International Journal of Computer Applications (0975 - 8887)
*Volume * - No.*, February 2025*

Support Mechanisms for Conducting Empirical Research in Software Engineering: A Systematic Mapping Study

Abeer Alarainy
Imam Mohammad Ibn Saud Islamic University
Software Engineering

Nora Madi
King Saud University
Software Engineering

Arwa Abolkhair
King Saud University
Software Engineering

Aljawharah AlMuaythir
Alfaisal University
Software Engineering

Aljohara Alyousef
King Saud University
Software Engineering

Zainab Altamimi
Hail University
Software Engineering

Ohud Almeshari
King Saud University
Software Engineering

Muna Al-Razgan
King Saud University
Software Engineering

ABSTRACT

Empirical research plays a crucial role in the evolution of the field of software engineering (SE), as it provides evidence-based insights that help improve the software development life cycle (SDLC). Recognizing its significance, this paper presents a systematic mapping study (SMS), analyzing 195 studies published between 2019 and 2024, selected from three highly reputable sources: IEEE, ACM, and Wiley Digital Libraries. Understanding trends in the different applications of empirical strategies and their associated support mechanisms, as well as the challenges in this context, is a necessary step toward advancing the field of empirical software engineering. The findings of this SMS reveal that empirical techniques are heavily employed in the testing and maintenance phases of SE, which accounted for 42% of the analyzed studies, while the management and deployment phases receive comparatively less attention. Experimental methods emerged as the most commonly used empirical strategies, followed by quantitative and qualitative analyses. Tools and techniques were identified as the most frequently employed support mechanisms, demonstrating their importance in empirical research. This study offers a comprehensive resource for researchers and practitioners, mapping empirical strategies onto specific SDLC phases and illuminating the challenges and possibilities of designing and executing empirical SE research.

General Terms

Empirical Software Engineering, Systematic Mapping Study, Empirical Strategies, Support Mechanisms, Software Development Life Cycle

Keywords

Experiment, Survey, Case study, Tool, Technique, Testing, Maintenance

1. INTRODUCTION

Empirical research plays a pivotal role in the success of studies across diverse fields and domains. This is particularly true for research in the field of software engineering (SE), since most of this research is supported by empirical strategies and the related mechanisms. In fact, the significance of empirical methods for the success and validity of software engineering research outcomes is reflected by the emergence of an entire field under the name of empirical software engineering (ESE). The field of ESE is concerned with the application of empirical strategies throughout the software development life cycle (SDLC). Adopting suitable empirical strategies and supporting mechanisms is key to the development of successful software, whether they are implemented during the phase of analysis and requirement collection, design and architecture, development, testing, deployment, maintenance and evolution, or management. ESE provides practitioners with the opportunity to gain insights into user and application behavior, make evidence-based decisions, and identify problems and areas for improvement. Furthermore, by following the proper practices and guidelines during empirical research and exploiting the proper tools, many threats can be mitigated, and research efforts can be replicated. Multiple studies have recognized and emphasized the importance of carrying out empirical studies correctly and systematically to increase validity, enhance reliability, and allow for generalizability and replicability [1], [2], [3]. The SE community has witnessed increased interest in the field of ESE due to the latter's importance and diverse applications. It is expected that this interest will continue to grow as technologies evolve and software products become more complex and advanced. Following in the footsteps of previous research efforts [1], this paper addresses the reasons for conducting a systematic mapping study (SMS). The goal is to promote a comprehensive understanding of the prevalence of empirical strategies and the use of support mechanisms in SE. By searching the literature and analyzing selected journal studies published between 2019 and 2024,

this research offers a current and holistic view of the application of empirical strategies within the different phases of the SDLC, as well as the reported use of support mechanisms during the implementation of these strategies. Specifically, this study aims to do the following:

(1) Identify and classify support mechanisms used in empirical studies: This study seeks to identify the most commonly used support mechanisms on which researchers rely for conducting and supporting the activities of different empirical studies.
(2) Map empirical strategies onto their respective SDLC phases: By identifying trends in the application of empirical strategies during the different phases of the SDLC, this research aims to determine which SDLC phases have received more empirical attention and highlight potential gaps in the research accordingly.
(3) Shed light on commonly reported challenges faced when conducting empirical studies: By explicating the key challenges and limitations that researchers encounter when conducting empirical research, this study provides a tool to help future researchers devise strategies to overcome them while identifying areas in which support is required.
(4) Identify gaps in current research: Highlighting unexplored areas, or areas that are not supported by sufficient research, makes a significant contribution to the research community.
(5) Provide guidance for future research: By identifying gaps, highlighting challenges, and recognizing areas for improvement, this research provides a roadmap for researchers that can guide future research efforts.

Overall, with this SMS, the present research aims to provide a comprehensive classification of empirical strategies and support mechanisms, which are analyzed from an SE perspective, and to provide insights into how effective applications of empirical strategies can help overcome existing challenges.

2. RELATED WORK

Empirical SE methods have evolved over the years, and numerous studies have explored ESE research in terms of the applications and underlying methodologies [4]. For example, Wohlin and Aurum [5] introduced a research-decision-making structure for ESE scholarship. The structure consists of eight decision points, each representing a specific part of ESE research, and is designed to serve as a starting point for developing research designs. In addition, Stol and Fitzgerald [6] introduced the ABC framework, a structured taxonomy for SE research strategies based on generalizability and obtrusiveness. It defines eight research strategies, providing researchers with a clear understanding of appropriate alternatives for achieving diverse objectives. While these studies provide valuable guidance on selecting research designs and strategies, others delve deeper into examinations of the tools and mechanisms used to support the application of these strategies. Table 1 offers an overview of such studies, presenting details such as the publication year, focus of the study, and number of primary papers covered. For example, Melegati et al. [7] conducted an SMS of a set of studies to explore the application of qualitative surveys in SE research. They found that most of these studies were related to software maintenance and software testing. Interestingly, although the study covers a broader variety of empirical strategies, this finding is consistent with the results, which include the observation that most papers have focused on aspects of software maintenance and testing. The authors of [7] also analyzed the implementation of qualitative surveys, focusing on sampling, data collection, and analysis. Based on their investigation, they proposed a set of guidelines for method design—focusing on standards for defining study goals, planning, sampling, and collecting and analyzing data—to aid researchers in designing studies and provide reviewers and readers with a framework for analyzing and assessing these studies. This research focused on surveys, however, whereas the study extends its scope to include a broader range of empirical methods. On the other hand, Borges et al. [1] conducted an SMS to aggregate and classify supporting mechanisms used in ESE studies published in key scientific venues, including International Conference on Evaluation and Assessment in Software Engineering (EASE), Inernational Symposium on Empirical Software Engineering and Measurement (ESEM), and Empirical Software Engineering Journal (ESEJ). They categorized the empirical strategies employed, and their analysis led to the identification of 412 mechanisms across multiple studies, focusing on research methods. Their results suggest that the most commonly used support mechanisms are related to experiments, which aligns with the findings. They also determined that the most frequently reported empirical strategies are experiments and case studies. In addition, the authors created a list of support mechanisms that SE researchers can employ to plan and conduct empirical studies. While the study provides similar value to the SE community, the focus is on examining the application of empirical strategies specifically across the SDLC. This approach allows us to understand how these methods support the different phases of software development, offering tailored insights that can aid both researchers and practitioners in choosing the right strategies and most appropriate mechanisms for specific phases of the SDLC. Similarly, Zhang et al. [8] conducted an SMS to explore ESE and determined that the most frequently used empirical methods are experiments, case studies, and surveys. They also examined the use of empirical research methods across SE subfields. Their findings show that such methods are mostly applied during the phases of software maintenance, quality, and testing. Their findings offer insight into commonly used empirical approaches, data sources, and processing tools, while also revealing areas in which ESE methodologies can be improved. The present study focuses exclusively on support mechanisms related to data collection, processing, and analysis. Molléri et al. [2] offer a rich catalogue of aggregated and organized guidelines, assessment instruments, and knowledge organization systems that support empirical research in SE. First, their study provides a comprehensive list of research methods, highlighting interrelated methods and providing detailed references. Regarding research phases (i.e., planning, execution, analysis, and reporting), their findings suggest that while most resources cover more than one phase, only 15% address all phases fully, suggesting that researchers may need to combine multiple resources to bridge methodological gaps. Finally, their catalogue emphasizes the importance of evaluated instruments, though only 10% of the resources studied had undergone evaluation for accuracy and completeness. However, their study does not analyze the application of empirical strategies across the SDLC. In addition, Guevara-Vega et al. [3] conducted an SMS on empirical SE strategies, analyzing the studies and categorizing them based on the types of empirical strategies employed, the nature of the hypotheses addressed, and the key characteristics of the research inception. Their findings point to a predominance of three main strategies: controlled experiments, case studies, and surveys. The study emphasizes that the selection of an empirical strategy depends on the nature of the research, its scope, and the available resources. Moreover, the authors identified a gap in the definition of the characteristics of empirical methods in SE, noting that this presents a challenge for new researchers when selecting appropriate strategies.

This work advocates for the further standardization and automation of empirical methods; however, it does not review the supporting mechanisms used within different empirical strategies. Unlike these studies, the review provides a comprehensive analysis of empirical strategies and discusses a variety of support mechanisms, considering their application across each phase of the SDLC, and integrating sources from the past six years (2019–2024). the study not only highlights current practices but also identifies the challenges that researchers face when conducting empirical studies, addresses existing gaps, and suggests directions for future research.

3. METHODOLOGY

To achieve the objectives of this research, an SMS was conducted in accordance with the guidelines provided by Kitchenham and Petersen et al. [9], [10]. The data collected through this study were later analyzed in a structured fashion designed to reduce bias. The first step of conducting an SMS is the planning process. Figure 1 provides an overview of the mapping process followed in this study.

3.1 Planning Process

The planning process is an umbrella concept that encompasses multiple subprocesses, including the definition and evaluation of the review protocol, and the formulation of the research questions.

3.1.1 Review Protocol. A review protocol was created to guide the execution of the systematic mapping and reduce researcher bias. The protocol outlined inclusion/exclusion criteria, search strategy, and data extraction procedures to reduce researcher bias. Additionally, independent verification and consensus meetings were conducted to resolve disagreements during the study selection process. The process of creating the review protocol is illustrated in Figure 1 and includes the following stages:

(1) Formulating research questions.
(2) Developing a search plan.
(3) Selecting the study criteria and procedures.
(4) Evaluating quality criteria for the collected studies.
(5) Extracting relevant data.
(6) Synthesizing and analyzing the extracted data.

3.1.2 Research Questions. The main objective of this mapping study was to analyze information relevant to developing and conducting ESE studies throughout the different phases of the SDLC in order to gain a comprehensive overview of trends and challenges, as well as gaps and opportunities. To achieve the objective, four research questions were formulated (Table 2).
In addition to the formulation of the research questions, as Table 3 shows, the scope and objectives of this review are clarified by employing the PICO (population, intervention, comparison, outcomes) criteria as defined by Kitchenham and Charters [11].

3.2 Conducting the Study

Once the planning process has been completed, the process of conducting the actual research unfolds through its subprocesses:

(1) Constructing a search query.
(2) Selecting relevant studies based on the defined questions and exclusion and inclusion criteria.
(3) Conducting quality assessments of primary studies after a full-text screening at the end of the process.

3.2.1 The Search Query. The guidelines and instructions provided by Petersen et al. and Kitchenham and Charters were followed to construct a search query that is relevant to and reflective of the research topic and its objectives.[10], [11]. Following these guidelines, the process can be broken down into the following steps:

(1) Deriving main terms from the research questions.
(2) Identifying abbreviations, spelling variants, alternative words, and synonyms for the major terms.
(3) Checking for keywords in relevant research studies to verify the prior steps.
(4) Using the Boolean operators ”OR” and ”AND” to construct search strings. The ”OR” operator connects synonyms, alternative words, and abbreviations, while ”AND” connects the main terms.
(5) Merging the main terms to create the ultimate search term.

The final constructed query is as follows:
Empirical AND Software AND ((Analysis OR Support OR Method OR Discipline OR Methodology OR Literature Review OR Meta-analysis OR Experimental OR Experiment OR Qualitative OR Quantitative OR Statistical OR Study OR Survey OR Archival Data OR Grounded Theory OR Simulation OR Simulating OR Modeling OR Ethnography OR Phenomenology OR Observation OR Historical OR Research OR Focus Group OR Content Analysis OR Think Aloud) AND (Framework OR Model OR Guideline OR Lessons OR Paradigm OR Process OR technique OR Technical OR Template OR Checklist OR Tool OR GQM OR Goal-question-metric OR Route-map OR Computing OR Principle OR Handbook OR Instrument OR Mechanism) AND (Planning OR Requirement OR Analysis OR Design OR Architecture OR Implementation OR Development OR Testing OR Deployment OR Maintenance OR SDLC))

3.2.2 Research Resources. To address the research questions, a search was conducted by entering the predefined search term into three well-known electronic databases: IEEE Xplore Digital Library, ACM Digital Library, and Wiley. These databases were selected based on their reputation in SE research, coverage of high-quality peer-reviewed studies, and frequent use in systematic mapping studies. They collectively provide a broad and reliable dataset for answering the research questions.

3.2.3 Study Selection Criteria. The process of selecting primary papers for analysis involved several iterations. Initially, all papers containing the relevant search terms were retrieved from the electronic data sources. These papers then underwent an initial screening process that involved reviewing their titles and abstracts and then excluding any irrelevant papers. Subsequently, any duplicate studies were removed. The remaining papers underwent a full-text screening, and after further exclusions based on this screening, the final set of papers was selected. The data in this studies were exracted and analyzed, and the mapping results were identified.
Prior to the initial screening step, all of the studies yielded by the three databases were imported into Rayyan QCRI [12]. Rayyan's duplication detection feature was used to ensure that no duplicate studies were identified. As a result, a total of 336 studies remained. Subsequently, the papers were filtered by the seven researchers based on the titles and abstracts, and their relevance was evaluated based on predefined exclusion and inclusion criteria. To minimize the risk of bias, the ”blind mode” feature in Rayyan was activated. When this blinding feature is on, the notes, labels, and decisions

Table 1. Related Work.

