView* and *ansRate* to measure the popularity and difficulty of the questions. Data mining has played a critical role across various topics.

Code Clone Detection. To investigate cross-platform code reuse, plagiarism, and evolution, researchers have introduced clone detection tools such as NiCad, PMD [119], and SourcererCC [122]. Yang et al.[122] applied multi-level approaches, including exact matching, token hashing, and partial clone detection, to analyze code snippets from GitHub (Python projects) and Stack Overflow. Baltes et al. [13], using large-scale datasets, employed regular expressions and code clone detectors to explore the prevalence of common Java code snippets on GitHub and validated the phenomenon of uncredited code usage through developer surveys. Code clone detection methods are primarily used in cross-platform research to study the impact of cross-platform code copying and pasting behaviors, providing essential insights into code reuse patterns and their potential risks.

Semantic/Textual Analysis. As a key technique for extracting deep semantic information from unstructured development data (e.g., issue reports, technical Q&A posts, README documents), semantic analysis leverages natural language processing (NLP) to decouple language from code context, enabling the effective identification of implicit technical requirements and behavioral intentions in text. For instance, Rahman et al. [110] combined Stack Overflow Q&A texts with GitHub code snippets and used KAC (Keyword-API Co-occurrence) and KKC (Keyword-Context Co-occurrence) algorithms to gather and rank candidate API classes. Semantic analysis has significant application value in scenarios such as API recommendation and Q&A matching.

Time Series Analysis. Time series analysis treats development activities (e.g., code commits, question postings, post edits, etc.) as time series data, investigating their dynamic trends and temporal associations in cross-platform information dissemination. For example, Manes et al. [14] treated SO edits and GitHub revisions as parallel time flows and studied their relationship by analyzing the "impact latency." Time series analysis is particularly suited for studying issues involving temporal factors, such as the speed of information diffusion and the efficiency of question answering.

Qualitative Studies. Qualitative research methods complement quantitative analysis by uncovering developer behavior patterns, collaboration processes, and decision-making factors that cannot be revealed through numerical data alone.

Interviews. For example, Bidar et al. [102] conducted 36 semi-structured interviews with members from Stack Overflow and GitHub, and, based on Service-Dominant Logic (S-D Logic) and Resource Integration Theory, developed an analytical framework for cross-platform value co-destruction. Zhang et al. [115] interviewed 21 developers to analyze the contextual adaptation mechanisms during the code migration process.

Surveys & Questionnaires. Online surveys are widely used to collect both quantitative and qualitative feedback from large-scale user populations. For example, Vadlamani et al. [50] conducted a survey with 73 developers who were active on both GitHub and Stack Overflow, focusing on the cognitive differences regarding the "expert" role and the drivers and potential barriers to cross-platform contribution behavior. Flores et al. [125] employed a multi-source sampling strategy, recruiting a heterogeneous user group from platforms like Twitter, Facebook, and Slack, and collected data through structured online surveys. The study systematically coded the results qualitatively, analyzing information dissemination patterns and also obtaining qualitative feedback on users' multi-platform behaviors.

Modeling & ML Approaches. With the rapid development of machine learning technologies, researchers have increasingly introduced traditional machine learning algorithms and large language models into the field of software

[8] https://github.com/ghtorrent/ghtorrent.org

Table 10
Publicly available datasets in cross-platform studies

Topic	Dataset Name/Related Research	Dataset Scale	Cited References	Application ("→" represents "support")
Problem Classification and Feature Extraction	[134]	91,704 bug reports, 5,024 GH commits, 909,812 SO posts. (link)	—	Software bug fixes
	[112]	415 SO bugs, 555 GH bugs, 320 SO bug fixes, 347 GH bug fixes for 5 DL libraries. (link)	—	
	[123]	1,981 bug-related commits, 1,392 issues and PRs, 2,653 SO posts. (link)	—	
	[88]	186 Akka Actor Bug. (link)	—	
	[92]	65 SO posts and 132 GH issues (TF Lite); 52 SO posts and 38 GH issues (Core ML); 304 faults (287 posts). (link)	—	ML frameworks
	[15]	26,887 posts, 19,400 issues, 16,930 pull requests (TensorFlow, PyTorch, Theano). (link)	—	
	[113]	1,075 posts: 511 about Horovod, 329 about TensorFlow, 157 about PyTorch, 83 about Keras. (link)	—	
	[115]	6,590 SO questions, 2,471 SO accepted answers, 315 GH Actions issues from 89 repos, 217 closed (2018-2022). (link)	—	GitHub Actions
	[96]	127 threads covering API mentions of 181 API methods. (link)	—	API
	[86]	164,328 SO posts, 869,544 repos using target libraries. (link)	—	
	[107]	4,057 issues, 925 answered discussions, 679 posts. (link)	—	GitHub Copilot
	[104]	769 SO questions, 1,437 relevant GH issues. (link)	—	AutoML
	[71]	6,755 SO posts, 4,962 forum posts, 3,332 GH issues, 3 GitLab issues, 43 GH discussions. (link, link, link)	—	ML asset management
	[117]	385 GH issues, 354 SO posts. (link)	—	WebAssembly
	[10]	12,432 CVE patches from repos, 12,458 insecure posts from Q&A sites. (link)	—	CVE patches
	[114]	135 posts, 199 issues. (link)	—	Sarp
	[91]	280,000 security-related dev discussions from SO and GH (15 languages). (link)	—	Programming language security
	[95]	174 SO posts, 128 GH issues. (link)	—	Architecture decisions
Platform Collaboration	[127]	3,133,106 messages across 24 chat rooms, 14,096 issue references, 457 manually analyzed issue reports. (link)	—	Gitter discussions → GitHub issue
	[130]	150 Space channels, 300 Slack channels, 2000 employees, 300 teams, 2000 Space repos, 600 GH repos. (link)	—	Slack channel recommendation
	[83]	10,531 tweets with GH Sponsors links (May 2019 - Apr 2022). (link, link)	—	Twitter tweets → GH sponsors
	[6]	15,975 tweets, 28,569 retweets, 2,370 repos (Nov 2018 - Apr 2019). (link)	[131]	Twitter tweets → GH repos
	[100]	Collected 196,276 pairs of annotation and API sequences. (link)	[137]	API
	[77]	12,928 GitHub CVEs, 11,448 Twitter CVEs, 5,297 Reddit CVEs (Jan 2015 - Sep 2017). (link)	—	CVE
	[128]	4,506 contributors who collaborate on GitHub and chat on Gitter. (link)	—	Common user on GitHub and Gitter
	[79]	12,081 projects, 2,349 links with 282 types of readme links. (link)	—	Analyze readme links
Code Reuse and Evolution	[119]	31,287,646 code snippets, 11,479 repos(4,098,397 files)(Dec 31, 2020). (link)	—	Code snippet evolution
	[13]	29,370 SO Java snippets, 1,720,587 GH Java files. (link)	—	Code snippet similarities
	[93]	276,547 SO code snippets, 292 GH repos, 12,579 clone pairs. (link)	—	
	[90]	793 repos (342,148 modified code snippets), 1,355,617 posts. (link)	—	
	[94]	312K SO posts, 51K non-forked GH repos. (link)	—	
	[89]	72,483 C++ code snippets. (link)	—	
	SOTorrent data set[11]	38.4M SO posts, 11M extracted URLs, and 5.81M linked posts in 430K GH repos. (link)	[119, 14, 116, 115]	Version history of SO text or code blocks
	BigQuery	2.8 million GH repos, 145 million commits. (link)	[97]	Powerful code search capabilities
User Characterization	[131]	70,427 GH-TW user pairs, 129,843 tweets linked to GH (Jan 1, 2018 - Jul 1, 2019). (link)	—	Users' tweet and development activity
Cross-platform Data Optimization	[99]	189,655 SO posts , 333,563 GH issues (Jul 2008 - Dec 2023). (link)	—	Title completion
	[129]	16,761 SO posts, 12816 GH repos. (link)	—	Semantic matching of GH repos and SO posts
General-purpose	StackOverflow Data Dump	Archived SO content, including posts, polls, tags, badges, etc (Updated every quarter). (link)	[101, 122, 109, 73, 106, 120, 97, 118, 70, 111, 98, 72]	Question and answer analysis
	GHTorrent[136, 31]	Over 900GB of raw data and 10GB of metadata (issues, commits, PRs, etc.) (link)	[14, 122, 6, 116, 97]	Querying GH public event data
	gharchive	2.36B event records (push, issue, pull request, etc., 14 types, 2017-2020)(Updated every hour). (link)	[109, 16]	

engineering to address complex tasks such as platform recommendation, defect fixing, and cross-platform data optimization.

Machine Learning / Model Building. Traditional machine learning methods, such as Random Forests and XGBoost, have shown significant advantages in classification and recommendation tasks. For example, Treude et al. [108] proposed a machine learning-based framework for optimizing topic modeling parameters. They first collected multilingual text data from GitHub and Stack Overflow, extracting 24 statistical features, including character count, word count, and entropy. They then applied the irace algorithm for automated parameter tuning, using perplexity as an evaluation metric for the performance of the LDA model. Finally, by training a cost-sensitive Random Forest model, they achieved parameter configuration prediction based on corpus features, providing an efficient and automated solution for cross-platform text mining tasks.