Title	Year	Focus	Review Method	Period	Primary Paper	Ref.
Qualitative Surveys in Software Engineering Research: Definition, Critical Review, and Guidelines	2015	Exploring the use of qualitative surveys in ESE, identifying key challenges, and proposing guidelines to enhance study design.	SMS	1998-September 2021	66	[7]
Support Mechanisms to Conduct Empirical Studies in Software Engineering: A Systematic Mapping Study	2015	Identifying support mechanisms (methodologies, tools, guidelines, processes, etc.) and analyzing their applications.	SMS	1996-2013	891	[1]
Empirical Research in Software Engineering — A Literature Survey	2018	Identifying trends and commonly used empirical methods, data sources, and tools, with the goal of providing insights and recommendations for future research in ESE.	SMS	2013-2017	538	[8]
CERSE - Catalog for Empirical Research in Software Engineering: A Systematic Mapping Study	2018	Identifying guidelines, assessment instruments, and knowledge organization systems for conducting and evaluating ESE.	SLR	1991-2014	341	[4]
Empirical Strategies in Software Engineering Research: A Literature Survey	2021	Classifying publications that establish empirical strategies; providing an overview of approaches, types of research, types of hypotheses, and characteristics.	SMS	Up to 2019	80	[3]

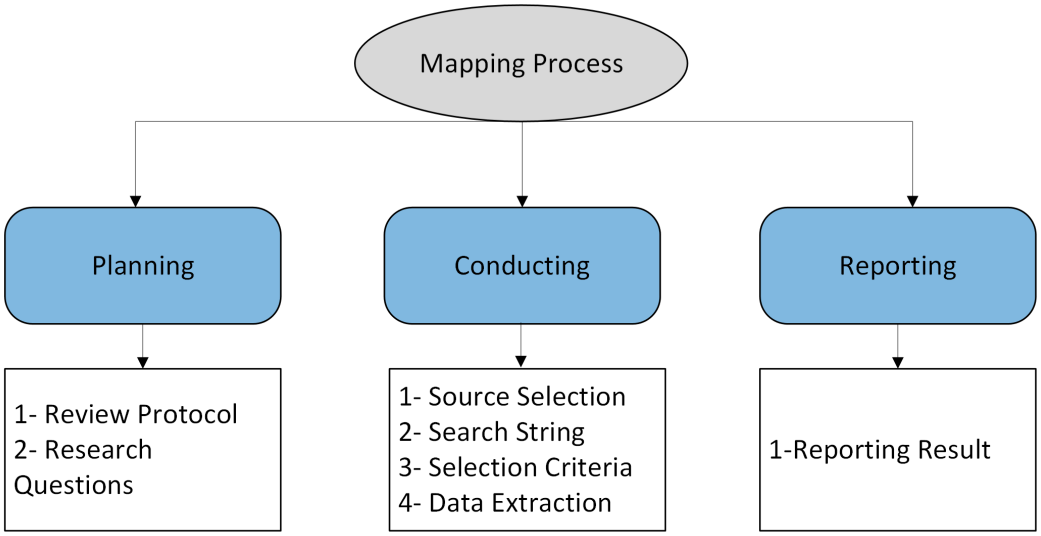

Fig. 1. The systematic mapping protocol process.

notes of each collaborator are not visible to the other collaborators. Each pair of researchers thoroughly examined the abstracts of 94 studies. When blind mode was turned off, conflicts arose over 109 studies, as unanimous decisions on inclusion or exclusion were not reached. Furthermore, 50 studies were unanimously selectied for exclusion by the assigned researchers. Conversely, 153 studies were agreed upon by the researchers to be included, and a total of 29 studies were categorized as ”maybe.” Each pair of researchers held

Table 2. Research Questions

No.	Question	Motivation
RQ1	What support mechanisms are used when conducting empirical studies?	To uncover the support mechanisms on which researchers rely to successfully conduct empirical studies in SE. Understanding these mechanisms helps identify best practices, technological advancements, and potential areas for improvement to support research.
RQ2	What are the most commonly used empirical strategies within each phase of the SDLC?	To identify which phases are the focus of empirical research. This helps illuminate the extent to which the entire SDLC is being studied and whether certain phases receive more research attention, thus revealing gaps and trends in the literature.
RQ3	What gaps are observed in research on ESE?	To reveal unexplored areas and opportunities for improving research practices in the field.
RQ4	What key challenges do researchers face when conducting empirical studies in software engineering?	To uncover the main challenges researchers encounter during empirical studies.

Table 3. Using the PICO criteria to define the scope and goal of SMS

Population	Articles about empirical studies in SE
Intervention	Using support mechanisms to conduct empirical studies
Comparison	Identifing the empirical strategies employed in each stage of the SDLC and the support mechanisms used to conduct empirical studies.
Outcomes	Classifying empirical strategies based on SDLC and support mechanisms, along with synthesized evidence, to guide research and practice.

Table 4. Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Studies based on empirical research in software engineering that describe empirical strategies and identify support mechanisms for conducting empirical studies throughout the software development life cycle (SDLC)	Studies not related to the research objective
Studies published within the last six years (between 2019 and 2024)	Papers published before 2019
Peer-reviewed studies	Non-English studies
Journal papers only	Technical reports, government reports, letters and editorials, short notes, abstract-only studies
Primary studies only	Duplicate studies were removed

sessions to resolve their conflicting opinions, and decisions were made about whether to include or exclude the studies of concern. Once the researchers reached an agreement, an immediate decision was made, and the studies were moved to either the ”included” or ”excluded” category.

3.2.4 Inclusion and Exclusion Criteria. Inclusion and exclusion criteria were defined in order to narrow down the most relevant studies. Table 4 provides an overview of the inclusion and exclusion criteria.

3.2.5 Quality Assessment Criteria. To ensure the relevance of the selected research articles and minimize selection bias, criteria for quality assessment were established, and each study was evaluated against them. Studies that met these criteria were chosen for further

Table 5. Quality Assessment Criteria.

Criteria No.	Quality Criteria	Score
QC 1	Are the support mechanisms clearly defined?	Y(1)-N(0)-P(0.5)
QC 2	Are the empirical strategies clearly defined?	Y(2)-N(0)-P(1)
QC 3	Is the SDLC stage explicitly defined?	Y(1)-N(0)-P(0.5)

data analysis; otherwise, the papers were excluded.In this systematic mapping, QC2 was identified as the most significant quality assessment criterion, and a double weight of 2 was assigned to it. QC1 and QC3 were each assigned a weight of 1 point if fully answered, 0.5 points if partially answered, and 0 points if not answered. Meawhile, QC2 was assigned 2 points if fully answered, 1 point if partially answered, and 0 points if not answered. Each study required a combined score of 3 out of 4 to be accepted. The quality assessment criteria are presented in Table 5.

3.2.6 Data Extraction. A total of 336 studies was initially selected. Following a detailed review of the full texts, the number was narrowed down to 231 for further analysis. Quality assessment criteria were then applied, resulting in the exclusion of 36 studies. In the end, 195 papers were considered suitable for data extraction. Figure 2 illustrates the complete process of searching for and selecting the primary studies. Forms were created to serve as templates for extracting specific data items from the selected studies. After satisfying the inclusion/exclusion criteria and passing the quality assessment process, the studies advanced to the phase at which the defined data extraction items were gathered to map the results. The data extraction process consisted of two steps. First, all relevant data were extracted based on the procedure presented

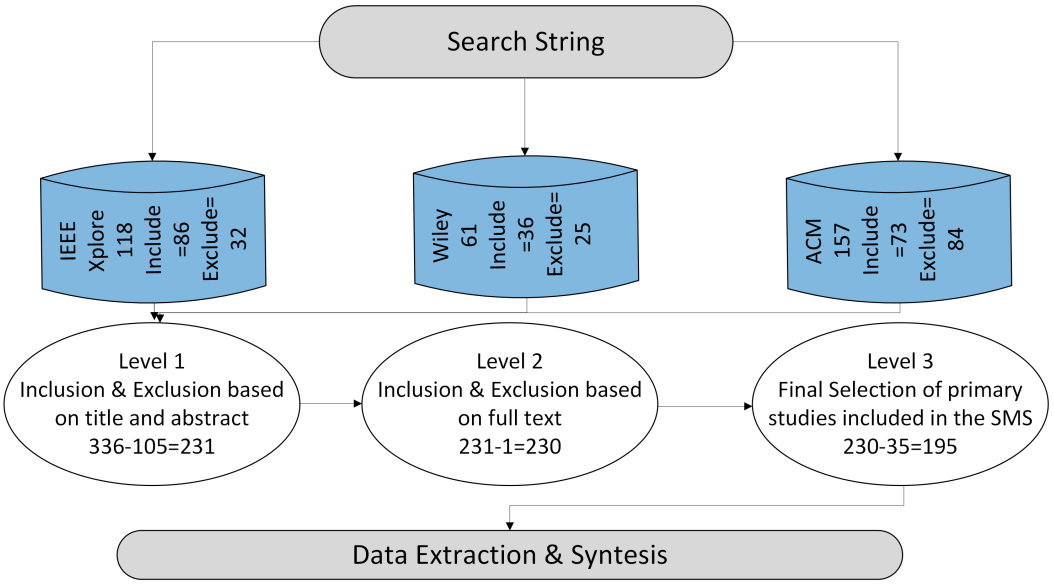

Fig. 2. The systematic mapping protocol process.

Table 6. Data Extraction.

ID	Data Item	Description	Relevant RQ
ID1	Title of paper	What is the article's title?	Manuscript information
ID2	Year of publication	Year the study was published	Manuscript information
ID3	Digital library	Name of repository (e.g, IEEE, ACM....)	Manuscript information
ID4	Types of empirical strategies	Type of empirical strategy (e.g. experiment, survey....)	RQ1, RQ2
ID5	Types of support mechanisms	Type of mechanism (e.g. framework, model, tool....)	RQ1
ID6	SDLC stage	SDLC phase(s) in which empirical strategies were applied (e.g., testing, design)	RQ2
ID7	Challenges identified	Challenges observed in empirical studies	RQ3

in Table 6. Two mapping forms, presented in Table 7 and Table 8, were then used to help answer the research questions.

3.2.7 Data Synthesis. In the data synthesis process, extracted data are transformed into valuable information that addresses the research questions. To facilitate the analysis, a structured format was created, and all extracted data were classified into three categories: one based on the SDLC stage, the second on the type of empirical strategy employed, and the third on the support mechanism used.

4. RESULTS

This section is focused on presenting the findings of the systematic mapping. General information about research trends over the chosen six-year period is provided in Section A, including the distribution of studies across the selected sources and years (Figure 4). Section B addresses (RQ1) by detailing the findings regarding the support mechanisms used in empirical SE studies. Section C answers (RQ2), focusing on the correlation between the empirical studies and the phases of the SDLC. Section D responds to (RQ3) by highlighting and addressing existing gaps in ESE research. Finally, Section E discusses the challenges commonly faced when applying empirical strategies, thus addressing RQ4.

4.1 General Information

Figure 3 shows the distribution of 195 primary studies drawn from three data sources: IEEE, ACM, and Wiley. IEEE leads with 86 studies (44%), highlighting its prominence in engineering and technology research. ACM follows with about 37% of the studies, emphasizing its importance in computer science and software engineering. In contrast, Wiley contributed only 36 studies (19%), indicating a more specialized focus on empirical research. IEEE and ACM together account for 82% of the studies, reinforcing their status as key sources for empirical research, while Wiley's smaller share suggests a less targeted role in this field. Figure 4 illustrates the primary studies, which are grouped by publication date. From 2019 to 2020, the number of studies on this topic remained consistent, with 28 published in both years, indicating a moderate level of interest. In 2021, there was a slight increase to 30 studies, followed by a decrease to 26 in 2022, which may reflect a temporary shift in focus. However, in 2023, there was a significant rise to 47 studies,

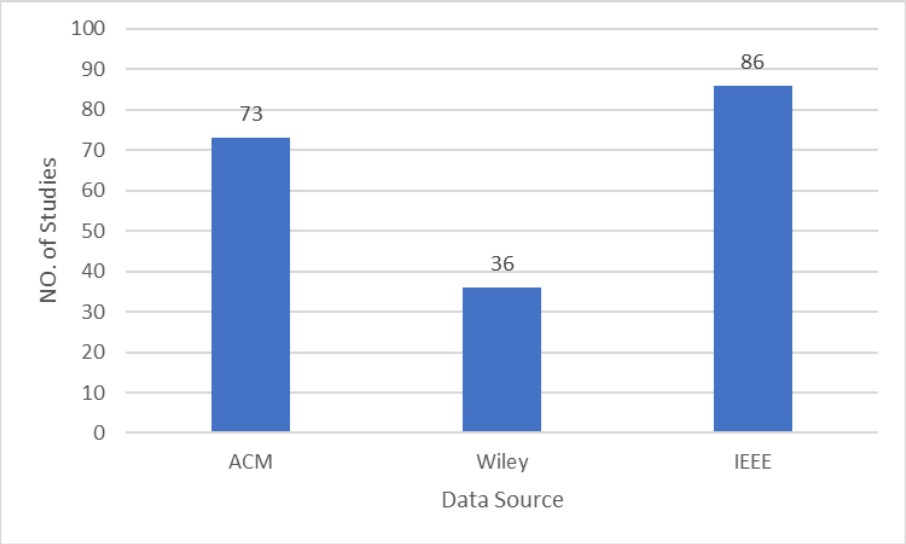

Fig. 3. No. of Studies per Resource.

the highest number recorded in this six-year period. This increase suggests a growing interest, likely driven by advancements in tools and techniques. In 2024, the number of studies decreased to 36, remaining above the levels seen from 2019 to 2022, indicating sustained interest but stabilization after the peak in 2023. Overall, the data reveal an upward trend in research, particularly over the last two years.

4.2 Mechanisms of Support for Empirical Studies in SE

This subsection addresses the first research question, "What support mechanisms are used to conduct empirical studies?" First, the types of empirical strategies employed in the studies and their frequency of use are presented. The various support mechanisms employed to carry out successful empirical strategies are laid out and discussed. Finally, the support mechanisms observed to be the most beneficial for conducting each type of empirical research are explored. A total of 409 empirical strategies were extracted, and Figure 5 below shows the distribution of the elicited reports. After analyzing the extracted data relative to the reported use frequency of the various empirical strategies in research, it was observed (see Figure 5) that the most commonly adopted empirical strategy is the experiment, which comprised 28% of instances. The "experiment" category includes experiments, controlled experiments, and quasi-experiments, which Wholin et al. [13] consider synonyms denoting the same concept. The results indicate that the second-most frequently used strategy is quantitative analysis, accounting for almost 22% of the total reported strategies, which suggests a strong emphasis on approaches that are controlled, objective, and statistically valid. Qualitative analyses, surveys, and case studies came next and were also widely employed, reflecting the importance of detailed and contextual exploration. Meanwhile, methods such as modeling, simulation, observation, and interviews exhibited moderate usage, suggesting their relevance and applicability within specific contexts of research. The less frequently used strategies include grounded theory, focus groups, action research, field studies, meta-analyses, and ethnography. These strategies may represent specialized applications or underutilized opportunities for exploration. The data demonstrate a balance between the use of quantitative methods (such as experiments and quantitative analyses) and qualitative methods (including qualitative analyses and case studies). This illustrates these methods' broad applicability to empirical research while highlighting the potential for greater use of less commonly employed strategies. Many studies combine two or more methods to strengthen and support their findings. However, the present research focuses on identifying the support mechanisms used to conduct empirical studies rather than on methods for integrating or combining multiple strategies. As Figure 6 shows, a wide use of practical support mechanisms are used in empirical research. The results suggest that techniques and tools are the most commonly used mechanisms, accounting for approximately 48% of reported instances. This demonstrates their broad applicability and effectiveness in supporting research activities. Following closely behind are instruments, models, guidelines, and frameworks, which together contribute to 42%. These mechanisms provide practical methodologies for conducting data collection, analysis, and processing. In contrast, less frequently used mechanisms—such as the goal-question-metric (GQM) method, processes, principles, and checklists—serve more specialized or supplementary needs. The lower frequencies of occurrence for these mechanisms may indicate that they have limited use or are applicable only in specific contexts or scenarios. Another reason might be that many studies use certain support mechanisms but do not explicitly mention them. Overall, the data emphasize a strong preference for methods that deliver immediate practical value, while more abstract or specialized mechanisms are less preferred. Information on exploited mechanisms was extracted individually, resulting in a total of 388 mechanisms and 409 empirical strategies. Following this, the researchers held a meeting to negotiate the classification of these mechanisms and came to the decision to recategorize them into three general categories as follows:

(1) **Methodological Frameworks and Models:** This category encompasses comprehensive structures that provide theoretical or methodological foundations for conducting empirical re-

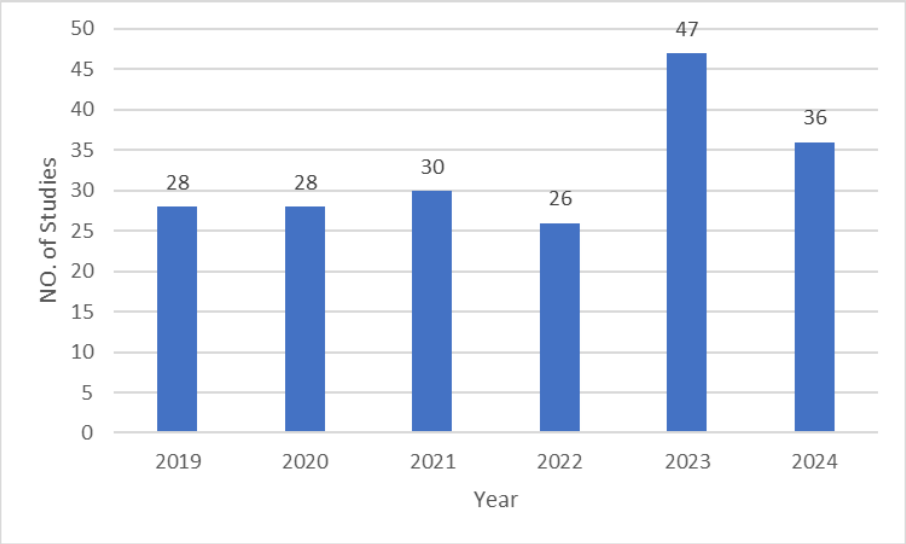

Fig. 4. No. of Studies per Year.

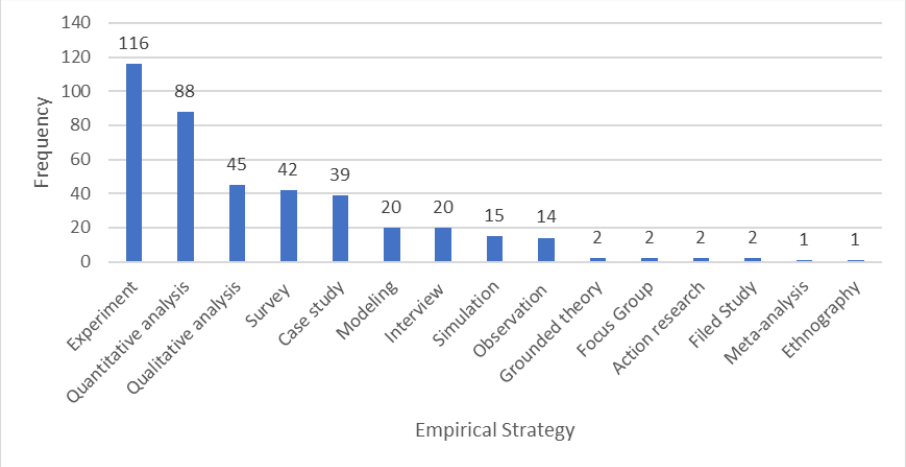

Fig. 5. No. of Studies per Year.

search. Items in this category include frameworks, models, paradigms, GQM, route maps, processes, and principles.

(2) **Guidance and Best Practices:** This category encompasses items that offer practical advice, step-by-step instructions, or best practices for researchers. It includes guidelines, lessons, templates, checklists, handbooks, standards, benchmarks, forms, and protocols.

(3) **Tools and Techniques:** This category covers supporting software, technical tools, techniques, mechanisms, instruments, and computing tools.

To achieve a more comprehensive classification and minimize subjective decisions, various elements were excluded from the investigation, including sampling techniques, statistical tests and methods, communication strategies, social media channels, machine learning techniques, datasets, and subjects for case studies. To reduce ambiguity, the strategies were reclassified into categories such as experiments, surveys, case studies, interviews, qualitative analyses, and quantitative analyses. In the beginning, only the research types explicitly mentioned in the repository were extracted. However, it was noticed that not all of them were mentioned. As a result, each paper was revisited in depth to extract this information. This led to a reduction in the number of papers in the repository to 354 and the number of techniques to 324. Figure 7 illustrates the distribution of the empirical strategies used in the primary studies. Experiments are the most frequently employed method, accounting for 168 (47%) of the studies. This indicates a strong preference for experimental approaches among all the results, likely due to their ability to provide controlled results. Following this, it was found that surveys make up 51 (14%) of the methods used, highlighting their importance in gathering a diverse range of data. Case studies, at 25 (7%), also play a significant role, as they offer detailed insights into specific contexts. In contrast, interviews were used less often, appearing in 17 (5%) of the instances. This may suggest that interviews are either less applica-

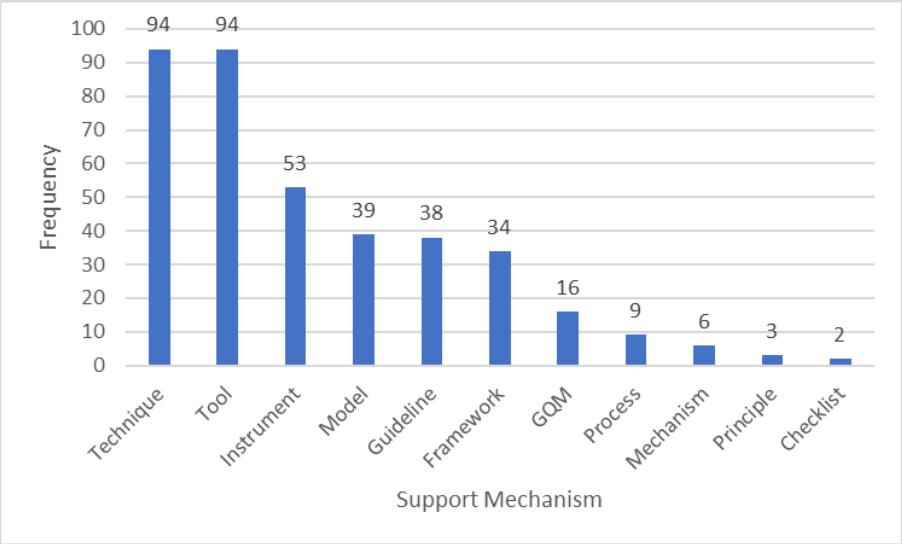

Fig. 6. No. of Studies per Year.

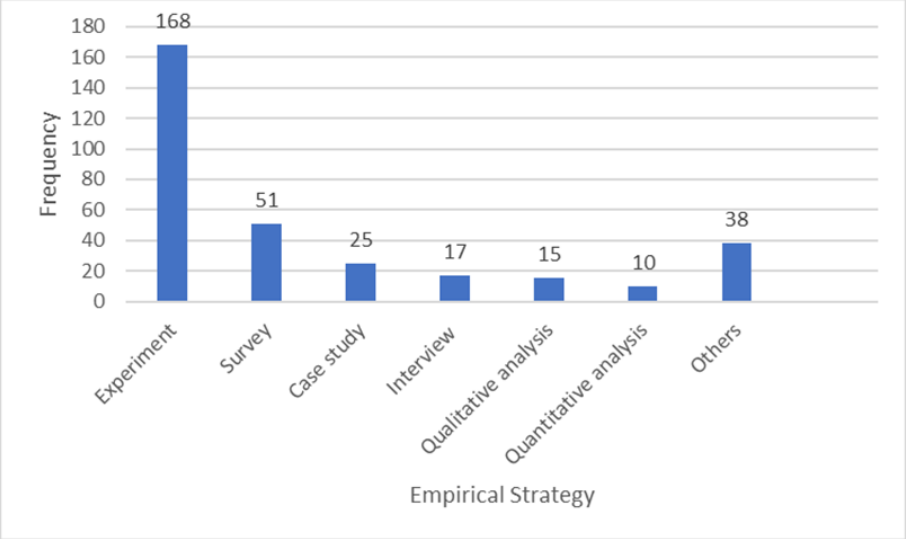

Fig. 7. No. of Studies per Year.

ble or are primarily employed to support findings reached using other strategies. The use of qualitative analysis was reported in 15 (4%), while quantitative analysis appeared in 10 (3% of) instances. Although it was originally expected that quantitative analysis would be reported more than qualitative analysis, given the dominant use of experiments in empirical studies, it was later determined that this outcome was probably the result of the lack of explicit identification of quantitative and qualitative strategies by researchers in the primary studies. Meanwhile, the prominence of experiments underlines the focus on systematic empirical validation in primary studies. Meanwhile, the relatively lower frequency of qualitative and quantitative analyses might suggest an opportunity to further implement these methods to foster a more comprehensive understanding of research phenomena. The ”Others” category includes meta-analysis, action research, think-aloud methods, simulation, observation, grounded theory, focus groups, ethnography, and modeling. The most frequent methods were observations, simulations, and action research, respectively, while ethnography was the least common approach. Figure 8 illustrates the frequency of use of the various support mechanisms used in the primary studies, highlighting the varying degrees of reliance on methodological frameworks, guidance practices, and tools. Among the three aforementioned categories, tools and techniques were the most frequently utilized support mechanisms, with a total of 173 (53%) of the reported instances. This dominance underscores their critical role in facilitating practical implementation and problem-solving in research, likely due to their direct applicability across various domains. Within this category, tools constituted (53%) of the total percentage, techniques accounted for (19%), and instruments made up (22%), with most instruments being either questionnaires or Lik-

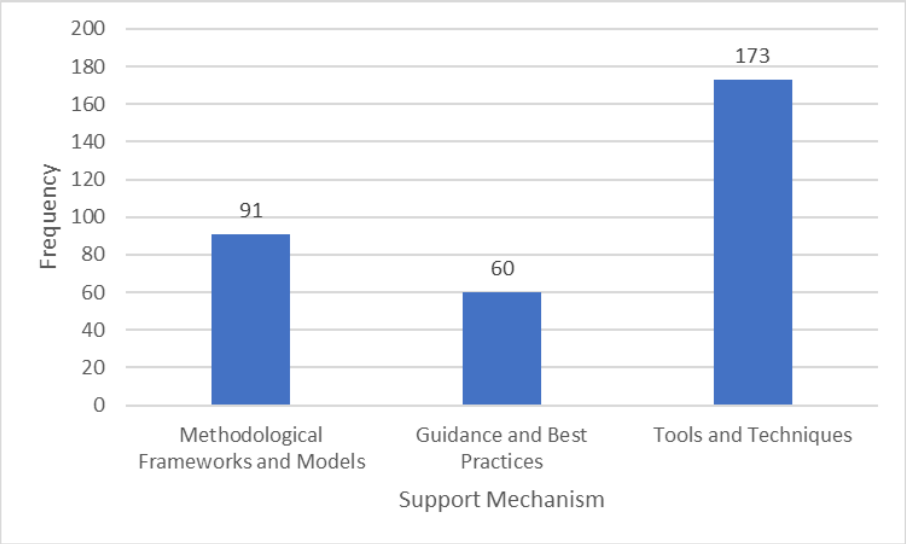

Fig. 8. Frequency of Support Mechanisms.

ert scales. Software constituted (6%). In contrast, methodological frameworks and models were employed in 91 (28%) of the studies, highlighting their significance in providing structured and theoretical foundations for research. These frameworks serve as essential guides for shaping study designs and ensuring rigorous methodological standards, although they are less prominent than tools and techniques. The most commonly used were frameworks and models, which accounted for (46%). In contrast, the least used were GQM, principles, and metrics, making up about (24%). Moderately used were methods and approaches, which constituted (30%). The least frequently observed category was "Guidance and Best Practices," which appeared in 60 (19% of) cases. While this support mechanism offers valuable recommendations and standards, its lower frequency suggests that it may be considered supplementary or context-dependent compared to the other mechanisms. This could also indicate a preference for more tangible and actionable support mechanisms, such as tools, over general guidelines. Notably, guidelines made up (65%) of this category, while the least-used resources—benchmarks, templates, and recommendations—occurred only two to three times each. Table 7 highlights the clear preference for tools and techniques in the primary studies, emphasizing a focus on actionable and results-oriented support. It further indicates that theoretical and guidance-oriented mechanisms, while still important, serve a more complementary role in aiding research efforts.