Large Language Models. Pretrained language models, such as GPT-4 and CodeBERT, are widely applied to complex tasks such as code generation, defect fixing, and knowledge inference. For example, Bo et al. [134] proposed a knowledge-enhanced large language model approach for software bug fixing. They first collected bug reports and corresponding fix information from GitHub, Stack Overflow, and Bugzilla, using Named Entity Recognition to extract bug entities and construct a Bug Knowledge Graph (BKG). Then, they retrieved relevant historical information based on syntactic and semantic similarity. Finally, they input the bug description, code, and retrieved historical fix information into GPT-4 to generate interpretable patches.

Tool Development and Implementation. To validate the research methods, many studies further develop prototype systems and evaluate their performance in real-world environments.

Tool Prototyping. Researchers develop tool prototypes to support software engineering practices. For example, Luong et al. [96] developed the ARSearch system, which helps developers understand API usage by matching GitHub example code with Stack Overflow threads. Heckman et al. [80] built the Canary system, integrating professional tools such as GitHub, Jenkins, and Eclipse to support code commits, collaborative development, continuous integration, and automated grading, providing a comprehensive framework for supporting software engineering practices.

Deployment & User Evaluation. After tool development, researchers assess tool performance and user experience through both quantitative and qualitative methods. For example, Mahajan et al. [111] developed the Maestro tool and conducted internal evaluations using 78 instances from the top 500 Java projects on GitHub. They compared the performance of Maestro and its baseline variants with competing tools, and further validated the tool's effectiveness through a user experience study involving 10 Java developers.

Evolution of Research Methods. The current research methods exhibit several prominent trends: the dominance of data-driven methods, an increasing integration of multiple methodologies, and the rising exploration of intelligent methods. Data-driven methods (accounting for 71.1%) have become the foundational approach, widely applied in areas such as code reuse analysis [122] and problem classification and feature extraction [115]. At the same time, researchers have begun to integrate cross-technology stack methodologies. For instance, Wang et al. [114] combined data mining, surveys and questionnaires, and user evaluation in their study analyzing the challenges posed by the runtime permission model in Android 6.0 for developers. Although the application of large language models (LLMs) is currently limited (accounting for 10.5%), their potential in tasks such as defect repair [134] and title completion [99] has already begun to surface, marking the initial exploration and application of intelligent methods in cross-platform research.

Analysis of Research Method Strengths and Weaknesses. As shown in Table 11, various research methods exhibit distinct advantages and limitations in cross-platform research. Data-driven methods (such as data mining) demonstrate high efficiency and scalability in cross-platform studies, but their effectiveness relies on high-quality data and semantic enhancement techniques. Modeling and machine learning approaches (such as large language models) offer automated support for complex tasks, but they face challenges related to computational resources and domain adaptation. Tool development and implementation methods enhance practicality through validation in real-world scenarios, but issues related to cross-platform compatibility and maintenance costs still require further optimization. Qualitative studies, while providing in-depth analysis of behavioral logic, are limited by sample size and generalizability.

> **Finding 3.** In the domain of cross-platform research, we systematically compiled and organized 40 publicly available datasets. In terms of research methods, existing studies primarily adopt four main approaches: data-driven methods (71.1%), qualitative research (9.2%), modeling & ml approaches (10.5%), and tool development and implementation (9.2%).

4.5. RQ4: What are the key challenges and research opportunities identified in the existing literature?

Cross-platform research facilitates resource sharing and collaboration between different platforms, thereby enhancing development efficiency and the quality of information dissemination. Despite the promising potential of this field, cross-platform research faces numerous challenges, while also offering a wealth of research opportunities. This section will summarize the main challenges and research opportunities related to cross-platform research as identified in the existing literature, with the aim of providing guidance and insights for future research directions.

Table 11
Advantages and Limitations of Research Methods

Method Category	Core Advantages	Main Limitations
Data-Driven Methods [127, 130, 131, 119, 101, 14, 115, 78, 122, 132, 13, 86, 109, 83, 92, 73, 93, 75, 105, 126, 16, 6, 107, 90, 94, 106, 112, 123, 110, 116, 89, 120, 108, 97, 118, 77, 87, 15, 121, 88, 98, 104, 115, 128, 74, 71, 72, 117, 10, 114, 79, 125, 95, 113](71.1%)	• Efficient processing of large-scale heterogeneous data (e.g., Jallow et al. [119] detected 1.5 million code snippets)	• Strong dependency on data quality (e.g., Raglianti et al. [79] reported a significant amount of noise that is difficult to filter) • Limited semantic understanding (requires supplementary semantic analysis)
Modeling & ML Approaches [134, 100, 108, 103, 70, 99, 91, 129](10.5%)	• Automation of complex tasks (e.g., Chen et al. [134] for title completion) • Cross-modal information integration (e.g., Bo et al. [134] combined BKG knowledge graphs)	• High computational and data requirements (e.g., Chen et al. [99] processed 523,000 data entries) • Limited domain adaptability (e.g., Bo et al. [134] achieved a correctness rate of 28.52%)
Tool Development and Implementation [124, 78, 96, 80, 111, 103, 114](9.2%)	• Real-world validation capability (e.g., ARSearch [96] for cross-platform API matching) • Full-process support (e.g., Canary system [80] covering development, collaboration, and evaluation)	• High maintenance costs (e.g., synchronization of GitHub/Jenkins/Eclipse [80]) • Limited scalability (e.g., Mahajan et al.[111] focused on Java exceptions)
Qualitative Studies [102, 133, 123, 97, 50, 115, 114](9.2%)	• In-depth behavioral insights (e.g., Bidar et al.[102] conducted 36 interviews) • Fine-grained contextual analysis (e.g., Zhang et al.[115] analyzed code migration contexts)	• Limited sample size • Weak generalizability (constrained by participant backgrounds)

4.5.1. *Challenges and opportunities of Problem classification and feature extraction.*

In the field of problem classification and feature extraction, the main challenges are concentrated in areas such as subjective evaluation bias, limitations of the research context, insufficient coverage of data sources, data recognition accuracy, and limitations of classification methods.

Subjective evaluation bias. Subjective evaluation bias is a significant challenge in problem classification and feature extraction. Many studies rely on manual classification, labeling, and analysis of data, where different researchers may evaluate the relevance of the data according to their own standards, leading to inconsistent search results and affecting the accuracy of experimental outcomes. Although some studies use multiple authors to label the data, resolve disputes with arbitrators, and calculate consistency using the kappa coefficient to ensure labeling accuracy, this method effectively reduces bias, but there remains the potential for subjective influence [107].

Limitations of the research context. The limitations of the research context present a significant challenge in cross-platform research. Many studies analyze problems within specific domains or small groups, which limits the generalizability of the findings and may also be influenced by unobserved variables, thereby affecting the accuracy of the results. For example, some studies assume that developers choose projects based on their skills, but in reality, developers' choices may be influenced not only by their skills but also by factors such as personal interests and network relationships[101]. Moreover, different platforms, programming languages, datasets, and frameworks have distinct characteristics, making it difficult to generalize research results to other domains or platforms[86]. For instance, the retweet behavior on Twitter cannot be directly compared with the like behavior on Facebook, as different social media platforms operate under different conceptual frameworks and motivations[125].

Insufficient coverage of data sources. Insufficient coverage of data sources is a common issue in cross-platform research. User activity data is often not confined to a single platform but is widely distributed across multiple platforms. Although many studies primarily rely on Stack Overflow and GitHub as data sources, and these platforms provide representative datasets, some key issues may still be overlooked, as not all problems are discussed on these platforms[88, 98].

Data recognition accuracy. Due to the high degree of freedom in how users define problems, many issues are expressed without using explicit keywords, making it difficult to accurately identify problems that are vaguely phrased but closely related to the research topic[101, 10]. This vague expression complicates the selection of appropriate keywords during the search for related information, thus affecting the accuracy of information recognition and extraction.

Limitations of classification methods. Some studies classify posts and issues using labels and keywords, but new users may not use appropriate tags, and determining relevant keywords can be difficult[98]. As a result, many studies have turned to topic clustering methods, such as Latent Dirichlet Allocation (LDA), for classification. However, LDA also has several shortcomings. First, as a probabilistic model, LDA can produce different results when run multiple times on the same corpus[73, 15, 91]. Second, selecting the optimal number of topics is challenging because the topic inference process is subjective, which directly impacts the quality of the topics generated by LDA[120, 91]. Additionally, the poster may include a large amount of irrelevant content in their posts, introducing significant noise into the topic analysis performed by LDA[15]. Han et al.[15] further noted that the LDA model often blindly captures topics without considering the diversity of the dataset or domain-specific knowledge, resulting in topics that lack meaningful connections to actual domain concepts.

The main opportunities are primarily focused on increasing the diversity of data sources, optimizing data classification methods, and improving data recognition accuracy.

Increase the diversity of data sources. Increasing the diversity of data sources is one of the most critical opportunities identified by researchers. The researchers plan to analyze more relevant platforms and their available information resources[115], extend coverage to different programming languages and frameworks[96, 105]. Given that platforms will continue to generate new information, the research will also focus on continuously collecting and updating data to ensure its timeliness and comprehensiveness[107].

Optimize data classification methods. Future research should focus on improving the accuracy of topic classification for posts and issues[73, 115, 86], although no effective solutions have been proposed so far. Waseem et al.[117] suggest validating the classification methods for problems, causes, and solutions through industry surveys, seeking deeper insights from practitioners' perspectives. Additionally, Li et al. [73] plans to explore different clustering methods to assess whether more representative clustering results can be obtained.

Improve data recognition accuracy. Given that submitted information often lacks clear keyword descriptions, future research plans to improve the accuracy of information recognition by analyzing the entire content of posts[10]. Furthermore, the research will focus on further developing and optimizing automated validation mechanisms, using natural language processing techniques to identify and filter inaccurate or unclear responses, while also integrating user behavior data from the platform to enhance recognition accuracy[134]. Optimizing the selection of tag sets is also an important direction for improving the accuracy of issue recognition[73].

4.5.2. *Challenges and opportunities of Platform Collaboration.*

In the field of Platform Collaboration, there are challenges similar to those in problem classification and feature extraction, such as limitations in the research context, insufficient coverage of data sources, sample selection bias, data recognition accuracy, subjective evaluation bias, and the appropriateness of evaluation metrics. However, unlike in the problem classification and feature extraction domain, these challenges have not received the same level of widespread attention. Research under the theme of platform collaboration is more focused on exploring future opportunities.

Among these key challenges, two aspects are particularly noteworthy:

Insufficient coverage of data sources. Sahar et al. [127] point out that the rich volume of reports, citations, and discussions on social media platforms has not been fully utilized in existing research. Another overlooked aspect is deleted content, such as deleted tweets. Despite being removed, these pieces of information should not be underestimated, as they hold potential value [6].

Data recognition accuracy. Fang et al. [6] highlight that errors may occur when matching users across platforms. Additionally, Reinhardt et al. [103] note that while mining API usage patterns from GitHub projects can help detect API misuse in Stack Overflow code snippets, the common patterns extracted do not necessarily represent correct API usage, which may lead to false positives. The accuracy of data recognition directly impacts the reliability and validity of research outcomes.

In the field of Platform Collaboration, researchers have identified several areas that warrant further investigation. First, regarding information analysis and tool development, existing solutions struggle to effectively integrate large volumes of data and cannot accurately identify the most challenging technical help requests. Consequently, there is an urgent need to develop tools capable of automatically analyzing and summarizing platform content [127]. Second, the potential of platform support for bots remains underutilized. Researchers advocate incorporating more bots to automate routine collaboration tasks (e.g., automatically merging pull requests) and to facilitate coordination between experts and newcomers, thereby enhancing overall collaboration efficiency [127].

In addition, regarding the relationship between social media platforms and social coding platforms, it has been observed that many issues are cited only after a considerable delay, at which point the likelihood of their resolution increases. Researchers have called for a deeper investigation into the impact of the timing of issue citation on the resolution process within social coding platforms [127]. Moreover, over half of open-source projects do not utilize visible communication channels, which may negatively affect project efficiency and success rates, warranting further exploration [78]. Additionally, GitHub Sponsors official templates have a significant influence on social media activities. There is a need for empirical analysis to design more engaging social media content and to understand how open-source projects across various organizations and domains (e.g., security, machine learning) attract different types of sponsors [83]. Furthermore, the specific elements of tweets, such as whether they focus on a particular issue or pull request, and their influence on attracting new contributors, remain an area requiring further investigation [6, 128]. During the sharing of information on social media, personal opinions are often appended, potentially altering the original meaning and impact of the information. Consequently, it is necessary to study how this "information evolution" process affects collaboration and dissemination mechanisms [77]. At the same time, as privacy concerns grow increasingly prominent, it is imperative to explore ways to enhance the functionality of collaborative platforms while ensuring data security [75]. Finally, although the difference-in-differences (DiD) method is widely applied in causal inference research within social sciences, its use in software engineering remains relatively rare. Researchers are encouraged to adopt similar causal inference designs in software engineering contexts [6].

4.5.3. *Challenges and opportunities of Code Reuse and Evolution.*

In the study of code reuse and evolution, bias in code snippet sources and limitations of clone detection tools bias have emerged as two primary issues of focus.

Bias in code snippet sources. Existing studies often assume that code snippets are directly copied from Stack Overflow to GitHub projects. However, in reality, code snippets on GitHub may originate from various sources, such as tutorials, other GitHub projects. This assumption may result in insufficient data representativeness, leading to biases in the analysis of code reuse [119, 122, 13, 115].

Limitations of clone detection tools bias. The choice of clone detection tools can also introduce bias, as different tools adopt varying definitions of code clones, matching algorithms, and detection standards. Consequently, research findings that rely on specific clone detection tools may vary if alternative tools are employed [93].

In addition to addressing existing challenges, researchers have highlighted several opportunities to advance the field of code reuse and evolution.

Researchers have observed that code snippets from Stack Overflow are often modified for reasons related to security or correctness. As a result, these modifications may render corresponding snippets on GitHub outdated, posing potential risks to developers. To address this issue and assist developers in monitoring changes to Stack Overflow code snippets, researchers have proposed the development of dynamic update tools. Additionally, Baltes et al. [13] and Manes et al. [14] proposed the development of code version history datasets. By analyzing the historical versions of Stack Overflow snippets, these datasets can track the evolution of code, identify potential defects, and automatically flag erroneous versions. Such tools provide valuable insights to developers, enabling them to avoid replicating flawed code snippets. Given the significant variation in the quality and reliability of code snippets on Stack Overflow, researchers have also proposed leveraging the evolution history and usage history of Stack Overflow content to develop predictive models for assessing snippet quality. This kind of model can assist developers in determining whether a particular code snippet is sufficiently mature and suitable for use in their projects [14].

4.5.4. *Challenges and opportunities of User characterization.*

The evaluation of user characteristics faces significant challenges, particularly in addressing *user role differences*. Users in different roles exhibit distinct behavior patterns and data usage. For instance, non-technical employees, such as HR personnel, often lack access to technical repository data, resulting in differing usage patterns [130, 75]. To address this, Papoutsoglou et al. [75] proposed linking user roles to the thematic content they produce. Furthermore, Singer et al. [133] focused on the differences in social media usage among various types of developers. Their research aims to compare the social media usage patterns of web developers versus low-level systems programmers and users of static languages versus dynamic languages. Such analyses not only reveal the diversity of user characteristics but also provide researchers with a foundation for building more accurate user profile models.

Furthermore, researchers have identified two primary research opportunities: *consideration of new users* and *clarification of user skill requirements*.

Firstly, to enhance the representativeness and validity of research findings, the behaviors and characteristics of new users need to be taken into consideration [50]. Wan et al. [103] suggested that future research should focus on addressing the challenges posed by cold-start users, who are those with insufficient information available from any data source. For these users, research aims to infer their areas of expertise and interests from limited data, enabling a more comprehensive understanding of user characteristics and better addressing their needs.

The evaluation of user expertise primarily focuses on clarifying skill requirements and conducting an in-depth analysis of user characteristics. Vadlamani et al. [50] proposed the development of a cross-platform expertise framework encompassing a broader definition of "expertise". This framework aims to examine the attributes of experts who actively contribute across multiple platforms. Similarly, Croft et al. [91] emphasized the importance of conducting more extensive qualitative analyses or user studies to explore user expertise in greater detail, offering a comprehensive understanding of their skills and engagement levels. Meanwhile, Smirnova et al. [101] highlighted that founders and maintainers of open-source software (OSS) projects should clearly communicate the specific skills they require from contributors to enhance collaboration efficiency and drive project development.

4.5.5. *Challenges and opportunities of cross-platform data optimization.*

As programming tasks increasingly rely on data from multiple platforms, optimizing cross-platform data to enable effective retrieval and understanding has become a critical research area. One of the key challenges lies in addressing the semantic gap between user queries and relevant answers, particularly in programming-related contexts.

Semantic Gap Challenges. Traditional information retrieval (IR) methods typically rely on keyword matching; however, programming-related tasks exhibit significant linguistic discrepancies between queries and answers. Queries are often expressed in natural language, while answers may consist of code, technical jargon, or a combination of both. This semantic gap makes it challenging for simple keyword-based methods to effectively capture the deep relationships between user needs and potential solutions. Furthermore, programming tasks involve extensive use of domain-specific terminology (e.g., programming languages, library functions, and technical concepts), which increases the complexity for non-specialized systems and models to process effectively [129].

Improve the quality of real titles and utilize large language model technologies. To address these challenges, Chen et al. [99] proposed a semantic-based title completion method for GitHub issues and Stack Overflow posts. They plan to design novel evaluation strategies to measure the quality of generated titles through semantic consistency. Additionally, by leveraging advanced large language models, they aim to efficiently learn title generation knowledge using classification features from questions or posts and further train personalized models. This approach is expected to extend to title generation tasks across various domains in software engineering.

Optimize data classification methods. Furthermore, to address the limitations of traditional topic modeling methods (such as LDA) in classification tasks, Treude et al. [108] proposed that by optimizing topic model parameters, utilizing larger and more diverse corpora, and incorporating additional features, classification performance can be improved. They also suggested further exploring the optimal relationships between features and model configurations. These studies offer new directions for improving the utilization and optimization of cross-platform data.

> **Finding 4.** Based on the extracted challenges and opportunities, different research topics commonly face several technical limitations, such as subjective evaluation bias in manual data classification, insufficient data source coverage, and inaccurate data recognition. Beyond these shared constraints, each topic also presents unique challenges, research opportunities emphasize the need to enhance the diversity of data sources, improve data recognition accuracy, optimizing data classification methods, and clarifying user skill requirements.

5. Discussion

This section builds on the findings of the preceding research to conduct an in-depth discussion of the key challenges and potential opportunities identified in cross-platform studies. Targeted future research directions and practical recommendations are proposed, offering specific guidance for researchers, service providers, tool developers, and practitioners.