Table 7 provides a detailed breakdown of the support mechanisms discovered alongside the relevant empirical strategies. Experiments exhibited the most diverse and intensive use of support mechanisms, with tools and techniques being the most prominent in this strategy. This strategy is further enhanced by the use of methodological frameworks and models. The integration of guidance and best practices is also evident. This variety highlights the complex nature of experiments, which require robust frameworks and diverse tools to ensure precision and repeatability. In contrast, quantitative analyses exhibited minimal use of methodological frameworks, with only one such framework reported. However, tools and techniques are essential and serve as the primary support mechanisms. This dependency underscores the analytical and computational demands of quantitative studies. Qualitative analyses demonstrated a balanced use of support mechanisms. The limited presence of methodological frameworks indicates a focus on practical applicability rather than on theoretical foundations. Surveys depend heavily on the various tools and techniques used, particularly instruments. Guidelines are integrated, and a balanced use of tools and techniques alongside guidance and best practices underlines the necessity of both theoretical grounding and practical tools. In summary, the data reveal a strong preference for tools and techniques across most empirical strategies, particularly in experiments and surveys. Methodological frameworks are prominent with strategies such as experiments and case studies, where they provide a solid theoretical foundation. While guidance and best practices are less dominant, they offer critical support in surveys and experiments. This distribution reflects the varying requirements of empirical strategies and highlights the complementary roles of these support mechanisms in primary studies.

4.3 Empirical strategies within SDLC phases

The distribution of empirical applications across different phases of the SDLC reveals significant variation in both research focuses and preferred methodologies. Figure 9 illustrates the number of instances documented for each SDLC phase. The testing and maintenance phases are particularly prominent, showcasing a strong emphasis on experimental and quantitative methodologies. In contrast, the earlier phases, such as Planning/Analysis and Requirements, along with the Management phase, receive minimal research attention, indicating critical gaps. The results highlight a clear research focus on the later SDLC phases. For instance, testing accounts for 25.7% of the total results (157 instances), while maintenance comprises 21.3% (130 instances). In comparison, the early phases, such as planning/analysis, with only 39 contributions (6.4%), and requirements, with 50 contributions (8.2%), fall significantly behind, as shown in Figure 9. These early phases primarily employ qualitative strategies, such as interviews and case studies, with a lack of robust methodologies such as experiments. The management phase is particularly underexplored, as represented by the shortest bar, accounting for just 2% of the results (12 instances). Overall, the

Table 7. Support Mechanism Mapping with Empirical Strategies.

Empirical Strategy	Methodological Frameworks and Models	Frequency	Guidance and Best Practices	Frequency	Tools and Techniques	Frequency
Interview	GQM	1	Template	4	Technique	2
			Guideline	2	Tool	3
			Practice	3	Instrument	1
			Protocol	1		
Experiment	Model	14	Benchmark	2	Tool	60
	Framework	17	Guideline	11	Software	9
	GQM	5	Standard	2	Instrument	10
	Metric	3	Checklist	1	Technique	13
	Method	6	Recommendation	3		
	Approach	9	Technique	1		
			Lesson	2		
Quantitative analysis	Framework	1			Tool	6
					Technique	3
Qualitative analysis	Approach	1	Guideline	4	Tool	1
	Method	1	Standard	1	Technique	7
Survey	Model	1	Template	1	Instrument	20
	Principle	5	Guideline	16	Tool	2
	Method	3			Technique	1
	Approach	1				
	Framework	1				
Case study	Model	3	Standard	1	Software	1
	GQM	1			Tool	10
	Approach	2			Instrument	2
	Framework	2				
	Method	2				
	Guideline	1				
Other	Principle	1	Guideline	3	Software	2
	GQM	2	Best Practice	1	Tool	11
	Model	3	Model	1	Instrument	4
	Approach	3			Technique	5
	Framework	1				
	Method	1				

results indicate a substantial focus on validation and maintenance while highlighting gaps and opportunities for further exploration in the early SDLC phases and in managerial aspects.

4.4 Gaps observed in ESE research

An analysis of the empirical strategies applied in the research revealed significant methodological gaps. The experiment was the most frequently used approach, with 168 occurrences, as shown in both Table 8 and Figure 10. In contrast, qualitative methods, such as case studies and interviews, were used much less frequently, with only 25 and 17 occurrences, respectively. Additionally, with 10 and 51 occurrences, respectively, quantitative analyses and surveys remained moderately utilized, indicating room for broader adoption. Figure 10 illustrates the distribution of empirical strategies across the three support mechanisms: methodological frameworks and models, guidance and best practices, and tools and techniques. The size of the bubbles in the chart reflects the frequency of usage. The figure highlights the predominant role of tools and techniques, particularly in experiments, which peaked at 93 occurrences. Surveys and other strategies also demonstrated a significant reliance on tools, with 23 and 22 occurrences, respectively. While moderately common overall, methodological frameworks and models were notably associated with experiments, accounting for 53 occurrences. They demonstrated consistent but lower usage in surveys, case studies, and other strategies, each with 11 occurrences. Guidance and best practices appeared to be secondary support mechanisms, mainly used in experiments (22 occurrences) and surveys (17 occurrences), while their application in qualitative strategies remained minimal or nonexistent. Figure 10 further indicates that specific empirical strategies, such as interviews and quantitative analyses, were scarcely represented across all three support mechanisms, suggesting limited application or possibly unreported data in these areas. Experiments emerged as the most resource-intensive strategy, relying heavily on both tools and frameworks. In contrast, surveys and other strategies exhibited a more balanced reliance on the three support mechanisms, although tools still dominated. This analysis underscores the central role of tools in empirical research, especially experiments, while also highlighting the varied contributions of frameworks and best practices, which vary according to the empirical strategy used.

The SDLC phase trends exhibited a similar disparity, with testing (157 contributions, 25.7%) and maintenance (130 contributions, 21.3%) receiving the most attention, again emphasizing the centrality of the later stages. The early SDLC phases, such as planning/analysis (39 contributions, 6.4%) and requirements (50 contributions, 8.2%), are significantly underrepresented, as is management (12 contributions, 2%), indicating a lack of focus on foundational and managerial aspects. The bubble chart further highlights

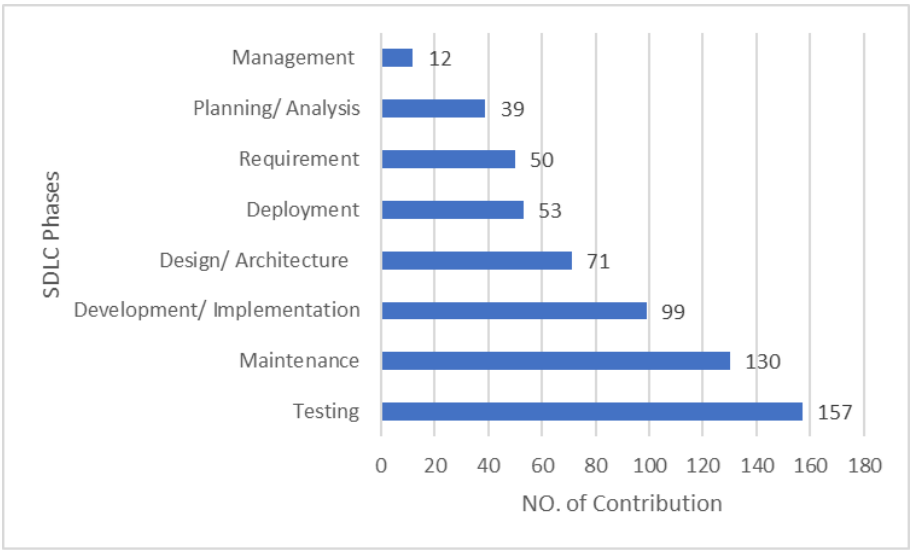

Fig. 9. No. of Studies per SDLC.

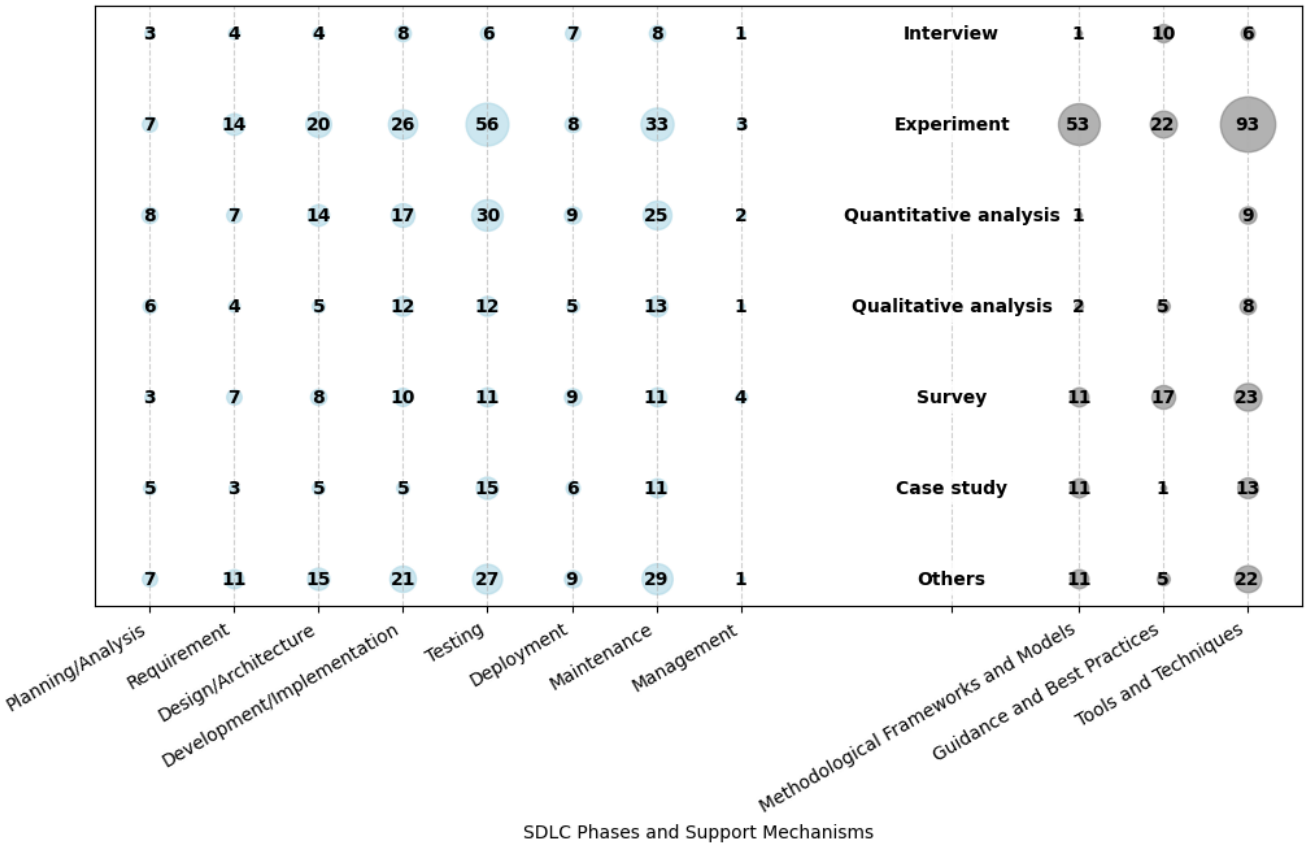

Fig. 10. Empirical Strategies Across SDLC Phases and Support Mechanisms.

the need for more guidance and best practices in empirical strategies, with most contributions concentrated in experiments and technical tools, leaving gaps in the use of practical frameworks and theoretical models to guide research in underrepresented SDLC phases. This uneven distribution presents opportunities to balance

Table 8. Mapping of Support Mechanisms with Empirical Strategies.

Empirical Strategy	Methodological Frameworks and Models	Guidance and Best Practices	Tools and Techniques
Interview	1	10	6
Experiment	53	22	93
Quantitative analysis	1	0	9
Qualitative analysis	2	5	8
Survey	11	17	23
Case study	11	1	13
Others	11	5	22

the methodological and phase-wise focus, which could address critical gaps in qualitative strategies and earlier SDLC stages.

4.5 Key Challenges

In this section, the challenges faced by researchers in the primary studies are presented. These challenges can impact the reliability, validity, and generalizability of research findings. To address existing obstacles affecting ESE research and to highlight their impact on future research directions, the challenges are categorized under the following labels:

(1) **Generalizability:** A frequent concern in ESE research is the limited generalizability of findings reached using specific datasets, contexts, or projects studied. This challenge arises due to several factors:
- —**Small Sample Sizes:** Many studies point out that having access to a limited number of participants or software projects makes it difficult to generalize conclusions to broader contexts.
- —**Focus on Specific Technologies:** A focus on specific programming languages, frameworks, or platforms is exhibited in many research studies. Focusing research on specific contexts limits the application of findings to other technologies or contexts.
- —**Reliance on Open-Source Data:** Open-sourced projects are considered valuable data sources; however, such projects do not fully represent the characteristics of closed-source or commercial software development. This difference affects the applicability of research findings in real-life industrial settings.