5.1. Future Agenda for Cross-Platform Studies

5.1.1. *Diversity of data sources*

The research findings (concerning RQ1 and RQ4) suggest that current cross-platform studies primarily rely on technical information, project/post/bug report metadata, interaction logs [101, 115, 92]. While these traditional data sources offer some insights into platform activities, their limitations are becoming increasingly apparent. For example, emerging data such as bots [138] and emojis [139] have yet to be fully explored for their potential value in cross-platform research, even though these factors are crucial for enhancing platform cohesion and long-term sustainability [139]. Furthermore, traditional data sources may be biased towards certain user groups, failing to reflect the diversity of heterogeneous platforms, which could lead to skewed research conclusions [103]. As platform ecosystems evolve rapidly, relying on a single data source increasingly struggles to address the complexity of dynamic interaction patterns, making the research outcomes less universally applicable.

Future research directions. A more comprehensive understanding of cross-platform research will require the integration of richer data sources, particularly emerging data such as bots and emojis. Additionally, research should extend its scope to include a diverse range of programming languages, platform types, and user groups [115, 96, 105, 107, 104].

5.1.2. *Employing multiple methods to address discontinuities and biases of data*

By analyzing the challenges identified in RQ4 and the public datasets compiled in RQ3, it was found that data discontinuities and biases significantly impact the reliability

Table 12
Summary Table of Research Topics, Challenges, Related Studies, and Opportunities

Research Topic	Challenges	Related Study	Opportunities	Related Study
Problem classification and feature extraction	Data recognition accuracy	[134, 120, 88, 104, 71, 91, 95, 113]	Improve data recognition accuracy	[134, 73, 112]
	Sample selection bias	[134, 104, 91]	Optimize the organizational structure of OSS projects	[101]
	High degree of freedom in issue descriptions	[134]	Increase the diversity of data sources	[115, 96, 105, 107, 104, 134, 102, 73, 15, 88, 91, 92]
	Limitations of the research context	[101, 96, 86, 105, 112, 87, 15, 71, 95, 113]	Optimize data classification methods	[73, 115, 86]
	Limitations of classification methods	[73, 123, 120, 15, 91]	Increase the diversity of error types	[105]
	Difficulty in recruiting interview subjects	[123]	Practical validation and real-world applications	[107]
	Appropriateness of evaluation metrics	[120, 71]	Explore the impact of document quality	[98]
	Completeness of information usage	[88]	Content analysis of social media platforms	[71]
	Data timeliness issues	[98]	Develop automation tools	[113, 112]
	Insufficient coverage of data sources	[101, 115, 92, 73, 107, 120, 88, 98, 117]		
	Subjective evaluation bias	[134, 115, 86, 92, 105, 107, 112, 123, 15, 88, 104, 117, 91, 95, 113]		
	effectiveness of the fix measures	[105]		
Platform Collaboration	Limitations of the research context	[130, 133, 100]	Develop automation tools	[127]
	Appropriateness of evaluation metrics	[128]	Expand the application scenarios of robotic tools	[127]
	Insufficient public information	[78]	Investigate the impact of citation timing on problem-solving efficiency	[127]
	Insufficient coverage of data sources	[83, 6, 128, 127]	Impact of multimodal data in project applications	[130]
	Sample selection bias	[133, 79]	The impact of the lack of social media use in research projects	[78]
	Data recognition accuracy	[6, 128, 103, 79]	Completeness of information usage	[130, 100]
	Low accuracy of semantic alignment	[100]	Increase the diversity of data sources	[83, 133, 100, 83]
	Difficulty in related code search	[110]	The role of GitHub templates in open-source projects	[83]
	Limitations of text similarity measurement methods	[110]	How organizations utilize social media	[83]
	Performance optimization challenges	[110]	Explore the impact of open-source project domains and functions	[83]
	Complexity of code fragment analysis	[118, 103]	Strengthen data privacy protection	[75]
	Subjective evaluation bias	[111]	Research on the application of differential design methods in software engineering	[6]
			Study the influencing factors in the evolution of information	[77]
			Improve data recognition accuracy	[111]
			Quantify the impact of social media on open-source platforms	[128]
			Practical validation and real-world applications	[79]
			Content analysis of social media platforms	[6, 128]
Code Reuse and Evolution	Lack of management tools for SO code snippet dependencies	[119]	Develop dynamic code snippet update tools	[119, 14]
	Bias in code snippet sources	[119, 122, 13, 115]	Create code version history datasets	[14, 13]
	Insufficient coverage of data sources	[14, 116]	Evaluate the quality of SO content	[119, 14]
	Broad definition of code snippet	[14]	Completeness of information usage	[122]
	Data recognition accuracy	[14, 90, 94, 116, 89]	Study the sources of code snippets	[122]
	Limitations of the research context	[13, 90, 94, 115]	Use reverse engineering techniques to identify missing code references	[13]
	Sample selection bias	[13, 93, 94, 115]	Increase the diversity of data sources	[93, 90, 89]
	Limitations of clone detection tools	[93, 90]	Enhance detection capabilities for Type III and IV code clones	[90]
	Subjective evaluation bias	[93, 89, 115]	Strengthen detection of code security and privacy protection	[94]
	Differences in data source versions	[93]	Analyze the impact of code snippet evolution	[116]
	Data source version tracking	[89]	Develop automated detection tools	[97]
			Address copyright and policy issues in code snippets	[107]
User Characterization	Inconsistent data	[131]	Analyze the user role lifecycle	[132]
	Data recognition accuracy	[131, 109, 74]	User role differences	[109]
	Insufficient coverage of data sources	[131, 132, 126, 106]	Completeness of information usage	[126, 74]
	Limitations of the research context	[124, 126, 106, 72]	Expand the boundaries of research domains	[106, 103, 72]
	User role differences	[124, 130, 75, 133]	Improve data recognition accuracy	[106]
	Interference from bots on GitHub	[109]	Consideration of new users	[103, 130]
	Sampling bias	[126, 50]	Analyze the trends in user interests and expertise over time	[103]
	Subjective evaluation bias	[50]	Explore the impact of user profiles and reputation on SO	[103]
	Vagueness in the definition of "active users"	[50]	Develop cross-platform user identification automation tools	[70]
			Increase the diversity of data sources	[50, 74]
			Clarify evaluation standards for professional knowledge	[50]
			Cross-platform knowledge transfer mechanism research	[50]
			Conduct large-scale quantitative research	[72]
			Clarify user skill requirements	[101, 91, 50]
Cross-platform Data Optimization	Appropriateness of evaluation metrics	[108]	Optimize data classification methods	[108]
	Limitations of the research context	[108]	Develop automation tools	[108]
	Data recognition accuracy	[121, 99]	Content analysis of social media platforms	[121]
	Subjective evaluation bias	[99]	Improve the quality of real titles	[99]
	Difficulty in semantic alignment of long-format data	[129]	Utilize large language model technologies	[99]
			Practical validation and real-world applications	[129]

of research. Due to the intermittent nature of user activity, researchers often rely on subjective judgment when selecting data collection periods, which leads to temporal bias in the data. Furthermore, the voluntary deletion of data by users and routine platform maintenance (e.g., removal of outdated or improperly formatted data) exacerbates the issues of data discontinuity and bias. Although previous studies have highlighted the impact of data loss and deletion on research outcomes [140, 131], effective solutions to these challenges remain largely unexplored.

Future research directions. To address these issues, future research could explore various data processing methods. For instance, weighted and sampling techniques could be employed in combination with K-means clustering-based neural network approaches [141] or multivariate interpolation methods to enhance data completeness and analytical accuracy. These techniques can effectively fill in missing data, reduce bias, and thereby improve the reliability of the research.

5.1.3. Considerations for dynamic heterogeneous graphs or networks

In the study of heterogeneous data across platforms, researchers typically analyze data from each platform independently before integrating these datasets. While this approach is simple and straightforward, it may overlook the consistency of user interactions and behaviors across platforms. Moreover, user interactions in open-source platforms naturally form graph structures, making heterogeneous information networks [142, 70] and Graph Convolutional Networks (GCN) [143] ideal tools for analysis. However, with the rapid development of open-source projects, collaboration patterns, technological preferences, trending topics, and the interests and expertise of platform developers are evolving rapidly. Current models based on heterogeneous networks have not yet fully captured the dynamic changes in user features. On the one hand, these models rarely explore the dynamic evolution of user characteristics; on the other hand, existing GCN models lack the ability for real-time incremental data acquisition, and their training speed still needs to be improved.

Future research directions. To address the above issues, future research could focus on improving the dynamic modeling of social graphs or networks. Specifically, there is a need to develop models capable of capturing the dynamic changes in user features. Additionally, enhancing the ability of Graph Convolutional Networks (GCNs) for real-time incremental data acquisition and accelerating their training speed would contribute to more efficient and accurate analysis of dynamic heterogeneous networks.

5.1.4. Identifying off-topic conversations or non-technical interactions

The data environment in open source platforms is inherently complex due to the abundance of unstructured information, such as irrelevant discussions and duplicate content. This information is often regarded as "noise," posing significant challenges to data processing. Current research primarily focuses on basic text preprocessing techniques, with limited exploration into effectively identifying and filtering non-technical interactions. Tao et al. [144] have made notable progress in this area by selecting high-quality commit messages as training samples and applying knowledge enhancement and dynamic denoising techniques, significantly improving the quality of the generated commit messages. According to the findings of RQ2, cross-platform studies often rely on diverse unstructured information to establish connections across platforms. Therefore, this approach holds potential for further application in cross-platform research, such as analyzing GitHub issues, README files, and Stack-Overflow posts, to more efficiently identify and connect related information across different platforms.

Future research directions. Building on the current research, future efforts should focus on more effectively identifying and filtering irrelevant discussions and non-technical interactions in open source platforms. This requires the development of intelligent algorithms that leverage deep learning and knowledge graph techniques to enhance the accuracy of noise filtering. Additionally, integrating multiple data types, such as code snippets, comments, and text, could further improve the models ability to understand and process complex data.

5.1.5. Strengthening the capacity to detect Type-3 and Type-4 clones

In the study of cross-platform code reuse, tools such as CCFinder [145] and SourcererCC [122] are commonly employed for code clone detection. Code clones are generally classified into four types [42]. Among these, CCFinder is effective at detecting Type-1 and Type-2 clones, while SourcererCC extends this capability to include Type-3 clones. However, according to the findings of RQ4, the performance of existing tools in detecting Type-3 clones across platforms remains limited and requires further improvement [122]. Type-3 clones involve more complex structural or syntactic modifications, making their accurate detection particularly critical as they are most likely to introduce errors in code repositories, potentially compromising software quality [?]. Additionally, Type-4 clones, characterized by semantic rather than syntactic similarity, pose even greater challenges for detection. In efforts to achieve a more comprehensive analysis of code reuse, accurately identifying and processing Type-3 and Type-4 clones, particularly those involving semantic similarities, remains a major challenge in current research. Existing detection methods exhibit significant limitations when addressing the semantic complexity and the high false positive rates often associated with Type-4 clones [146].

Future research directions. Future research should focus on enhancing the detection capabilities for Type-3 and Type-4 clones. For Type-3 clones, it is necessary to further improve existing algorithms and tools to enhance the accuracy and efficiency of cross-platform detection. For Type-4 clones, which are characterized by semantic similarity, more precise methods need to be developed to address their high false positive rates and complex semantic structures.

5.1.6. Exploring the connection between HuggingFace and GitHub

In recent years, collaborative practices within the open AI ecosystem have been rapidly emerging, with Hugging Face and GitHub playing significant roles in the construction, sharing, and maintenance of AI models [147]. Hugging Face, as a platform for showcasing and distributing AI models, hosts over 400,000 AI models, 150,000 applications, and 100,000 datasets[9], attracting widespread participation from developers [147]. Meanwhile, GitHub serves as a key platform for code hosting and collaboration, occupying a critical position in AI model development. However, despite the complementary functions of these two platforms, their specific interconnections have not been sufficiently explored, as highlighted by the platforms listed in RQ1.

The openness of AI models encompasses various dimensions, including training data, source code, model architectures, model parameters, documentation, and associated licensing [148]. These components are often distributed across different platforms, such as code being hosted on GitHub and models being published on Hugging Face [149]. While this cross-platform distribution enhances collaboration flexibility, it also significantly increases the complexity of collaboration. For instance, the interaction between components across platforms and the effective management of data flow remain underexplored, with no clear research framework established. Furthermore, these components frequently adopt different open-source licenses, and compatibility issues between these licenses could affect the usability of AI software and the redevelopment of models. Identifying these cross-platform distributed components and systematically analyzing their potential impact on the efficiency of AI project development and the health of the ecosystem remain critical challenges for current research.

Future research directions. Future research should focus on exploring the collaborative dynamics between Hugging Face and GitHub. This includes investigating the collaboration patterns of the same project across different platforms and analyzing their practical implications for AI model development and sharing [147]. Additionally, to address issues related to open-source license compatibility, future studies should develop systematic methods to identify distributed open-source components across platforms and conduct in-depth analyses of license compatibility to facilitate more efficient model redevelopment and integration of AI software.

5.2. Implications and Practical Recommendations

5.2.1. Implications to developers

This review provides guidance for developers in addressing relevant issues and promoting open-source projects. It highlights key considerations when reusing code snippets during the development process, such as tracking source code updates, ensuring security and quality, and adhering to copyright compliance. By addressing these aspects, developers can more effectively mitigate potential risks associated with code reuse.

5.2.2. Implications to researchers

Although cross-platform research has gained some attention in recent years, many challenging and unexplored areas remain. Researchers can build on the future research directions proposed in this study to further expand relevant research. Additionally, RQ2 summarized the types of information that cross-platform research relies on, while RQ3 provided an overview of existing public datasets and research methods, offering researchers convenient guidance for conducting related studies. Furthermore, RQ3 revealed that existing cross-platform research tools, such as GrimoireLab [150], have not been fully utilized. GrimoireLab is capable of automatically and incrementally collecting data from various platforms, including version control systems, issue tracking systems, and forums [151]. This functionality addresses the major challenge of insufficient data sources in cross-platform research, enabling more comprehensive data collection and analysis. Future researchers are encouraged to effectively integrate such tools into cross-platform data analysis workflows to enhance research efficiency and data coverage.

5.2.3. Implications to service/tool providers

The findings of RQ4 indicate that existing studies predominantly rely on manual analysis, which introduces significant subjective evaluation biases [134, 115, 86]. Future research should focus on developing automated analysis and information extraction tools, as well as training classification tools to improve the efficiency of issue resolution [127]. Moreover, there is a lack of tools for managing dependencies on code snippets, particularly those sourced from platforms like Stack Overflow. Currently, no tools exist to effectively manage these dependencies or track updates and security discussions related to such code snippets [119]. Developing these tools could enhance the reliability and maintainability of software projects and represents a promising research and development direction for service and tool providers.

6. Threats to validity

This section is divided into four parts based on the guidelines proposed by Runeson et al. [152], including construct validity, internal validity, external validity, and reliability.

6.1. Construct validity

In our study, a significant threat to validity arises from the fact that many relevant papers do not explicitly mention cross-platform related search terms, but instead use specific platform names. This practice limits the literature retrieval process and may cause us to overlook relevant studies, thereby affecting the comprehensiveness and accuracy of the research findings. To mitigate this threat, we first constructed

[9] https://huggingface.co/

an initial search string using cross-platform keywords, then applied Named Entity Recognition to extract relevant entities from the titles and abstracts of the collected papers. After manual review, we identified 19 common platform-related entities and iteratively refined the search string based on these entities to conduct a more comprehensive literature search. This process effectively alleviated the limitations in literature retrieval and enhanced the breadth of literature coverage.

6.2. Internal validity

Paper searching. Selection bias may occur during the paper screening phase due to the personal preferences and diverse background knowledge of the researchers, which could lead to the exclusion of essential studies. To minimize selection bias and ensure the reliability of our selection process, the first two authors independently reviewed a randomly selected subset of papers, assessing the consistency of their inclusion decisions. Inconsistencies were discussed, leading to a unified outcome.

Data extraction. At the same time, we clearly listed the specific data to be extracted from each paper, and from which section these data should be obtained, to minimize the risk of omitting relevant data.

Data analysis. To alleviate the impact of personal bias in addressing the data analysis process, we employed the open card sorting method to categorize data relevant to each research question. Furthermore, in order to decrease potential misinterpretation of the experimental design and analytical methods used in the related study, we conducted additional validations and held several discussions.

6.3. External validity

Our review is focused on cross-platform research in the open-source domain. Although our study does not extend to interactions among platforms such as YouTube, the platform connectivity strategies and analysis methods we have summarized primarily utilize user behavioral data within the platforms involved in our study. Consequently, our findings may offer valuable insights for understanding interactions across various online platforms.

6.4. Reliability

To enhance the replicability of our findings, we have shared every aspect of our research process in our open-source project. This includes the search strings used for each database and the papers retrieved at each stage.

7. Conclusion

This paper provides a systematic review of the current state and evolution of cross-platform research in open-source platforms, with a focus on social coding, social Q&A, and social media platforms. We analyze the types of cross-platform connections, key research themes, and commonly used experimental designs, while extracting the opportunities and challenges highlighted in relevant studies. The research identifies several key areas in cross-platform research, including problem classification and feature extraction, platform collaboration, code reuse and evolution, user characterization, and cross-platform data optimization. Additionally, this study summarizes 40 publicly available datasets and categorizes research methods into data-driven methods, qualitative studies, modeling & ml approaches, and tool development and implementation.

Based on the challenges and opportunities identified, we propose six future research directions and practical recommendations, aiming to provide comprehensive guidance for researchers and to promote further exploration and development in this field.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 62332005).

References

[1] L. F. Dias, I. Steinmacher, G. Pinto, D. Alencar Da Costa, M. Gerosa, How does the shift to github impact project collaboration?, in: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2016, pp. 473–477. doi:10.1109/ICSME.2016.78.

[2] X. Song, J. Yan, Y. Huang, H. Sun, H. Zhang, A collaboration-aware approach to profiling developer expertise with cross-community data, in: 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS), 2022, pp. 344–355. doi:10.1109/QRS57517.2022.00043.

[3] B. Vasilescu, A. Serebrenik, P. Devanbu, V. Filkov, How social q&a sites are changing knowledge sharing in open source software communities, in: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 342354. URL: https://doi.org/10.1145/2531602.2531659. doi:10.1145/2531602.2531659.

[4] Adrian Twarog, StackOverflow isn't as useful anymore? I use GitHub more often, https://dev.to/adriantwarog/stackoverflow-isn-t-as-useful-anymore-i-use-github-more-often-638, 2020. [Online; accessed: 2024-05-16].

[5] H. Kwak, C. Lee, H. Park, S. Moon, What is twitter, a social network or a news media?, in: Proceedings of the 19th International Conference on World Wide Web, WWW '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 591600. URL: https://doi.org/10.1145/1772690.1772751. doi:10.1145/1772690.1772751.

[6] H. Fang, H. Lamba, J. Herbsleb, B. Vasilescu, "this is damn slick!": Estimating the impact of tweets on open source project popularity and new contributors, in: ICSE '22: Proceedings of the 44th International Conference on Software Engineering, 2022.