(2) **Data Quality and Noise:** Several difficulties that affect the identification of high-quality and relevant data for use as analysis subjects were identified.
- —**Data Acquisition Difficulties:** Obtaining real-world data is often difficult due to industry-specific confidentiality issues and restrictions on sharing proprietary information.
- —**Incomplete and Inaccurate Data:** Some datasets are found to be incomplete, inconsistent, or erroneous, forcing researchers to clean the data. Proper and thorough data cleaning requires devoted effort to facilitate meaningful analysis.
- —**Noise and Irrelevant Information:** Irrelevant data points can introduce noise, obscuring meaningful patterns and potentially leading to skewed or inaccurate results.

(3) **Bias and Subjectivity:** It is crucial to maintain the objectivity and impartiality of research findings. The reviewed studies highlight several potential sources of bias that should be carefully considered by researchers carrying out empirical studies:
- —**Human Assessment and Interpretation:** The researcher's perspective and interpretations can affect many research activities, such as manual data labeling, software artifact categorization, or qualitative analysis.
- —**Self-Reported Data:** Potential biases in participant replies must be considered in studies that use surveys, interviews, or other strategies reliant upon self-reporting. This form of bias is called respondent bias, and it occurs when participants respond inaccurately or falsely to proposed questions or scenarios. Such bias can compromise the validity of self-reported data.
- —**Selection Bias:** When choosing research participants or software projects, bias may be unintentionally introduced. For instance, the results may not be applicable to developers who are new to the industry if the research solely includes highly experienced developers.

(4) **Tool Usability and Limitations:**Tools used in analyzing software development data frequently have limitations that impact the efficiency and accuracy of research findings:
- —**Manual Data Extraction and Processing:** Due to the limits of current tools or the absence of suitable solutions, several studies exhibited a significant reliance on manual data extraction and processing. This dependence on manual efforts can significantly impede the scale and efficiency of research, particularly when handling large datasets.
- —**Limited Tool Functionality:** Studies mentioned a lack of support and the limited functionality of existing tools for specific tasks as an obstacle that can restrict the scope of research.

(5) **Participant-Related Challenges:** Several studies addressed the challenges caused by recruiting and managing participants in software engineering research.
- —**Recruitment Difficulties:** Finding and recruiting participants with the required experience can be a significant challenge. To overcome these difficulties, researchers may need to employ creative recruitment strategies, collaborate with industry partners, or consider alternative approaches, such as student participants, when accessing experienced practitioners is difficult.
- —**Participant Fatigue and Cognitive Load:** Some studies discussed the importance of managing participants' fatigue and cognitive load, particularly in studies involving long tasks or complex experiments.

(1) **Computational Costs and Complexity:** The computational demand of analyzing software development data is increasing, especially as datasets grow in size and complexity.
- —**Data Processing and Analysis:** Several studies acknowledged the challenges of processing large-scale datasets. This underscores the importance of scalable data management systems and efficient algorithms for handling and analyzing large datasets.
- —**Model Training Overhead:** The research sheds light on the computational costs of developing and training complex models, such as machine learning models. Meeting these computational demands requires the employment of the proper computing infrastructure and smart strategies to make the best use of the available resources.

(7) **Methodological Limitations:** There are certain inherent constraints or weaknesses in research designs or execution that fall

under the label of methodological limitations and can impact the validity and reliability of findings:

—**Threats to Validity** Most of the empirical studies reported facing a range of validity threats, which can affect construct, internal, and external validity. In the context of empirical research, these threats can be mitigated by applying solid and valid study designs, selecting appropriate data collection and data analysis methods, and possessing a strong background in methodological limitations.

—**Reliance on Specific Metrics:** Certain limitations may be introduced by the measures chosen to assess study findings.

As Figure 11 shows, among the 195 studies reviewed, the most commonly reported challenge was generalizability, followed by methodological limitations and then data quality and noise. Bias and subjectivity came after, followed by computational cost and complexity, participant-related challenges, and finally, tool usability and limitations. Figure 12 summarizes all the challenges. The adduced research problems underscore the complexity of conducting empirical studies in the context of software engineering. In order to overcome these obstacles and limit their effects, strong methodologies, close attention to potential biases, and constant effort must be put toward improving support mechanisms and thus allowing a wider application of study findings.

5. THREATS TO VALIDITY

To identify the threats to the validity of the SMS, the four basic types of validity threats outlined by Wohlin et al. are drawn upon. [13]. Each of these threats is discussed in the following subsections.

—**Conclusion validity** refers to issues that may arise when drawing conclusions and determines whether the SMS can be replicated. According to Wohlin et al. [13], the primary point of conclusion validity is to ensure that accurate conclusions are drawn regarding the relationships between a study's design and its outcomes. In this SMS, the most prominent type of validity threats was conclusion validity threats. Threats to conclusion validity include subjective measures, such as the manual categorization of empirical strategies and support mechanisms. To mitigate the impact of these subjective measures, each paper in the resulting set was classified by the researchers based on the classification introduced by Borges et al. [1] with changes based on the results (Table 7). Additionally, a general classification scheme was developed to accommodate all support mechanisms targeted by this study. The classifications were discussed among the team members, and any discrepancies that arose were carefully reviewed before the papers were reclassified. Furthermore, as in many studies conducted by a team of multiple researchers, the accuracy of the reported results can be influenced by the threat of researcher bias, which stems from the impact of individual perceptions or interpretations. This bias could also emerge from ambiguous reporting, implicit identification, and varying classifications of mechanisms and their applications within the primary studies. To reduce this risk, frequent discussions were held to pin down any areas of concern or confusion. However, even with all these measures taking place to eliminate the effect of conclusion validity threats, a margin of inaccuracy in the results remains inevitable due to this threat.

—**Internal Validity** involves factors that can indicate a causal relationship, including hidden variables, a phenomenon often referred to as spurious correlation. Thorough searches were conducted to identify all relevant studies from the three digital libraries. However, it is possible that some important papers may have been missed. It is believed that if any studies are missing, their number is likely minimal, and a variety of alternative keywords were used in the search string to help retrieve all pertinent research.

—**Construct Validity** According to Wohlin et al. [12], threats to construct validity pertain to issues that may arise during the research design phase. For instance, the research questions may not be fully comprehensive and might not cover every aspect of the empirical studies. To address this concern, brainstorming sessions were carried out to help effectively identify a set of research questions that appropriately reflect the current research landscape in this study.

—**External Validity** refers to the ability to generalize study results beyond the specific context of the research. In the SMS, completeness was aimed for within the defined scope, including keywords and timeframes. However, it can be acknowledged that no extensive literature review can claim to be entirely comprehensive. The SMS focuses mainly on scientific research involving empirical strategies and should not be generalized to closely related fields of research.

6. CONCLUSION

This study provides a comprehensive mapping of empirical strategies and their associated support mechanisms across the different phases of the SDLC, focusing on the analysis of data extracted from 195 studies published between 2019 and 2024. The findings reveal a strong focus on the testing and maintenance phases, with experiments emerging as the most widely employed empirical strategy. On the other hand, earlier phases, such as planning and requirements, as well as software project management aspects, remain underexplored. This reveals a notable and significant research gap that needs to be addressed by future scholars. Additionally, qualitative strategies, while essential in the initial phases of the SDLC, proved to be underutilized. The results of this study further underscore the significance of diversifying empirical methodologies and shed light on their adaptability within different contexts of research and in accordance with the unique challenges that result from each SDLC phase. By identifying these gaps and trends, the study lays a foundation and acts as a road map for future research efforts. It is strongly believed that this contribution enhances empirical approaches and facilitates their correct application across broader contexts within software engineering. Furthermore, this work serves as a valuable resource for researchers and practitioners seeking to improve the methodological consistency, accuracy, and relevance of empirical studies in software engineering.

7. REFERENCES

[1] A. Borges, W. Ferreira, E. Barreiros, A. Almeida, L. Fonseca, E. Teixeira, D. Silva, A. Alencar, and S. Soares, "Support mechanisms to conduct empirical studies in software engineering," pp. 1–4, 2014.

[2] J. S. Molléri, K. Petersen, and E. Mendes, "Cerse-catalog for empirical research in software engineering: A systematic mapping study," *Information and Software Technology*, vol. 105, pp. 117–149, 2019.

[3] C. Guevara-Vega, B. Bernárdez, A. Durán, A. Quina-Mera, M. Cruz, and A. Ruiz-Cortés, "Empirical strategies in software engineering research: A literature survey," pp. 120–127, 2021.

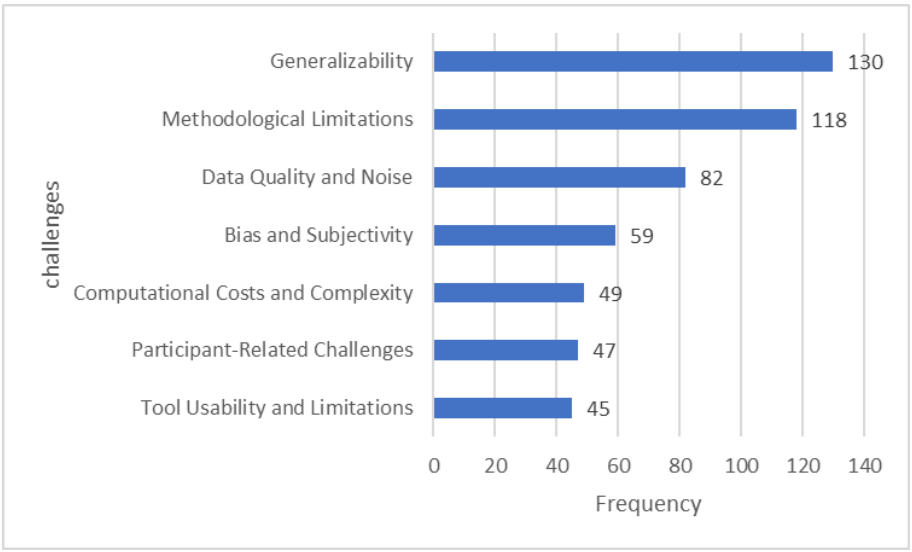

Fig. 11. The most common challenges.

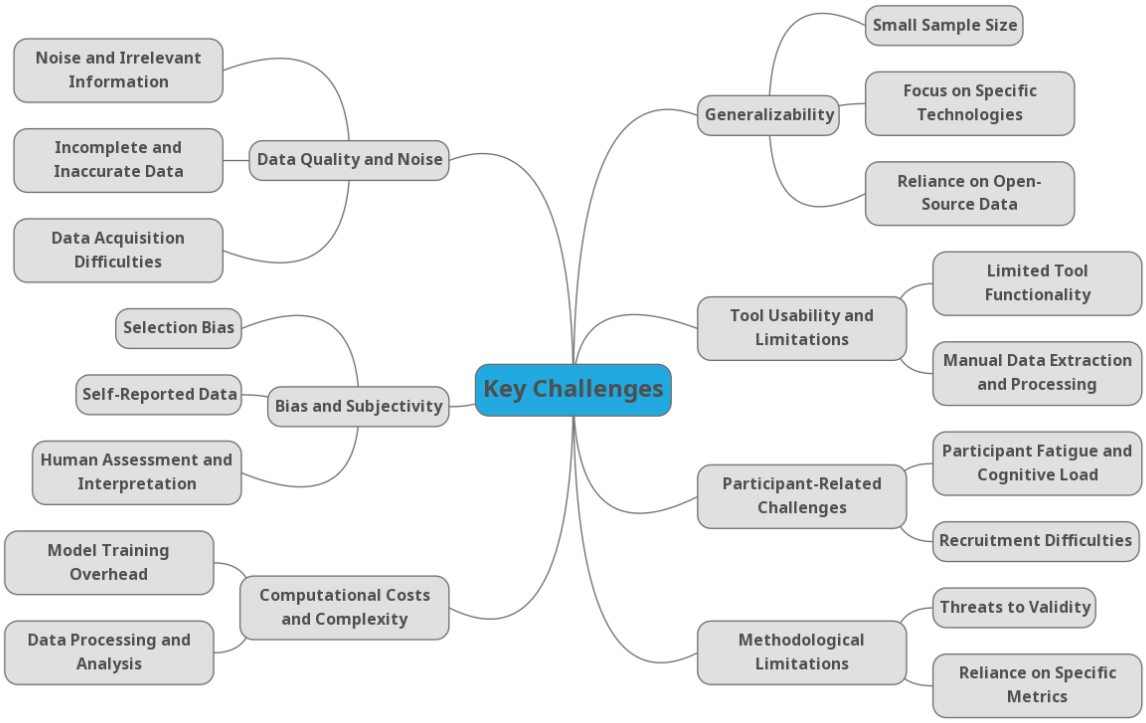

Fig. 12. The challenges.

[4] M. Felderer and G. H. Travassos, "The evolution of empirical methods in software engineering," *Springer*, pp. 1–24, 2020.

[5] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design in empirical software engineering," *Empirical Software Engineering*, vol. 20, pp. 1427–1455, 2015.

[6] K.-J. Stol and B. Fitzgerald, "The abc of software engineering research," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 3, pp. 1–51, 2018.

[7] J. Melegati, K. Conboy, and D. Graziotin, "Qualitative surveys in software engineering research: Definition, critical review, and guidelines," *IEEE Transactions on Software Engineering*, 2024.

[8] L. Zhang, J.-H. Tian, J. Jiang, Y.-J. Liu, M.-Y. Pu, and T. Yue, "Empirical research in software engineering—a literature survey," *Journal of Computer Science and Technology*, vol. 33, pp. 876–899, 2018.

[9] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[10] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th international conference on evaluation and assessment in software engineering (EASE)*, BCS Learning & Development, 2008.

[11] B. Kitchenham and S. M. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*, no. January 2007, pp. 1–57, 2007.

[12] M. Ouzzani, H. Hammady, Z. Fedorowicz, and A. Elmagarmid, "Rayyan—a web and mobile app for systematic reviews," *Systematic reviews*, vol. 5, pp. 1–10, 2016.

[13] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*, vol. 236. Springer, 2012.

[14] J. Al Dallal, "Categorisation-based approach for predicting the fault-proneness of object-oriented classes in software post-releases," *IET Software*, vol. 14, no. 5, pp. 525–534, 2020.

[15] W. Lam, S. Winter, A. Wei, T. Xie, D. Marinov, and J. Bell, "A large-scale longitudinal study of flaky tests," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 1–29, 2020.

[16] O. Nourry, Y. Kashiwa, W. Shang, H. Shu, and Y. Kamei, "My fuzzers won't build: An empirical study of fuzzing build failures," *ACM Transactions on Software Engineering and Methodology*.

[17] D. Weyns, I. Gerostathopoulos, N. Abbas, J. Andersson, S. Biffl, P. Brada, T. Bures, A. Di Salle, M. Galster, P. Lago, *et al.*, "Self-adaptation in industry: A survey," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 18, no. 2, pp. 1–44, 2023.

[18] U. I. Janjua, T. M. Madni, M. F. Cheema, and A. R. Shahid, "An empirical study to investigate the impact of communication issues in gsd in pakistan's it industry," *IEEE Access*, vol. 7, pp. 171648–171672, 2019.

[19] C. Mandrioli and M. Maggio, "Testing self-adaptive software with probabilistic guarantees on performance metrics: extended and comparative results," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3554–3572, 2021.

[20] F. Palomba, D. Andrew Tamburri, F. Arcelli Fontana, R. Oliveto, A. Zaidman, A. Serebrenik, *et al.*, "Beyond technical aspects: How do community smells influence the intensity of code smells?," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 47, no. 1, pp. 108–129, 2021.

[21] A. C. Oran, N. Valentim, G. Santos, and T. Conte, "Why use case specifications are hard to use in generating prototypes?," *IET Software*, vol. 13, no. 6, pp. 510–517, 2019.

[22] H. Ben Braiek and F. Khomh, "Testing feedforward neural networks training programs," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 4, pp. 1–61, 2023.

[23] X. Tan and M. Zhou, "How to communicate when submitting patches: An empirical study of the linux kernel," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–26, 2019.

[24] J. Wang, Y. Li, Q. Qi, Y. Lu, and B. Wu, "Multilayered fault detection and localization with transformer for microservice systems," *IEEE Transactions on Reliability*, 2024.

[25] Y. Lyu, H. Li, Z. M. Jiang, and A. E. Hassan, "On the model update strategies for supervised learning in aiops solutions," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 7, pp. 1–38, 2024.

[26] S. K. Pandey, D. Rathee, and A. K. Tripathi, "Software defect prediction using k-pca and various kernel-based extreme learning machine: an empirical study," *IET Software*, vol. 14, no. 7, pp. 768–782, 2020.

[27] C.-A. Sun, A. Fu, J. Jia, M. Li, and J. Han, "Improving conformance of web services: A constraint-based model-driven approach," *ACM Transactions on the Web*, vol. 17, no. 2, pp. 1–36, 2023.

[28] J. C. Neto, C. H. da Silva, T. E. Colanzi, and A. M. M. M. Amaral, "Are mas profitable to search-based pla design?," *IET Softw.*, vol. 13, no. 6, pp. 587–599, 2019.

[29] M. Kuhrmann, P. Tell, R. Hebig, J. Klünder, J. Münch, O. Linssen, D. Pfahl, M. Felderer, C. R. Prause, S. G. MacDonell, *et al.*, "What makes agile software development agile?," *IEEE transactions on software engineering*, vol. 48, no. 9, pp. 3523–3539, 2021.

[30] E. Kalliamvakou, C. Bird, T. Zimmermann, A. Begel, R. DeLine, and D. M. German, "What makes a great manager of software engineers?," *IEEE Transactions on Software Engineering*, vol. 45, no. 01, pp. 87–106, 2019.

[31] J. Liu, Y. Huang, Z. Wang, L. Ma, C. Fang, M. Gu, X. Zhang, and Z. Chen, "Generation-based differential fuzzing for deep learning libraries," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 2, pp. 1–28, 2023.

[32] A. Gupta, R. Gandhi, N. Jatana, D. Jatain, S. K. Panda, and J. V. N. Ramesh, "A severity assessment of python code smells," *IEEE Access*, 2023.

[33] Y. Li, T. Zhang, X. Luo, H. Cai, S. Fang, and D. Yuan, "Do pre-trained language models indeed understand software engineering tasks?," *IEEE Transactions on Software Engineering*, 2023.

[34] C. Wang, H. He, U. Pal, D. Marinov, and M. Zhou, "Suboptimal comments in java projects: From independent comment changes to commenting practices," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 2, pp. 1–33, 2023.

[35] Q. Chen, C. Yu, R. Liu, C. Zhang, Y. Wang, K. Wang, T. Su, and L. Wang, "Evaluating the effectiveness of deep learning models for foundational program analysis tasks," *Proceedings of the ACM on Programming Languages*, vol. 8, no. OOPSLA1, pp. 500–528, 2024.

[36] Y. Qu, Q. Zheng, J. Chi, Y. Jin, A. He, D. Cui, H. Zhang, and T. Liu, "Using k-core decomposition on class dependency networks to improve bug prediction model's practical

performance," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 348–366, 2019.

[37] A. Vranković, T. Galinac Grbac, and Ž. Car, "Software structure evolution and relation to subgraph defectiveness," *Iet software*, vol. 13, no. 5, pp. 355–367, 2019.

[38] B. Komal, U. I. Janjua, F. Anwar, T. M. Madni, M. F. Cheema, M. N. Malik, and A. R. Shahid, "The impact of scope creep on project success: An empirical investigation," *IEEE Access*, vol. 8, pp. 125755–125775, 2020.

[39] M. Garriga, S. Dalla Palma, M. Arias, A. De Renzis, R. Pareschi, and D. Andrew Tamburri, "Blockchain and cryptocurrencies: A classification and comparison of architecture drivers," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 8, p. e5992, 2021.

[40] C. Camacho, P. C. Cañizares, L. Llana, and A. Núñez, "Chaos as a software product line—a platform for improving open hybrid-cloud systems resiliency," *Software: Practice and Experience*, vol. 52, no. 7, pp. 1581–1614, 2022.

[41] C. A. Furia, R. Torkar, and R. Feldt, "Applying bayesian analysis guidelines to empirical software engineering data: The case of programming languages and code quality," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 3, pp. 1–38, 2022.

[42] L. Erazo-Garzón, P. Cedillo, G. Rossi, and J. Moyano, "A domain-specific language for modeling iot system architectures that support monitoring," *IEEE Access*, vol. 10, pp. 61639–61665, 2022.

[43] M. Zhang, A. Belhadi, and A. Arcuri, "Javascript sbst heuristics to enable effective fuzzing of nodejs web apis," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, pp. 1–29, 2023.

[44] F. Ebrahimi, M. Tushev, and A. Mahmoud, "Classifying mobile applications using word embeddings," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–30, 2021.

[45] L. Lavazza, A. Locoro, G. Liu, and R. Meli, "Estimating software functional size via machine learning," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, pp. 1–27, 2023.

[46] T. Chen and M. Li, "The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 1, pp. 1–40, 2023.

[47] S. Hussain, J. Keung, M. K. Sohail, A. A. Khan, G. Ahmad, M. R. Mufti, and H. A. Khatak, "Methodology for the quantification of the effect of patterns and anti-patterns association on the software quality," *IET Software*, vol. 13, no. 5, pp. 414–422, 2019.

[48] X. Ju, J. Qian, Z. Chen, C. Zhao, and J. Qian, "Mulr4fl: effective fault localization of evolution software based on multivariate logistic regression model," *Ieee Access*, vol. 8, pp. 207858–207870, 2020.

[49] C. Wan, Y. Liu, K. Du, H. Hoffmann, J. Jiang, M. Maire, and S. Lu, "Run-time prevention of software integration failures of machine learning apis," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA2, pp. 264–291, 2023.

[50] Y. Xiang, H. Huang, M. Li, S. Li, and X. Yang, "Looking for novelty in search-based software product line testing," *IEEE Transactions on Software Engineering*, vol. 48, no. 07, pp. 2317–2338, 2022.

[51] H. Lu, Z. Liu, S. Wang, and F. Zhang, "Dtd: Comprehensive and scalable testing for debuggers," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1172–1193, 2024.

[52] X. Chen, Z. Yuan, Z. Cui, D. Zhang, and X. Ju, "Empirical studies on the impact of filter-based ranking feature selection on security vulnerability prediction," *IET Software*, vol. 15, no. 1, pp. 75–89, 2021.

[53] A. Agrawal and R. K. Singh, "Predicting co-change probability in software applications using historical metadata," *IET Software*, vol. 14, no. 7, pp. 739–747, 2020.

[54] E. Merrill, A. Fern, X. Fern, and N. Dolatnia, "An empirical study of bayesian optimization: Acquisition versus partition," *Journal of Machine Learning Research*, vol. 22, no. 4, pp. 1–25, 2021.

[55] J. I. Panach, Ó. Dieste, B. Marín, S. España, S. Vegas, Ó. Pastor, and N. Juristo, "Evaluating model-driven development claims with respect to quality: A family of experiments," *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 130–145, 2021.

[56] C. Birchler, S. Khatiri, P. Derakhshanfar, S. Panichella, and A. Panichella, "Single and multi-objective test cases prioritization for self-driving cars in virtual environments," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 2, pp. 1–30, 2023.

[57] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, 2020.

[58] S. Zhang, S. Jiang, and Y. Yan, "A Software Defect Prediction Approach Based on Hybrid Feature Dimensionality Reduction," *Scientific Programming*, vol. 2023, no. 1, p. 5585130, 2023.

[59] C. Gavidia-Calderon, F. Sarro, M. Harman, and E. T. Barr, "The assessor's dilemma: Improving bug repair via empirical game theory," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2143–2161, 2021.

[60] M. A. Akbar, S. Mahmood, A. Alsanad, M. Shafiq, A. Gumaei, and A. A.-A. Alsanad, "Organization type and size based identification of requirements change management challenges in global software development," *IEEE Access*, vol. 8, pp. 94089–94111, 2020.

[61] H. Kahtan, M. Abdulhak, A. S. Al-Ahmad, and Y. I. Alzoubi, "A model for developing dependable systems using a component-based software development approach (mddscbsd)," *IET Software*, vol. 17, no. 1, pp. 76–92, 2023.

[62] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, H. C. Gall, *et al.*, "Exploiting natural language structures in software informal documentation," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 47, no. 8, pp. 1587–1604, 2021.

[63] D. Issa Mattos, A. Dakkak, J. Bosch, and H. H. Olsson, "The hurrier process for experimentation in business-to-business mission-critical systems," *Journal of Software: Evolution and Process*, vol. 35, no. 5, p. e2390, 2023.

[64] M. Ojdanic, E. Soremekun, R. Degiovanni, M. Papadakis, and Y. Le Traon, "Mutation testing in evolving systems: Studying the relevance of mutants to code evolution," *ACM*

Transactions on Software Engineering and Methodology, vol. 32, no. 1, pp. 1–39, 2023.

[65] I. Illahi, H. Liu, Q. Umer, and S. A. H. Zaidi, "An empirical study on competitive crowdsource software development: motivating and inhibiting factors," *IEEE Access*, vol. 7, pp. 62042–62057, 2019.

[66] P. Ciancarini, M. Farina, S. Masyagin, G. Succi, S. Yermolaieva, and N. Zagvozkina, "Non verbal communication in software engineering–an empirical study," *IEEE Access*, vol. 9, pp. 71942–71953, 2021.

[67] J. Cheng, C. Gao, and Z. Zheng, "Hinnperf: Hierarchical interaction neural network for performance prediction of configurable systems," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 2, pp. 1–30, 2023.

[68] R. Fatima, A. Yasin, L. Liu, J. Wang, W. Afzal, and A. Yasin, "Improving software requirements reasoning by novices: a story-based approach," *IET Software*, vol. 13, no. 6, pp. 564–574, 2019.

[69] A. B. Fuertes, M. Pérez, and J. Meza, "nmorph framework: An innovative approach to transpiler-based multi-language software development," *IEEE Access*, vol. 11, pp. 124386–124429, 2023.

[70] Z. Wang, S. Xu, L. Fan, X. Cai, L. Li, and Z. Liu, "Can coverage criteria guide failure discovery for image classifiers? an empirical study," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 7, pp. 1–28, 2024.

[71] J. Jiarpakdee, C. Tantithamthavorn, and A. E. Hassan, "The impact of correlated metrics on the interpretation of defect models," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 320–331, 2021.

[72] P. Temple, M. Acher, and J.-M. Jezequel, "Empirical assessment of multimorphic testing," *IEEE Transactions on Software Engineering*, vol. 47, no. 07, pp. 1511–1527, 2021.

[73] P. Wang, X. Zhou, T. Yue, P. Lin, Y. Liu, and K. Lu, "The progress, challenges, and perspectives of directed greybox fuzzing," *Software Testing, Verification and Reliability*, vol. 34, no. 2, p. e1869, 2024.

[74] A. Souza, B. Ferreira, N. Valentim, L. Correa, S. Marczak, and T. Conte, "Supporting the teaching of design thinking techniques for requirements elicitation through a recommendation tool," *IET Software*, vol. 14, no. 6, pp. 693–701, 2020.

[75] S. Meliá, R. Reyes, and C. Cachero, "The effect of developers' general intelligence on the understandability of domain models: an empirical study," *IEEE Access*, 2023.

[76] F. Zampetti, D. Tamburri, S. Panichella, A. Panichella, G. Canfora, and M. Di Penta, "Continuous integration and delivery practices for cyber-physical systems: An interview-based study," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 3, pp. 1–44, 2023.

[77] R. Nadri, G. Rodríguez-Pérez, and M. Nagappan, "On the relationship between the developer's perceptible race and ethnicity and the evaluation of contributions in oss," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2955–2968, 2022.

[78] W. Hutiri, A. Y. Ding, F. Kawsar, and A. Mathur, "Tiny, always-on, and fragile: Bias propagation through design choices in on-device machine learning workflows," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, pp. 1–37, 2023.

[79] S. Dashevskyi, A. Brucker, and F. Massacci, "A screening test for disclosed vulnerabilities in foss components," *IEEE Transactions on Software Engineering*, vol. 45, no. 10, pp. 945–966, 2019.

[80] D. A. Tamburri and F. Palomba, "Evolving software forges: An experience report from apache allura," *Journal of Software: Evolution and Process*, vol. 33, no. 12, p. e2397, 2021.

[81] S. Romano, C. Vendome, G. Scanniello, D. Poshyvanyk, *et al.*, "A multi-study investigation into dead code," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 46, no. 1, pp. 71–99, 2020.

[82] M. M. John, H. H. Olsson, and J. Bosch, "Towards an ai-driven business development framework: A multi-case study," *Journal of Software: Evolution and Process*, vol. 35, no. 6, p. e2432, 2023.

[83] E. Dilorenzo, E. Dantas, M. Perkusich, F. Ramos, A. Costa, D. Albuquerque, H. Almeida, and A. Perkusich, "Enabling the reuse of software development assets through a taxonomy for user stories," *IEEE Access*, vol. 8, pp. 107285–107300, 2020.

[84] Q. Hu, Y. Guo, X. Xie, M. Cordy, L. Ma, M. Papadakis, and Y. Le Traon, "Active code learning: Benchmarking sample-efficient training of code models," *IEEE Transactions on Software Engineering*, 2024.

[85] E. Edward, A. S. Nyamawe, and N. Elisa, "On the impact of refactorings on software attack surface," *IEEE Access*, 2024.

[86] W. Mauerer, M. Joblin, D. Tamburri, C. Paradis, R. Kazman, and S. Apel, "In search of socio-technical congruence: A large-scale longitudinal study," *IEEE Transactions on Software Engineering (TSE)*, vol. 48, no. 8, pp. 3159–3184, 2022.

[87] C. Birchler, T. K. Mohammed, P. Rani, T. Nechita, T. Kehrer, and S. Panichella, "How does simulation-based testing for self-driving cars match human perception?," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 929–950, 2024.

[88] M. Tufano, C. Watson, G. Bavota, M. D. Penta, M. White, and D. Poshyvanyk, "An empirical study on learning bug-fixing patches in the wild via neural machine translation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 4, pp. 1–29, 2019.

[89] G. Uddin, Y.-G. Guéhénuc, F. Khomh, and C. K. Roy, "An empirical study of the effectiveness of an ensemble of standalone sentiment detection tools for software engineering datasets," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 3, pp. 1–38, 2022.

[90] A. Almogahed, H. Mahdin, M. Omar, N. H. Zakaria, G. Muhammad, and Z. Ali, "Optimized refactoring mechanisms to improve quality characteristics in object-oriented systems," *IEEE Access*, 2023.

[91] W. Pan, H. Ming, C. K. Chang, Z. Yang, and D.-K. Kim, "Elementrank: Ranking java software classes and packages using a multilayer complex network-based approach," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2272–2295, 2021.

[92] D. Girardi, F. Lanubile, N. Novielli, and A. Serebrenik, "Emotions and perceived productivity of software developers at the workplace," *IEEE Transactions on Software Engineering*, vol. 48, no. 09, pp. 3326–3341, 2022.

[93] T. Bi, B. Xia, Z. Xing, Q. Lu, and L. Zhu, "On the way to sboms: Investigating design issues and solutions in practice," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, pp. 1–25, 2024.

[94] R. Torkar, C. A. Furia, R. Feldt, F. G. de Oliveira Neto, L. Gren, P. Lenberg, and N. A. Ernst, "A method to assess and argue for practical significance in software engineering," 2022.

[95] S. Di Martino, A. R. Fasolino, L. L. L. Starace, and P. Tramontana, "Gui testing of android applications: Investigating the impact of the number of testers on different exploratory testing strategies," *Journal of Software: Evolution and Process*, vol. 36, no. 7, p. e2640, 2024.

[96] M. Emmi and C. Enea, "Weak-consistency specification via visibility relaxation," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–28, 2019.

[97] C.-a. Sun, H. Dai, H. Liu, T. Y. Chen, and K.-Y. Cai, "Adaptive partition testing," *IEEE Transactions on Computers*, vol. 68, no. 02, pp. 157–169, 2019.

[98] F. M. Almansour, R. Alroobaea, and A. S. Ghiduk, "An empirical comparison of the efficiency and effectiveness of genetic algorithms and adaptive random techniques in dataflow testing," *IEEE Access*, vol. 8, pp. 12884–12896, 2020.

[99] D. Jayasuriya, V. Terragni, J. Dietrich, and K. Blincoe, "Understanding the impact of apis behavioral breaking changes on client applications," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1238–1261, 2024.

[100] M. A. Babar, H. Shen, S. Biffl, and D. Winkler, "An empirical study of the effectiveness of software architecture evaluation meetings," *IEEE Access*, vol. 7, pp. 79069–79084, 2019.

[101] D. Olewicki, S. Habchi, and B. Adams, "An empirical study on code review activity prediction and its impact in practice," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 2238–2260, 2024.

[102] M. Openja, F. Khomh, A. Foundjem, Z. M. Jiang, M. Abidi, and A. E. Hassan, "An empirical study of testing machine learning in the wild," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 1, pp. 1–63, 2024.

[103] Y. Wu, Y. Zhang, T. Wang, and H. Wang, "Characterizing the occurrence of dockerfile smells in open-source software: An empirical study," *IEEE Access*, vol. 8, pp. 34127–34139, 2020.

[104] D. López-Fernández, J. Mayor, J. Pérez, and A. Gordillo, "Learning and motivational impact of using a virtual reality serious video game to learn scrum," *IEEE Transactions on Games*, vol. 15, no. 3, pp. 430–439, 2022.

[105] M. Paltenghi and M. Pradel, "Analyzing quantum programs with lintq: A static analysis framework for qiskit," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 2144–2166, 2024.

[106] M. Londschien, P. Bühlmann, and S. Kovács, "Random forests for change point detection," *arXiv e-prints*, pp. arXiv–2205, 2022.

[107] F. Sarro, R. Moussa, A. Petrozziello, and M. Harman, "Learning from mistakes: Machine learning enhanced human expert effort estimates," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 1868–1882, 2022.

[108] A. Katbi, M. Hammad, and W. Elmedany, "Multi-view city-based approach for code-smell evolution visualisation," *IET Software*, vol. 14, no. 5, pp. 506–516, 2020.

[109] C. D. Ngo, F. Pastore, and L. Briand, "Testing updated apps by adapting learned models," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, pp. 1–40, 2024.

[110] A. M. Aranda, Ó. Dieste, J. I. Panach Navarrete, and N. Juristo, "Effect of requirements analyst experience on elicitation effectiveness: a family of quasi-experiments," 2023.

[111] P. Gyimesi, B. Vancsics, A. Stocco, D. Mazinanian, Á. Beszédes, R. Ferenc, and A. Mesbah, "Bugsjs: a benchmark and taxonomy of javascript bugs," *Software Testing, Verification and Reliability*, vol. 31, no. 4, p. e1751, 2021.

[112] S. Xu, Y. Gao, L. Fan, Z. Liu, Y. Liu, and H. Ji, "Lidetector: License incompatibility detection for open source software," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 1, pp. 1–28, 2023.

[113] A. S. Ghiduk and A. M. Qahtani, "An empirical study of local-decision-making-based software customization in distributed development," *IET Software*, vol. 15, no. 2, pp. 174–187, 2021.

[114] N. Qamar and A. A. Malik, "A quantitative assessment of the impact of homogeneity in personality traits on software quality and team productivity," *IEEE Access*, vol. 10, pp. 122092–122111, 2022.

[115] O. Pedreira, F. Silva-Coira, A. S. Places, M. R. Luaces, and L. G. Folgueira, "Applying feature-oriented software development in saas systems: Real experience, measurements, and findings," *Journal of Web Engineering*, vol. 18, no. 4–6, pp. 447–475, 2019.

[116] S. Tahvili, R. Pimentel, W. Afzal, M. Ahlberg, E. Fornander, and M. Bohlin, "sortes: A supportive tool for stochastic scheduling of manual integration test cases," *IEEE Access*, vol. 7, pp. 12928–12946, 2019.

[117] M. Dilhara, A. Ketkar, and D. Dig, "Understanding software-2.0: A study of machine learning library usage and evolution," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 4, pp. 1–42, 2021.

[118] X. Fu, H. Cai, W. Li, and L. Li, "S eads: Scalable and cost-effective dynamic dependence analysis of distributed systems via reinforcement learning," *ACM Transactions on Software Engineering and Methodology*, vol. 30, no. 1, 2021.

[119] H. S. Qiu, Y. L. Li, S. Padala, A. Sarma, and B. Vasilescu, "The signals that potential contributors look for when choosing open-source projects," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–29, 2019.

[120] J. Sandobalin, E. Insfran, and S. Abrahao, "On the effectiveness of tools to support infrastructure as code: Model-driven versus code-centric," *IEEE Access*, vol. 8, pp. 17734–17761, 2020.

[121] B. Lin, S. Wang, Z. Liu, X. Xia, and X. Mao, "Predictive comment updating with heuristics and ast-path-based neural learning: A two-phase approach," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1640–1660, 2023.

[122] P. Yang, Z. Liu, J. Xu, Y. Huang, and Y. Pan, "An empirical study on the ability relationships between programming and testing," *IEEE access*, vol. 8, pp. 161438–161448, 2020.

[123] Y. He, J. Huang, H. Yu, and T. Xie, "An empirical study on focal methods in deep-learning-based approaches for asser-

tion generation," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1750–1771, 2024.

[124] H. Jahanshahi, M. Cevik, K. Mousavi, and A. Başar, "Adptriage: Approximate dynamic programming for bug triage," *IEEE Transactions on Software Engineering*, 2023.

[125] C. A. Furia, R. Feldt, and R. Torkar, "Bayesian data analysis in empirical software engineering research," *IEEE Transactions on Software Engineering*, vol. 47, no. 09, pp. 1786–1810, 2021.

[126] R. M. Shadab, Y. Zou, S. Gandham, A. Awad, and M. Lin, "Hmt: A hardware-centric hybrid bonsai merkle tree algorithm for high-performance authentication," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 4, pp. 1–28, 2023.

[127] C.-a. Sun, A. Fu, X. Guo, and T. Y. Chen, "Remusse: A redundant mutant identification technique based on selective symbolic execution," *IEEE Transactions on Reliability*, vol. 71, no. 1, pp. 415–428, 2020.

[128] X. Zhang, T. F. Stafford, T. Hu, and H. Dai, "Measuring task conflict and person conflict in software testing," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–19, 2020.

[129] M. Chouchen, A. Ouni, and M. W. Mkaouer, "Multicr: Predicting merged and abandoned code changes in modern code review using multi-objective search," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp. 1–44, 2024.

[130] T. Bock, N. Alznauer, M. Joblin, and S. Apel, "Automatic core-developer identification on github: a validation study," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, pp. 1–29, 2023.

[131] A. Ahmad, O. Leifler, and K. Sandahl, "Empirical analysis of practitioners' perceptions of test flakiness factors," *Software Testing, Verification and Reliability*, vol. 31, no. 8, p. e1791, 2021.

[132] K.-J. Stol, B. Caglayan, and B. Fitzgerald, "Competition-based crowdsourcing software development: A multi-method study from a customer perspective," *IEEE Transactions on Software Engineering*, vol. 45, no. 03, pp. 237–260, 2019.

[133] E. Rosales, M. Basso, A. Rosà, and W. Binder, "Large-scale characterization of java streams," *Software: Practice and Experience*, vol. 53, no. 9, pp. 1763–1792, 2023.

[134] X. Zhang, Z. Lin, X. Hu, J. Wang, W. Lu, and D. Zhou, "Secon: Maintaining semantic consistency in data augmentation for code search," *ACM Transactions on Information Systems*, 2024.

[135] M. Islam, A. K. Jha, I. Akhmetov, and S. Nadi, "Characterizing python library migrations," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 92–114, 2024.

[136] D. Hellhake, J. Bogner, T. Schmid, and S. Wagner, "Towards using coupling measures to guide black-box integration testing in component-based systems," *Software Testing, Verification and Reliability*, vol. 32, no. 4, p. e1811, 2022.

[137] N. Qamar and A. A. Malik, "Birds of a feather gel together: Impact of team homogeneity on software quality and team productivity," *IEEE Access*, vol. 7, pp. 96827–96840, 2019.

[138] M. Almaliki, "Misinformation-aware social media: A software engineering perspective," *IEEE Access*, vol. 7, pp. 182451–182458, 2019.

[139] J. Chen and C. Suo, "Boosting compiler testing via compiler optimization exploration," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–33, 2022.

[140] F. A. Bhuiyan and A. Rahman, "Log-related coding patterns to conduct postmortems of attacks in supervised learning-based projects," *ACM Transactions on Privacy and Security*, vol. 26, no. 2, pp. 1–24, 2023.

[141] R. Coppola, T. Fulcini, L. Ardito, M. Torchiano, and E. Alègroth, "On effectiveness and efficiency of gamified exploratory gui testing," *IEEE Transactions on Software Engineering*, vol. 50, no. 2, pp. 322–337, 2024.

[142] Y. Zhang, Z. Qiu, K.-J. Stol, W. Zhu, J. Zhu, Y. Tian, and H. Liu, "Automatic commit message generation: A critical review and directions for future work," *IEEE Transactions on Software Engineering*, 2024.

[143] K. Müller, C. Koch, D. Riehle, M. Stops, and N. Harutyunyan, "Challenges of working from home in software development during covid-19 lockdowns," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, pp. 1–41, 2023.

[144] K. Liu, D. Kim, T. F. Bissyandé, S. Yoo, and Y. Le Traon, "Mining fix patterns for findbugs violations," *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 165–188, 2021.

[145] C. Karanikolas, G. Dimitroulakos, and K. Masselos, "Simulating software evolution to evaluate the reliability of early decision-making among design alternatives toward maintainability," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 3, pp. 1–38, 2023.

[146] M. R. Wróbel, J. Szymukowicz, and P. Weichbroth, "Using continuous integration techniques in open source projects–an exploratory study," *IEEE Access*, 2023.

[147] T. Bi, X. Xia, D. Lo, J. Grundy, and T. Zimmermann, "An empirical study of release note production and usage in practice," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 1834–1852, 2022.

[148] Z. Li, Y. Yu, T. Wang, S. Li, and H. Wang, "Opportunities and challenges in repeated revisions to pull-requests: An empirical study," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 1–35, 2022.

[149] X. Wang, A. Muqeet, T. Yue, S. Ali, and P. Arcaini, "Test case minimization with quantum annealers," *ACM Transactions on Software Engineering and Methodology*, 2024.

[150] P. Zhu, Y. Li, T. Li, W. Yang, and Y. Xu, "Gui widget detection and intent generation via image understanding," *IEEE Access*, vol. 9, pp. 160697–160707, 2021.

[151] X. Zheng, Z. Wan, Y. Zhang, R. Chang, and D. Lo, "A closer look at the security risks in the rust ecosystem," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 2, pp. 1–30, 2023.

[152] A. Zirak and H. Hemmati, "Improving automated program repair with domain adaptation," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 3, pp. 1–43, 2024.

[153] P. Rutz, C. Kotthaus, A. F. Pinatti de Carvalho, D. Randall, and V. Pipek, "The relevance of kes-oriented processes for the implementation of erp systems: Findings from an empirical study in german smes," *Proceedings of the ACM on*

Human-Computer Interaction, vol. 7, no. CSCW2, pp. 1–34, 2023.

[154] Z. Huang, J. Chen, J. Jiang, Y. Liang, H. You, and F. Li, "Mapping apis in dynamic-typed programs by leveraging transfer learning," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 4, pp. 1–29, 2024.

[155] A. Rahman, S. I. Shamim, D. B. Bose, and R. Pandita, "Security misconfigurations in open source kubernetes manifests: An empirical study," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 4, pp. 1–36, 2023.

[156] C. Bogart, C. Kästner, J. Herbsleb, and F. Thung, "When and how to make breaking changes: Policies and practices in 18 open source software ecosystems," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 4, pp. 1–56, 2021.

[157] A. Javan Jafari, D. E. Costa, E. Shihab, and R. Abdalkareem, "Dependency update strategies and package characteristics," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, pp. 1–29, 2023.

[158] R. Kumar, A. Chaturvedi, and L. Kailasam, "An unsupervised software fault prediction approach using threshold derivation," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 911–932, 2022.

[159] K. Farias, T. de Oliveira Cavalcante, L. José Gonçales, and V. Bischoff, "Uml2merge: a uml extension for model merging," *IET Software*, vol. 13, no. 6, pp. 575–586, 2019.

[160] D. Rajapaksha, C. Tantithamthavorn, J. Jiarpakdee, C. Bergmeir, J. Grundy, and W. Buntine, "Sqaplanner: generating data-informed software quality improvement plans," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2814–2835, 2022.

[161] M. Daud and A. A. Malik, "Improving the accuracy of early software size estimation using analysis-to-design adjustment factors (adafs)," *IEEE Access*, vol. 9, pp. 81986–81999, 2021.

[162] S. Wagner, D. M. Fernández, M. Felderer, A. Vetrò, M. Kalinowski, R. Wieringa, D. Pfahl, T. Conte, M.-T. Christiansson, D. Greer, *et al.*, "Status quo in requirements engineering: A theory and a global family of surveys," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 2, pp. 1–48, 2019.

[163] M. M. Hassan, J. Salvador, S. K. K. Santu, and A. Rahman, "State reconciliation defects in infrastructure as code," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1865–1888, 2024.

[164] B. Kitchenham, L. Madeyski, G. Scanniello, and C. Gravino, "The importance of the correlation in crossover experiments," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2802–2813, 2022.

[165] S. Ali, P. Arcaini, D. Pradhan, S. A. Safdar, and T. Yue, "Quality indicators in search-based software engineering: An empirical evaluation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 2, pp. 1–29, 2020.

[166] E. Zabardast, K. Ebo Bennin, and J. Gonzalez-Huerta, "Further investigation of the survivability of code technical debt items," *Journal of Software: Evolution and Process*, vol. 34, no. 2, p. e2425, 2022.

[167] M. Soltani, A. Panichella, and A. van Deursen, "Search-based crash reproduction and its impact on debugging," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1294–1317, 2020.

[168] R. Koitz-Hristov, T. Sterner, L. Stracke, and F. Wotawa, "On the suitability of checked coverage and genetic parameter tuning in test suite reduction," *Journal of Software: Evolution and Process*, p. e2656, 2024.

[169] M. Joblin and S. Apel, "How do successful and failed projects differ? a socio-technical analysis," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–24, 2022.

[170] A. Bertolino, G. De Angelis, B. Miranda, and P. Tonella, "In vivo test and rollback of java applications as they are," *Software Testing, Verification and Reliability*, vol. 33, no. 7, p. e1857, 2023.

[171] M. Choraś, T. Springer, R. Kozik, L. Lopez, S. Martínez-Fernández, P. Ram, P. Rodriguez, and X. Franch, "Measuring and improving agile processes in a small-size software development company," *IEEE access*, vol. 8, pp. 78452–78466, 2020.

[172] W. Behutiye, N. Tripathi, and M. Isomursu, "Adopting scrum in hybrid settings, in a university course project: Reflections and recommendations," *IEEE Access*, 2024.

[173] J. N. Qureshi, M. S. Farooq, A. Khelifi, and Z. Atal, "Harnessing the potential of blockchain in chainagileplus framework for the improvement of distributed scrum of scrums agile software development," *IEEE Access*, 2024.

[174] S. Gilmer, A. Bhat, S. Shah, K. Cherry, J. Cheng, and J. L. Guo, "Summit: Scaffolding open source software issue discussion through summarization," *Proceedings of the ACM on Human-Computer Interaction*, vol. 7, no. CSCW2, pp. 1–27, 2023.

[175] T. Chen and M. Li, "Do performance aspirations matter for guiding software configuration tuning? an empirical investigation under dual performance objectives," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 3, pp. 1–41, 2023.

[176] Q. Yu, S. Jiang, J. Qian, L. Bo, L. Jiang, and G. Zhang, "Process metrics for software defect prediction in object-oriented programs," *IET Software*, vol. 14, no. 3, pp. 283–292, 2020.

[177] R. Arizon-Peretz, I. Hadar, and G. Luria, "The importance of security is in the eye of the beholder: Cultural, organizational, and personal factors affecting the implementation of security by design," *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4433–4446, 2022.

[178] D. A. A. Tamburri, F. Palomba, and R. Kazman, "Exploring community smells in open-source: an automated approach," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 630–652, 2021.

[179] W. Sun, M. Yan, Z. Liu, X. Xia, Y. Lei, and D. Lo, "Revisiting the identification of the co-evolution of production and test code," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, pp. 1–37, 2023.

[180] M. Nayebi, G. Ruhe, and T. Zimmermann, "Mining treatment-outcome constructs from sequential software engineering data," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 393–411, 2021.

[181] A. Metzger, J. Laufer, F. Feit, and K. Pohl, "A user study on explainable online reinforcement learning for adaptive systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 19, no. 3, pp. 1–44, 2024.

[182] H. U. Rahman, M. Raza, P. Afsar, and H. U. Khan, "Empirical investigation of influencing factors regarding offshore outsourcing decision of application maintenance," *IEEE Access*, vol. 9, pp. 58589–58608, 2021.

[183] C. Wei, X. Yao, D. Gong, and H. Liu, "Test data generation for mutation testing based on markov chain usage model and estimation of distribution algorithm," *IEEE Transactions on Software Engineering*, 2024.

[184] S. Yu, C. Fang, Q. Zhang, Z. Cao, Y. Yun, Z. Cao, K. Mei, and Z. Chen, "Mobile app crowdsourced test report consistency detection via deep image-and-text fusion understanding," *IEEE Transactions on Software Engineering*, vol. 49, no. 8, pp. 4115–4134, 2023.

[185] R. Khojah, M. Mohamad, P. Leitner, and F. G. de Oliveira Neto, "Beyond code generation: An observational study of chatgpt usage in software engineering practice," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1819–1840, 2024.

[186] R. Amirova, G. Dlamini, A. Repryntseva, G. Succi, and H. Tarasau, "Attention and concentration for software developers," *IEEE Access*, 2023.

[187] G.-I. Trujillo-Tzanahua, U. Juárez-Martínez, A.-A. Aguilar-Lasserre, M.-K. Cortés-Verdín, and C. Azzaro-Pantel, "Multiple software product lines to configure applications of internet of things," *IET Software*, vol. 14, no. 2, pp. 165–175, 2020.

[188] A. Rahman, M. R. Rahman, C. Parnin, and L. Williams, "Security smells in ansible and chef scripts: A replication study," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 1, pp. 1–31, 2021.

[189] H. U. Khan, M. Niazi, M. El-Attar, N. Ikram, S. U. Khan, and A. Q. Gill, "Empirical investigation of critical requirements engineering practices for global software development," *IEEE Access*, vol. 9, pp. 93593–93613, 2021.

[190] P. Liu, L. Li, K. Liu, S. McIntosh, and J. Grundy, "Understanding the quality and evolution of android app build systems," *Journal of Software: Evolution and Process*, vol. 36, no. 5, p. e2602, 2024.

[191] C. Gavidia-Calderon, A. Kordoni, A. Bennaceur, M. Levine, and B. Nuseibeh, "The idea of us: An identity-aware architecture for autonomous systems," *ACM Transactions on Software Engineering and Methodology*, 2024.

[192] B. K. Ozkan, R. Majumdar, and S. Oraee, "Trace aware random testing for distributed systems," *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 1–29, 2019.

[193] A. Razzaq, A. Ventresque, R. Koschke, A. De Lucia, and J. Buckley, "The effect of feature characteristics on the performance of feature location techniques," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 2066–2085, 2022.

[194] M. Hoffmann, D. Méndez, F. Fagerholm, and A. Luckhardt, "The human side of software engineering teams: an investigation of contemporary challenges," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 49, no. 1, pp. 211–225, 2023.

[195] M. Leotta, F. Ricca, A. Marchetto, and D. Olianas, "An empirical study to compare three web test automation approaches: Nlp-based, programmable, and capture&replay," *Journal of Software: Evolution and Process*, vol. 36, no. 5, p. e2606, 2024.

[196] F. Aizaz, S. U. R. Khan, J. A. Khan, A. Akhunzada, *et al.*, "An empirical investigation of factors causing scope creep in agile global software development context: a conceptual model for project managers," *IEEE Access*, vol. 9, pp. 109166–109195, 2021.

[197] A. Ahmad, C. Feng, K. Li, S. M. Asim, and T. Sun, "Toward empirically investigating non-functional requirements of ios developers on stack overflow," *IEEE Access*, vol. 7, pp. 61145–61169, 2019.

[198] G. A. Ahmed, J. V. Patten, Y. Han, G. Lu, W. Hou, D. Gregg, J. Buckley, and M. Chochlov, "Nearest-neighbor, bert-based, scalable clone detection: A practical approach for large-scale industrial code bases," *Software: Practice and Experience*, 2024.

[199] D. Amara, E. Fatnassi, and L. Ben Arfa Rabai, "An empirical assessment and validation of redundancy metrics using defect density as reliability indicator," *Scientific Programming*, vol. 2021, no. 1, p. 8325417, 2021.

[200] S. Sato and T. Nakamaru, "Multiverse notebook: Shifting data scientists to time travelers," *Proceedings of the ACM on Programming Languages*, vol. 8, no. OOPSLA1, pp. 754–783, 2024.

[201] A. Vizcaíno, F. García, I. G. R. D. Guzmán, and M. Á. Moraga, "Evaluating gsd-aware: A serious game for discovering global software development challenges," *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 2, pp. 1–23, 2019.

[202] Y. Shao and B. Xiang, "Enhancing bug report summaries through knowledge-specific and contrastive learning pretraining," *IEEE Access*, vol. 12, pp. 37653–37662, 2024.

[203] R. Huang, W. Sun, T. Y. Chen, S. Ng, and J. Chen, "Identification of failure regions for programs with numeric inputs," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 4, pp. 651–667, 2020.

[204] F. Liu, Z. Fu, G. Li, Z. Jin, H. Liu, Y. Hao, and L. Zhang, "Non-autoregressive line-level code completion," *ACM Transactions on Software Engineering and Methodology*, 2024.

[205] Z. Liao, X. Huang, B. Zhang, J. Wu, and Y. Cheng, "Bdgoa: A bot detection approach for github oauth apps," *Intelligent and Converged Networks*, vol. 4, no. 3, pp. 181–197, 2023.

[206] M. Prudjinski, I. Hadar, and G. Luria, "Exploring the role of team security climate in the implementation of security by design: A case study in the defense sector," *IEEE Transactions on Software Engineering*, 2024.