[7] H. Huang, Y. Lu, X. Mao, Gathering github oss requirements from q&a community: An empirical study, in: 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), 2020.

[8] S. Islam, Y. S. Nugroho, C. M. Shahrear, N. Wahed, D. Gunawan, E. W. Pamungkas, M. H. Kabir, Y. I. Kurniawan, M. K. Uddin, An empirical study of software ecosystem related tweets by npm maintainers, PeerJ Computer Science 10 (2024) e1669.

[9] M. Tanzil, S. Chowdhury, S. Modaberi, G. Uddin, H. Hemmati, A systematic mapping study of crowd knowledge enhanced software engineering research using stack overflow, arXiv preprint arXiv:2408.07913 (2024).

[10] H. Hong, S. Woo, E. Choi, J. Choi, H. Lee, xvdb: A high-coverage approach for constructing a vulnerability database, IEEE Access 10 (2022) 85050–85063.

[11] S. Baltes, C. Treude, S. Diehl, Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets, in: 2019 IEEE/ACM

16th International Conference on Mining Software Repositories (MSR), IEEE, 2019, pp. 191–194.

[12] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, R. Oliveto, Toxic code snippets on stack overflow, IEEE Transactions on Software Engineering 47 (2019) 560–581.

[13] S. Baltes, S. Diehl, Usage and attribution of stack overflow code snippets in github projects, Empirical Software Engineering 24 (2019) 1259–1295.

[14] S. S. Manes, O. Baysal, Studying the change histories of stack overflow and github snippets, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), 2021, pp. 283–294. doi:10.1109/MSR52588.2021.00040.

[15] J. Han, E. Shihab, Z. Wan, S. Deng, X. Xia, What do programmers discuss about deep learning frameworks, Empirical Software Engineering 25 (2020) 2694–2747.

[16] Y. Wu, J. Kropczynski, P. C. Shih, J. M. Carroll, Exploring the ecosystem of software developers on github and other platforms, in: Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing, 2014, pp. 265–268.

[17] S. Yao, Cross_platform_study, https://github.com/YovM/Cross_Platform_Study, 2025.

[18] S. Yao, X. Zhang, Y. Zhang, T. Wang, Open-source_crossplatform_survey_data, 2025. doi:10.17632/m5hmnmxndr.2.

[19] GitHub, Inc., Let's build from here, https://github.com/about, 2024. [Online; accessed: 2024-05-11].

[20] S. Ovadia, Linux for academics, part ii: The advantages of free and open-source software, Behavioral & Social Sciences Librarian 33 (2014) 47–51.

[21] G. Schryen, R. Kadura, Open source vs. closed source software: towards measuring security, in: Proceedings of the 2009 ACM symposium on Applied Computing, 2009, pp. 2016–2023.

[22] D. Bosio, B. Littlewood, L. Strigini, M. Newby, Advantages of open source processes for reliability: clarifying the issues, in: Proceedings of the Open Source Software Development Workshop, 2002, pp. 30–46.

[23] The Linux Foundation, 6 Reasons Why Open Source Software Lowers Development Costs, https://www.linuxfoundation.org/blog/blog/6-reasons-why-open-source-software-lowers-development-costs, 2024. [Online; accessed: 2024-05-13].

[24] G. Gousios, M. Pinzger, A. v. Deursen, An exploratory study of the pull-based software development model, in: Proceedings of the 36th international conference on software engineering, 2014, pp. 345–355.

[25] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, V. Filkov, Quality and productivity outcomes relating to continuous integration in github, in: Proceedings of the 2015 10th joint meeting on foundations of software engineering, 2015, pp. 805–816.

[26] M. Wessel, J. Vargovich, M. A. Gerosa, C. Treude, Github actions: the impact on the pull request process, Empirical Software Engineering 28 (2023) 131.

[27] M. Wessel, T. Mens, A. Decan, P. R. Mazrae, The github development workflow automation ecosystems, in: Software Ecosystems: Tooling and Analytics, Springer, 2023, pp. 183–214.

[28] Y. Yu, H. Wang, G. Yin, T. Wang, Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment?, Information and software technology 74 (2016) 204–218.

[29] Y. Zhang, H. Wang, G. Yin, T. Wang, Y. Yu, Social media in github: the role of@-mention in assisting software development, Science China Information Sciences 60 (2017) 1–18.

[30] L. Li, Z. Ren, X. Li, W. Zou, H. Jiang, How are issue units linked? empirical study on the linking behavior in github, in: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), IEEE, 2018, pp. 386–395.

[31] G. Gousios, D. Spinellis, Ghtorrent: Github's data from a firehose, in: 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), IEEE, 2012, pp. 12–21.

[32] Matej Bačo, Why Discord Is a Must-Have for OSS, https://dev.to/appwrite/why-discord-is-a-must-have-for-oss-2jpj, 2022. [Online; accessed: 2024-05-16].

[33] Z. Zhao, Q. Yang, D. Cai, X. He, Y. Zhuang, Expert finding for community-based question answering via ranking metric network learning., in: Ijcai, volume 16, 2016, pp. 3000–3006.

[34] H. Yuan, A. A. Hernandez, User cold start problem in recommendation systems: A systematic review, IEEE Access 11 (2023) 136958–136977.

[35] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Information and software technology 51 (2009) 7–15.

[36] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and software technology 64 (2015) 1–18.

[37] B. Lin, N. Cassee, A. Serebrenik, G. Bavota, N. Novielli, M. Lanza, Opinion mining for software development: a systematic literature review, ACM Transactions on Software Engineering and Methodology (TOSEM) 31 (2022) 1–41.

[38] R. Santos, P. Soares, E. Rodrigues, P. H. M. Maia, A. Silveira, How blockchain and microservices are being used together: a systematic mapping study, in: Proceedings of the 5th International Workshop on Emerging Trends in Software Engineering for Blockchain, 2022, pp. 39–46.

[39] C. H. C. Duarte, The quest for productivity in software engineering: A practitioners systematic literature review, in: 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), IEEE, 2019, pp. 145–154.

[40] F. Basso, B. M. Soares Ferreira, R. Torres, R. Z. Frantz, D. Kreutz, M. Bernardino, E. de Macedo Rodrigues, Model-driven integration and the oslc standard: a mapping of applied studies, in: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, 2023, pp. 763–770.

[41] P. R. I. Gomes, M. S. d. Castro, T. H. Nascimento, Gesture recognition methods using sensors integrated into smartwatches: Results of a systematic literature review, in: Proceedings of the XXII Brazilian Symposium on Human Factors in Computing Systems, 2023, pp. 1–11.

[42] C. Liu, X. Xia, D. Lo, C. Gao, X. Yang, J. Grundy, Opportunities and challenges in code search tools, ACM Computing Surveys (CSUR) 54 (2021) 1–40.

[43] M. Alhindi, J. Hallett, Sandboxing adoption in open source ecosystems, in: Proceedings of the 12th ACM/IEEE International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems, 2024, pp. 13–20.

[44] R. C. Borges, M. d. G. Malheiros, C. Z. Billa, M. R. Pias, A. d. L. Bicho, An open-source framework using webrtc for online multiplayer gaming, in: Proceedings of the 22nd Brazilian Symposium on Games and Digital Entertainment, 2023, pp. 143–150.

[45] R. Team, Advantages and disadvantages of google scholar, 2024. URL: https://barrazacarlos.com/advantages-and-disadvantages-of-google-scholar/, explores the pros and cons of Google Scholar, including its wide accessibility and citation tracking features, as well as limitations such as quality control issues and lack of advanced filtering options.

[46] G. Times, Does google scholar have peer-reviewed articles?, 2024. URL: https://gbtimes.com/does-google-scholar-have-peer-reviewed-articles/, discusses the presence of peer-reviewed and non-peer-reviewed articles in Google Scholar's database, with approximate statistics provided.

[47] N. Grattan, D. A. da Costa, N. Stanger, The need for more informative defect prediction: A systematic literature review, Information and Software Technology (2024) 107456.

[48] A. S. Guinea, G. Nain, Y. Le Traon, A systematic review on the engineering of software for ubiquitous systems, Journal of Systems and Software 118 (2016) 251–276.

[49] B. Vasilescu, V. Filkov, A. Serebrenik, Stackoverflow and github: Associations between software development and crowdsourced knowledge, in: 2013 International Conference on Social Computing, IEEE, 2013, pp. 188–195.

[50] S. L. Vadlamani, O. Baysal, Studying software developer expertise and contributions in stack overflow and github, in: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2020, pp. 312–323.

[51] H. Wang, T. Wang, G. Yin, C. Yang, Linking issue tracker with q&a sites for knowledge sharing across communities, IEEE Transactions on Services Computing 11 (2015) 782–795.

[52] C. Tan, L. Lee, All who wander: On the prevalence and characteristics of multi-community engagement, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 1056–1066.

[53] A. S. Badashian, A. Esteki, A. Gholipour, A. Hindle, E. Stroulia, Involvement, contribution and influence in github and stack overflow, in: CASCON, 2014, pp. 19–33.

[54] G. Silvestri, J. Yang, A. Bozzon, A. Tagarelli, et al., Linking accounts across social networks: the case of stackoverflow, github and twitter., in: KDWeb, 2015, pp. 41–52.

[55] N. McDonald, S. Goggins, Performance and participation in open source software on github, in: CHI '13 Extended Abstracts on Human Factors in Computing Systems, CHI EA '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 139144. URL: https://doi.org/10.1145/2468356.2468382. doi:10.1145/2468356.2468382.

[56] E. Kalliamvakou, D. Damian, K. Blincoe, L. Singer, D. M. German, Open source-style collaborative development practices in commercial projects using github, in: 2015 IEEE/ACM 37th IEEE international conference on software engineering, volume 1, IEEE, 2015, pp. 574–585.

[57] F. Qi, X.-Y. Jing, X. Zhu, X. Xie, B. Xu, S. Ying, Software effort estimation based on open source projects: Case study of github, Information and Software Technology 92 (2017) 145–157.

[58] Y. Xiong, Z. Meng, B. Shen, W. Yin, Developer identity linkage and behavior mining across github and stackoverflow, International Journal of Software Engineering and Knowledge Engineering 27 (2017) 1409–1425.

[59] W. Viechtbauer, L. Smits, D. Kotz, L. Budé, M. Spigt, J. Serroyen, R. Crutzen, A simple formula for the calculation of sample size in pilot studies, Journal of clinical epidemiology 68 (2015) 1375–1379.

[60] J. R. Landis, G. G. Koch, The measurement of observer agreement for categorical data, biometrics (1977) 159–174.

[61] L. Yang, H. Zhang, H. Shen, X. Huang, X. Zhou, G. Rong, D. Shao, Quality assessment in systematic literature reviews: A software engineering perspective, Information and Software Technology 130 (2021) 106397.

[62] C. Gomes, J. P. Fernandes, G. Falcao, S. Kar, S. Tayur, A systematic mapping study on quantum and quantum-inspired algorithms in operations research, ACM Computing Surveys 57 (2024) 1–35.

[63] T. Wood, S. Perera, S. Yan, L. Padgham, A. Moffat, Core rankings portal, https://www.core.edu.au/home, 2021. Accessed: 2021-05.

[64] S. Elder, M. R. Rahman, G. Fringer, K. Kapoor, L. Williams, A survey on software vulnerability exploitability assessment, ACM Computing Surveys 56 (2024) 1–41.

[65] D. Amalfitano, S. Faralli, J. C. R. Hauck, S. Matalonga, D. Distante, Artificial intelligence applied to software testing: A tertiary study, ACM Computing Surveys 56 (2023) 1–38.

[66] S. Mosikyan, R. Dolan, A. M. Corsi, S. Bastian, A systematic literature review and future research agenda to study consumer acceptance of novel foods and beverages, Appetite (2024) 107655.

[67] S. Keele, et al., Guidelines for performing systematic literature reviews in software engineering, Technical Report, Technical report, ver. 2.3 ebse technical report. ebse, 2007.

[68] T. Zimmermann, Card-sorting: From text to themes, in: Perspectives on data science for software engineering, Elsevier, 2016, pp. 137–141.

[69] S. H. Khandkar, Open coding, University of Calgary 23 (2009) 2009.

[70] Y. Fan, Y. Zhang, S. Hou, L. Chen, Y. Ye, C. Shi, L. Zhao, S. Xu, idev: Enhancing social coding security by cross-platform user identification between github and stack overflow, in: 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019, 2019.

[71] Z. Zhao, Y. Chen, A. A. Bangash, B. Adams, A. E. Hassan, An empirical study of challenges in machine learning asset management, Empirical Software Engineering 29 (2024) 98.

[72] R. Saxena, N. Pedanekar, I know what you coded last summer: Mining candidate expertise from github repositories, in: Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, 2017, pp. 299–302.

[73] H. Li, F. Khomh, M. Openja, et al., Understanding quantum software engineering challenges an empirical study on stack exchange forums and github issues, in: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2021, pp. 343–354.

[74] M. R. Masud, B. Treves, M. Faloutsos, Disambiguating usernames across platforms: the geekman approach, Social Network Analysis and Mining 14 (2024) 177.

[75] M. Papoutsoglou, J. Wachs, G. M. Kapitsaki, Mining dev for social and technical insights about software development, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 415–419.

[76] H. Jiang, L. Shi, M. Che, Y. Zhang, Q. Wang, Bringing open source communication and development together: A cross-platform study on gitter and github, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 50 (2024) 2807–2826.

[77] P. Shrestha, A. Sathanur, S. Maharjan, E. Saldanha, D. Arendt, S. Volkova, Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of github, twitter and reddit, PLOS ONE 15 (2020) e0230250.

[78] V. Käfer, D. Graziotin, I. Bogicevic, S. Wagner, J. Ramadani, Poster: Communication in open-source projectsend of the e-mail era? (2018) 242–243.

[79] M. Raglianti, C. Nagy, R. Minelli, B. Lin, M. Lanza, On the rise of modern software documentation (pearl/brave new idea), in: 37th European Conference on Object-Oriented Programming (ECOOP 2023), Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.

[80] S. Heckman, J. King, Developing software engineering skills using real tools for automated grading, in: Proceedings of the 49th ACM technical symposium on computer science education, 2018, pp. 794–799.

[81] M. Wu, R. Aranovich, V. Filkov, Evolution and differentiation of the cybersecurity communities in three social question and answer sites: A mixed-methods analysis, Plos one 16 (2021) e0261954.

[82] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, B. Hartmann, Design lessons from the fastest q&a site in the west, in: Proceedings of the SIGCHI conference on Human factors in computing systems, 2011, pp. 2857–2866.

[83] Y. Fan, T. Xiao, H. Hata, C. Treude, K. Matsumoto, " my github sponsors profile is live!" investigating the impact of twitter/x mentions on github sponsors, in: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1–12.

[84] C. Tranoris, S. Denazis, Smart city issue management: Extending and adapting a software bug tracking system, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2018, pp. 1264–1270.

[85] Jenkins Project, Jenkins: Build great things at any scale, https://jenkins.io/, 2017. Accessed: 2017.

[86] A. Galappaththi, S. Nadi, C. Treude, An empirical study of api misuses of data-centric libraries, in: Proceedings of the 18th ACM/IEEE

International Symposium on Empirical Software Engineering and Measurement, 2024, pp. 245–256.

[87] M. J. Islam, G. Nguyen, R. Pan, H. Rajan, A comprehensive study on deep learning bug characteristics, in: Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, 2019, pp. 510–520.

[88] M. Bagherzadeh, N. Fireman, A. Shawesh, R. Khatchadourian, Actor concurrency bugs: a comprehensive study on symptoms, root causes, api usages, and differences, Proceedings of the ACM on Programming Languages 4 (2020) 1–32.

[89] M. Verdi, A. Sami, J. Akhondali, F. Khomh, G. Uddin, A. K. Motlagh, An empirical study of c++ vulnerabilities in crowd-sourced code examples, IEEE Transactions on Software Engineering 48 (2020) 1497–1514.

[90] X. Chen, F. Xu, Y. Huang, X. Zhou, Z. Zheng, An empirical study of code reuse between github and stack overflow during software development, Journal of Systems and Software 210 (2024) 111964.

[91] R. Croft, Y. Xie, M. Zahedi, M. A. Babar, C. Treude, An empirical study of developers discussions about security challenges of different programming languages, Empirical Software Engineering 27 (2022) 1–52.

[92] Z. Chen, H. Yao, Y. Lou, Y. Cao, Y. Liu, H. Wang, X. Liu, An empirical study on deployment faults of deep learning based mobile applications, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 674–685.

[93] M. R. H. Misu, A. Satter, An exploratory study of analyzing javascript online code clones, in: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, 2022, pp. 94–98.

[94] T. Zhang, D. Yang, C. Lopes, M. Kim, Analyzing and supporting adaptation of online code examples, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, 2019, pp. 316–327.

[95] B. Zhang, T. Liu, P. Liang, C. Wang, M. Shahin, J. Yu, Architecture decisions in ai-based systems development: an empirical study, in: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2023, pp. 616–626.

[96] K. Luong, F. Thung, D. Lo, Arsearch: searching for api related resources from stack overflow and github, in: Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings, 2022, pp. 11–15.

[97] S. Baltes, R. Kiefer, S. Diehl, Attribution required: Stack overflow code snippets in github projects, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), IEEE, 2017, pp. 161–163.

[98] P. Chakraborty, R. Shahriyar, A. Iqbal, G. Uddin, How do developers discuss and support new programming languages in technical q&a site? an empirical study of go, swift, and rust in stack overflow, Information and Software Technology 137 (2021) 106603.

[99] X. Chen, W. Pei, S. Yang, Y. Zhou, Z. Zhang, J. Pei, Automatic title completion for stack overflow posts and github issues, Empirical Software Engineering 29 (2024) 120.

[100] I. C. Irsan, T. Zhang, F. Thung, K. Kim, D. Lo, Picaso: enhancing api recommendations with relevant stack overflow posts, in: 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), IEEE, 2023, pp. 92–37.

[101] I. Smirnova, M. Reitzig, O. Alexy, What makes the right oss contributor tick? treatments to motivate high-skilled developers, Research Policy 51 (2022) 104368.

[102] R. Bidar, M. Jabbari, E. Luck, Value co-destruction in online collaborative networks, European Management Journal 42 (2024) 98–107.

[103] A. Reinhardt, T. Zhang, M. Mathur, M. Kim, Augmenting stack overflow with api usage patterns mined from github, in: Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, 2018, pp. 880–883.

[104] C. Wang, Z. Chen, M. Zhou, Automl from software engineering perspective: Landscapes and challenges, in: 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), IEEE, 2023, pp. 39–51.

[105] M. M. Morovati, A. Nikanjam, F. Khomh, Z. M. Jiang, Bugs in machine learning-based systems: a faultload benchmark, Empirical Software Engineering 28 (2023) 62.

[106] A. S. Badashian, A. Hindle, E. Stroulia, Crowdsourced bug triaging, in: 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2015, pp. 506–510.

[107] X. Zhou, P. Liang, B. Zhang, Z. Li, A. Ahmad, M. Shahin, M. Waseem, Exploring the problems, their causes and solutions of ai pair programming: A study on github and stack overflow, Journal of Systems and Software 219 (2025) 112204.

[108] C. Treude, M. Wagner, Predicting good configurations for github and stack overflow topic models, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), IEEE, 2019, pp. 84–95.

[109] V. Klotzman, F. Farmahinifarahani, C. Lopes, Public software development activity during the pandemic, in: Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2021, pp. 1–12.

[110] M. M. Rahman, C. K. Roy, D. Lo, Rack: Code search in the ide using crowdsourced knowledge, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), IEEE, 2017, pp. 51–54.

[111] S. Mahajan, N. Abolhassani, M. R. Prasad, Recommending stack overflow posts for fixing runtime exceptions using failure scenario matching, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 1052–1064.

[112] M. J. Islam, R. Pan, G. Nguyen, H. Rajan, Repairing deep neural networks: Fix patterns and challenges, in: Proceedings of the ACM/IEEE 42nd international conference on software engineering, 2020, pp. 1135–1146.

[113] X. Liu, D. Gu, Z. Chen, J. Wen, Z. Zhang, Y. Ma, H. Wang, X. Jin, Rise of distributed deep learning training in the big model era: From a software engineering perspective, ACM Transactions on Software Engineering and Methodology 32 (2023) 1–26.

[114] Y. Wang, Y. Wang, S. Wang, Y. Liu, C. Xu, S.-C. Cheung, H. Yu, Z. Zhu, Runtime permission issues in android apps: Taxonomy, practices, and ways forward, IEEE Transactions on Software Engineering 49 (2022) 185–210.

[115] T. Zhang, Y. Lu, Y. Yu, X. Mao, Y. Zhang, Y. Zhao, How do developers adapt code snippets to their contexts? an empirical study of context-based code snippet adaptations, IEEE Transactions on Software Engineering (2024).

[116] S. S. Manes, O. Baysal, How often and what stackoverflow posts do developers reference in their github projects?, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), IEEE, 2019, pp. 235–239.

[117] M. Waseem, T. Das, A. Ahmad, P. Liang, T. Mikkonen, Issues and their causes in webassembly applications: An empirical study, in: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, 2024, pp. 170–180.

[118] S. Subramanian, L. Inozemtseva, R. Holmes, Live api documentation, in: Proceedings of the 36th international conference on software engineering, 2014, pp. 643–652.

[119] A. Jallow, M. Schilling, M. Backes, S. Bugiel, Measuring the effects of stack overflow code snippet evolution on open-source software security (2024).

[120] C. Zimmerle, K. Gama, F. Castor, J. M. M. Filho, Mining the usage of reactive programming apis: a study on github and stack overflow, in: Proceedings of the 19th International Conference on Mining Software Repositories, 2022, pp. 203–214.

[121] V. Singh, S. Chakraborty, A. Kadian, The effect of knowledge sharing on open source contribution: a multi-platform perspective (2020).

[122] D. Yang, P. Martins, V. Saini, C. Lopes, Stack overflow in github: any snippets there?, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), IEEE, 2017, pp. 280–290.
[123] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, P. Tonella, Taxonomy of real faults in deep learning systems, in: Proceedings of the ACM/IEEE 42nd international conference on software engineering, 2020, pp. 1110–1121.
[124] S. K. Kuttal, X. Chen, Z. Wang, S. Balali, A. Sarma, Visual resume: Exploring developers online contributions for hiring, Information and Software Technology 138 (2021) 106633.
[125] C. Flores-Saviaga, S. Savage, Fighting disaster misinformation in latin america: the# 19s mexican earthquake case study, Personal and Ubiquitous Computing 25 (2021) 353–373.
[126] C. Hundhausen, P. Conrad, O. Adesope, A. Tariq, Combining github, chat, and peer evaluation data to assess individual contributions to team software development projects, ACM Transactions on Computing Education 23 (2023) 1–23.
[127] H. Sahar, A. Hindle, C.-P. Bezemer, How are issue reports discussed in gitter chat rooms?, Journal of Systems and Software 172 (2021) 110852.
[128] H. Jiang, L. Shi, M. Che, Y. Zhang, Q. Wang, Bringing open source communication and development together: A cross-platform study on gitter and github, IEEE Transactions on Software Engineering (2024).
[129] M. Chen, G. Li, C. Ma, J. Li, H. Fu, Repo4qa: Answering coding questions via dense retrieval on github repositories, in: 29th International Conference on Computational Linguistics (COLING 2022), 2022, pp. 1580–1592.
[130] E. Koshchenko, E. Klimov, V. Kovalenko, Multimodal recommendation of messenger channels, in: Proceedings of the 19th International Conference on Mining Software Repositories, 2022, pp. 495–505.
[131] H. Fang, D. Klug, H. Lamba, J. Herbsleb, B. Vasilescu, Need for tweet: How open source developers talk about their github work on twitter, in: MSR '20: Proceedings of the 17th International Conference on Mining Software Repositories, 2020.
[132] M. A. Harysi, B. Negoita, J. Nandhakumar, The evolution of leadership structures in online communities: A social network perspective (2019).
[133] L. Singer, F. Figueira Filho, M.-A. Storey, Software engineering at the speed of light: how developers stay current using twitter, in: Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 211–221.
[134] L. Bo, Y. He, X. Sun, W. Ji, X. Wu, A software bug fixing approach based on knowledge-enhanced large language models, in: 2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2024, pp. 169–179.
[135] Z. Kotti, R. Galanopoulou, D. Spinellis, Machine learning for software engineering: A tertiary study, ACM Computing Surveys 55 (2023) 1–39.
[136] G. Gousios, The ghtorent dataset and tool suite, in: 2013 10th Working Conference on Mining Software Repositories (MSR), IEEE, 2013, pp. 233–236.
[137] J. Martin, J. L. Guo, Deep api learning revisited, in: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, 2022, pp. 321–330.
[138] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, M. A. Gerosa, The power of bots: Characterizing and understanding bots in oss projects, Proceedings of the ACM on Human-Computer Interaction 2 (2018) 1–19.
[139] Y. Zhou, X. Lu, G. Gao, Q. Mei, W. Ai, Emoji promotes developer participation and issue resolution on github, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 18, 2024, pp. 1833–1846.
[140] J. Howison, A. Wiggins, K. Crowston, Validity issues in the use of social network analysis with digital trace data, Journal of the Association for Information Systems 12 (2011) 2.
[141] B. Yu, C. Zhang, Z. Tang, Missing data processing based on deep neural network enhanced by k-means, in: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, 2019, pp. 151–155.
[142] J. Yan, H. Sun, X. Wang, X. Liu, X. Song, Profiling developer expertise across software communities with heterogeneous information network analysis, in: Proceedings of the 10th Asia-Pacific Symposium on Internetware, Internetware '18, Association for Computing Machinery, New York, NY, USA, 2018. URL: https://doi.org/10.1145/3275219.3275226. doi:10.1145/3275219.3275226.
[143] H. Shao, D. Sun, J. Wu, Z. Zhang, A. Zhang, S. Yao, S. Liu, T. Wang, C. Zhang, T. Abdelzaher, paper2repo: Github repository recommendation for academic papers, in: Proceedings of The Web Conference 2020, 2020, pp. 629–639.
[144] W. Tao, Y. Zhou, Y. Wang, H. Zhang, H. Wang, W. Zhang, Kadel: Knowledge-aware denoising learning for commit message generation, ACM Transactions on Software Engineering and Methodology (2024).
[145] Y. Huang, F. Xu, H. Zhou, X. Chen, X. Zhou, T. Wang, Towards exploring the code reuse from stack overflow during software development, in: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, 2022, pp. 548–559.
[146] M. Yang, Y. Zhou, B. Li, Y. Tang, On code reuse from stackoverflow: An exploratory study on jupyter notebook, arXiv preprint arXiv:2302.11732 (2023).
[147] C. Osborne, J. Ding, H. R. Kirk, The ai community building the future? a quantitative analysis of development activity on hugging face hub, Journal of Computational Social Science (2024) 1–39.
[148] M. White, I. Haddad, C. Osborne, X.-Y. L. Yanglet, A. Abdelmonsef, G. A. Commons, S. M. Varghese, The model openness framework: Promoting completeness and openness for reproducibility, transparency, and usability in artificial intelligence, arXiv preprint arXiv:2403.13784 (2024).
[149] E. Bogomolov, A. Eliseeva, T. Galimzyanov, E. Glukhov, A. Shapkin, M. Tigina, Y. Golubev, A. Kovrigin, A. van Deursen, M. Izadi, et al., Long code arena: a set of benchmarks for long-context code models, arXiv preprint arXiv:2406.11612 (2024).
[150] S. Dueñas, V. Cosentino, J. M. Gonzalez-Barahona, A. d. C. San Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, A. P. García-Plaza, Grimoirelab: A toolset for software development analytics, PeerJ Computer Science 7 (2021) e601.
[151] S. Dueñas, V. Cosentino, G. Robles, J. M. Gonzalez-Barahona, Perceval: software project data at your will, in: Proceedings of the 40th international conference on software engineering: companion proceeedings, 2018, pp. 1–4.
[152] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical software engineering 14 (2009) 131–164.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: