Building the Landing Zone for Site Reliability Engineering: A Multivocal Literature Review

Luiz Alexandre Costa
UNIRIO
Rio de Janeiro, Brazil
luiz.costa@edu.unirio.br

Awdren Fontão
UFMS
Campo Grande, Brazil
awdren.fontao@ufms.br

Eleni Constantinou
UCY
Nicosia, Cyprus
constantinou.a.eleni@ucy.ac.cy

Rodrigo Pereira dos Santos
UNIRIO
Rio de Janeiro, Brazil
rps@uniriotec.br

Alexander Serebrenik
TU/e
Eindhoven, The Netherlands
a.serebrenik@tue.nl

ABSTRACT

In the era of software-intensive business (SiB), the interdependence between business and software is increasingly prominent. Organizations at the forefront of the digital revolution rely on complex systems to deliver innovative services and products. One of the major challenges is maintaining 24/7 online applications with consistent performance, a task that falls to both the development (Dev) and operations (Ops) teams. Traditionally, Dev focuses on innovation and new features, while Ops ensures the stability of the applications. This division leads to reliability gaps, especially in systems where availability is critical, such as financial services, e-commerce, or mission-critical applications. In such contexts, Site Reliability Engineering (SRE) offers a structured approach that applies Software Engineering (SE) practices to Ops, promotes reliability management through service-level objectives and error budgets, and reduces toil through automation. Given that several emerging practices in SRE occur outside scientific literature, we choose a multivocal approach to capture both theoretical advances and practical experiences. Our work aims to investigate what the multivocal literature is saying about the SRE approach, building the landing zone (a comprehensive foundation) for further studies. Thus, a multivocal literature review (MLR) was performed to identify practices, tools, benefits, and future perspectives from the point of view of researchers and practitioners. We selected 28 studies after applying the review procedures. Based on the results, we categorized SRE practices using the Stacey matrix to guide organizations in prioritizing their adoption. In addition, we propose a Flywheel model to illustrate the iterative and continuous nature of the SRE implementation.

KEYWORDS

software-intensive business, site reliability engineering, multivocal literature review, software quality

1 Introduction

The growing interdependence between business and software defines the era of software-intensive business (SiB) [10]. These organizations are at the forefront of the digital revolution, relying on very complex systems to provide innovative services and products to their customers. Imagine a video streaming company that constantly needs to offer new features to users to stay competitive. At the same time, the backend infrastructure of these platforms must function reliably and consistently, avoiding outages that could alienate users and result in lost revenue.

The need to keep applications online 24/7 with consistent performance and minimal response times has presented an ongoing challenge to the software development (Dev) and operations (Ops) teams. Traditional development and operation practices reside in distinct silos, resulting in gaps that impact the reliability of the services offered [29]. The focus of Dev teams is on innovation, developing new features, and continuously improving applications. These teams work across the software lifecycle, from conception to release, writing code, creating features, testing, and deploying updates. The main concern is to offer new resources quickly and efficiently to meet the demands of the user and the market [13].

On the other hand, Ops teams are tasked with ensuring that applications are running reliably and stably. The focus is on the infrastructure, configuration, monitoring, scalability, and maintenance of the environments where the software runs [13]. Responsibilities include managing servers, networks, databases, and the entire operational environment. Ops teams seek to minimize downtime, monitor system performance, resolve infrastructure incidents, and ensure continued service availability.

In this context, the challenge lies in the need to guarantee the availability and reliability of the applications in a constantly evolving environment [12]. Traditional development and operation structures may not be aligned with contemporary demands. The lack of collaboration between both teams results in systems that are prone to failures in maintaining the availability expected by users [13].

As the demand for continuous innovation and rapid releases of new features drives business evolution and market competitiveness, software reliability becomes the foundation of success [33]. This race for fast delivery often neglects the reliability and stability of the systems [17]. Software reliability failure not only impacts the user experience but also entails costs, from financial losses to damage to the organization's reputation (i.e., CrowdStrike incident caused a massive disruption in approximately 8.5 million systems running Microsoft Windows in July, 2024 [9]).

Considering these challenges, Site Reliability Engineering (SRE) emerges as essential when system reliability and availability are crucial, such as e-commerce platforms, financial services, or mission-critical applications. It is also suitable when organizations already track performance metrics but need rigorous quantitative management through service level objectives and error budgets. SRE also addresses excessive manual operational work by promoting advanced automation, enabling Ops teams to adopt a Software Engineering (SE) mindset. Initially proposed by Google, SRE is an approach that brings SE practices to Ops [5].

Furthermore, the SRE approach should not be seen as a set of isolated activities but rather as an integrated process that spans different phases of the SE product lifecycle. This lifecycle can be described in many ways, and one example is provided in the standard ISO 12207 [24]. This standard defines four high-level process areas: agreement processes, organizational project-enabling processes, project processes, and technical processes, with subprocesses defined for each of them. For example, the technical processes include activities such as implementation, integration, and operations and SRE related activities may occur in any of these processes.

Therefore, the present work aims to investigate what both the scientific and the gray literature (SL and GL) are saying about the SRE approach, building the landing zone (a comprehensive foundation) for further studies. Given that several innovations and emerging practices in SRE occur outside of the SL, we choose a multivocal approach to capture both theoretical advances and practical experiences. Thus, a multivocal literature review (MLR) was performed to identify practices, tools, benefits, and future perspectives from the point of view of researchers and practitioners.

This paper is organized as follows: Section 2 presents the background; Section 3 traces the related work; Section 4 details the research method; Section 5 reports the results; Section 6 describes the discussion; Section 7 presents the threats; and finally, Section 8 concludes with the final remarks.

2 Background

In recent years, the growing complexity of distributed systems, increased reliance on continuous delivery pipelines, and the critical nature of service availability have intensified the need for engineering approaches focused on reliability. Incidents, such as the CrowdStrike [9] illustrate how fragile large-scale infrastructure can become without robust reliability practices. In this context, SRE emerges as a structured response to these demands.

SRE was born at Google in the early 2000s when a team of software engineers led by Ben Treynor Sloss (VP of 24x7 Engineering), was tasked to make Google's already large-scale sites more reliable and scalable. The central problem was finding a balance among innovation and reliability. Before the concept of SRE, Dev teams often did not have direct responsibility for the operation and reliability of their services after they were released.

Thus, we have separate teams for Dev and Ops, which could lead to communication issues, a lack of clear accountability for software reliability, and a divide among goals [6]. SE as a discipline focuses on designing and building rather than operating and maintaining, despite estimates that 40% [21] to 90% [14] of the total costs are incurred after launch. This tension manifests as a natural trade-off: Dev teams prioritize rapid feature delivery, while Ops teams emphasize stability and risk mitigation [25].

Google's motivation for creating the SRE model was to ensure that its systems and services were highly reliable, even as it continued to innovate and scale. With SRE, Google introduced an approach in which software engineers were responsible not only for developing applications but also for ensuring their reliability and operational performance. It was achieved through practices such as automation, proactive monitoring, setting availability goals, and collaborative work between Dev and Ops teams.

By implementing the SRE model, Google was able to significantly reduce system incidents, improve the reliability of its services, and enable the continuous development of new features without compromising stability. SRE is a set of principles and practices that incorporate aspects of SE and apply them to Dev and Ops problems. This approach has gained recognition in the technology industry, and several companies, such as Netflix and Amazon, have started to adopt similar principles to improve the reliability and scalability of their own online services [7].

To operationalize this approach, SRE relies on a structured set of practices that guide its implementation across different organizational contexts. Practices are the specific activities, methods, or techniques used to implement the guidelines that drive the approach to SRE. The SRE practices (PR1 to PR17) can be grouped into five areas to provide a structure for organizing knowledge addressing different challenges and specific aspects, as follows [5, 6]:

1. Metrics, monitoring, and alerts.

- **PR1. Define service level indicators (SLI), service level objectives (SLO), and service level agreements (SLA):** define SLI as quantitative metrics to reflect service performance. SLO are targets set for SLI, while SLA are contractual commitments to customers based on the SLO;
- **PR2. Determine error budgets:** represent the amount of time a service can be outside SLO targets without affecting customer satisfaction. Error budgets help balance the need for innovation with reliability;
- **PR3. Automate recording system metrics:** facilitate continuous and reliable data collection, which is essential for effective monitoring;
- **PR4. Act upon condition detection:** involve rapid response to alerts generated by detected abnormal conditions, minimizing downtime and maintaining quality.

2. Demand forecasting and capacity planning.

- **PR5. Plan for organic growth:** involve planning for the natural growth in service usage based on trend analyses and historical patterns;
- **PR6. Determine inorganic growth:** predict demand spikes unrelated to regular growth, such as seasonal events or marketing campaigns, and plan resources for these;
- **PR7. Optimize efficiency and performance:** improve resource utilization to achieve a balance between cost, efficiency, and performance, avoiding overloading.

3. Change management.

- **PR8. Implement progressive roll outs:** implement changes in stages, starting with a small group and gradually expanding, to minimize risks;
- **PR9. Detect problems quickly and accurately:** indentify issues in new deployments quickly, allowing for interventions before they significantly impact users;
- **PR10. Roll back failed changes safely:** revert unsuccessful changes in a safe and controlled manner to restore normal service operation;

- **PR11. Automate deployment processes:** increase speed and consistency to reduce the risk of manual errors.

4. Emergency response.

- **PR12. Manage incident resolution:** manage incidents includes identification, diagnosis, and resolution of issues, as well as communication with stakeholders;
- **PR13. Define oncall escalating:** define escalation procedures for emergency situations and ensuring problems are addressed by the appropriate team members;
- **PR14. Learn with postmortem reports:** analyze incidents after their resolution to understand their causes and prevent recurrences, emphasizing learning rather than blaming.

5. Culture and toil management.

- **PR15. Develop the blamelessness philosophy:** promote a culture of no blame, where the focus is on identifying and resolving issues, not on punishing individuals;
- **PR16. Decrease operational work:** reduce repetitive operational work (toil) through automation and efficient processes frees up time for more strategic tasks;
- **PR17. Empower talent retention:** foster talent retention by creating a positive development work environment and encouraging innovation and professional growth.

Understanding these practices also helps to clarify how SRE relates to and differs from other approaches, such as DevOps. DevOps is a new way of thinking in the SE that brings together software developers and system administrators [13]. The main goal is to improve the ability to distribute applications and services at high speed. The practice allows for better communication and collaboration between departments to make product delivery faster [16].

The SE landscape required major changes in 2009. The traditional software delivery process could no longer support the rapidly changing business demands, and organizations wanted new features and new revenue streams as quickly as possible, but with stability and reliability requirements (without interruptions) [23]. This limitation is attributed to rigid development lifecycles, lack of cross-functional collaboration, and low levels of process automation, which hindered the ability to balance speed and stability.

DevOps emerged as a cultural shift that values cooperation between Dev and Ops teams [29]. Although SRE shares several conceptual elements with DevOps, such as the emphasis on automation and collaboration, it differs by providing a prescriptive, engineer-led model with defined roles and practices [5].

While DevOps offers broad cultural principles that vary across organizations, SRE addresses these gaps with a consistent set of practices and tools. SRE teams often collaborate with developers from the early stages of system design. Once reliability strategies are established, responsibility for operations can transition to the SRE team. SRE is not a replacement for DevOps, but rather a role-driven implementation that reinforces its collaborative philosophy through technical rigor [5]. Tab. 1 shows the comparison between SRE and DevOps.

3 Related Work

Axelsson and Skoglund [2] addressed the challenges of ensuring software quality in the context of software ecosystems (SECO). The authors performed a systematic mapping study to understand the state of research in this area and proposed a research agenda based on the identified gaps. The study focused on quality assurance practices across the entire software lifecycle in SECO.

Silva et al. [39] investigated faults in microservice-based applications with the aim of supporting activities such as fault tolerance, prevention, detection, and handling. The authors conducted an MLR and proposed a taxonomy comprising 117 faults related to eleven characteristics of the microservices architecture.

Although previous work contributes valuable insights into software quality (e.g., fault taxonomies or research agendas), they do not consolidate actionable practices or roadmaps for operational reliability. In contrast, our study goes further by mapping 24 SRE practices, both established and emerging, categorizing them using the Stacey matrix, and proposing a Flywheel model to guide progressive adoption. This positions our work as a bridge between academic research and practical application of SRE.

4 Research Method

As reported by Kitchenham and Madeyski [27], GL refers to various types of documents produced by governments, academics, businesses, and industry, in both print and electronic formats. These materials are protected by intellectual property rights and are of sufficient quality to be collected by libraries or institutional repositories. However, they are not controlled by commercial publishers, as publishing is not the primary activity.

Systematic Literature Review (SLR) and Systematic Mapping Study (SMS) that include both SL and GL were known as MLR in the early 1990s [32]. We chose this method because the vast majority of practitioners do not publish in academic forums [20, 27]. This means that their voices are not heard if we do not consider the GL in addition to the SL. Understanding their points of view can bring relevant perspectives. We followed the guidelines of Garousi et al. [18] for MLR studies, comprising the phases: i) planning; ii) execution; and iii) results. We also relied on the procedures of Kitchenham & Charters [26].

4.1 Planning

We adopt the Goal–Question–Metric (GQM) approach [3] to define the goal of this study: **analyze** the multivocal literature on SRE **with the purpose of** characterizing **with respect to** practices, tools, benefits, and future perspectives **from the point of view** of SE researchers and practitioners **in the context of** reliability engineering for modern software systems.

4.1.1 Research Questions To address the purpose of the study, the following research questions (RQ) were defined:

RQ1. What does the multivocal literature say about SRE? This question seeks to explore the diversity of perspectives and knowledge present in the multivocal literature on SRE. The analysis of this literature is essential to identify the definitions, new practices, tools, and benefits proposed by different voices in this field that contribute to the enrichment of knowledge.

RQ2. What are the research and practice needs related to SRE? This question aims to identify existing gaps and emerging needs in research in the field of SRE. Understanding these needs should

Table 1: Comparison between SRE and DevOps.

Criteria	SRE	DevOps
Nature	SRE was created with a concrete purpose: to build a set of methods and metrics to improve cooperation and service delivery.	DevOps is a set of philosophies that enable cultural thinking and collaboration across disparate teams.
Goal	SRE includes prescriptive ways to achieve reliability.	DevOps works like a template that guides collaboration to bridge the gap between development and operations.
Focus	SRE's responsibility is primarily focused on improving the availability and reliability of the system.	DevOps focuses on the rapidity of development and delivery while ensuring continuity.
Team structure	The team consists of site reliability engineers who have both operational and development experience.	DevOps team includes testers, developers, cloud architects, security, and SRE engineers.

guide future research and development efforts as well as, promote innovative practices and adaptation to technological changes.

4.1.2 Search Process The research strategy was based on electronic searches in SL and digital libraries. For the SL, searches were performed in the following databases: ACM Digital Library, IEEE Xplore, Science Direct, SpringerLink, and Scopus. These digital libraries were selected since they have been confirmed to cover relevant journals, conferences, and workshops proceedings within SE [15].

For the GL, the research strategy was based on electronic searches on Google Scholar and Google. Such databases were chosen due to their indexing model of other databases, in addition to keeping GL content and having an indexing algorithm based on relevance [18]. The study of Yasin et al. [45] aimed to measure the usage of GL in SLR in SE, and to explore the feasibility of using only Google Scholar to find scholarly articles for academic research.

The authors found that Google Scholar was able to retrieve (96%) of the primary studies from other search sources. Most of the primary studies that were not found in Google Scholar belonged to GL and were found through a simple direct Google search. Therefore, the authors concluded that the combination of Google Scholar and Google can increase the chances of finding the maximum number of primary studies. Kitchenham and Madeyski [27] also mention that the use of Google Scholar is an option for performing MLR. Furthermore, we also identified that several other authors used these same search sources to perform an MLR [18, 19, 42].

4.1.3 Definition of Search Terms We used the following search terms for the search: ("site reliability engineering"). As in most cases, these search terms were refined after three iterations to achieve relevant and complete results. Our objective is to find work related to SRE. To reduce the risk of missing relevant papers in that area, we chose to use the broader search term and then narrowed the scope through manual reviews.

The extraction of data from the studies were performed by two researchers with experience in Empirical Software Engineering (ESE). Several discussion meetings were held to clarify doubts that required double-checking the results. A third researcher with more than 15 years in ESE double-checked the results and ensured the compliance of the final dataset.

4.1.4 Study Selection Criteria For the selection of studies returned in the search, inclusion (IC) and exclusion (EC) criteria were defined and applied to the retrieved studies: IC01 - Study addresses SRE; IC02 - Study is written in English; EC01 - Study is not fully available; EC02 - Study is duplicated; EC03 - Study does not meet any inclusion criteria; EC04 - Study is not primary; and EC05 - Study is not written in English. Studies were included if they met the IC and excluded if they met at least one EC.

4.2 Execution

In this phase, researchers systematically search for, select, and evaluate relevant GL and SL, including data extraction to address the research questions.

The selection process comprised five steps: (1) search execution; (2) removal of duplicates; (3) title, abstract, and keyword evaluation; (4) complete reading; and (5) data extraction. For the SL, we add an extra step: introduction and conclusion evaluation. For the GL, this step was not performed because some studies do not have this structure. The study selection process was conducted in December 2023, and the compiled results were finished in April 2024. Iterative data analysis, coding, and review were performed from May to September 2024, and finally the elaboration and review of the report until December 2024.

In step 1, the SL collection started with 212 studies retrieved from the digital libraries. The retrieved results were stored in the Microsoft Excel tool for the execution of step 2 of the study, when duplicate studies were removed with the support of the Atlas.ti tool and the authors' review. The SL results were then filtered using the steps detailed in Fig. 1, obtaining 11 studies. The complete list (S1 to S11) is informed in Artifact Availability Section.

As stated by Garousi et al. [18], unlike the SL, the question of when to stop the search in MLR is not simple, given the amount of results obtained. Therefore, the authors describe three possible stopping criteria for searches: (i) theoretical saturation, i.e., when no new concept emerges from the search results; (ii) limited effort, i.e., includes only the top "N" search engine results; and (iii) evidence exhaustion, i.e., extraction of all evidence.

For the GL, 1,250 Google Scholar and 1,080,000 Google search results were retrieved. Thus, as also mentioned by Garousi et al. [18], the present study applied limited effort as a stopping criterion, in which the first 100 search results were evaluated for each base (total of 200 studies) and the search continued if the results in the last page would still reveal additional relevant results. The next page after the first 100 records was evaluated, and no new studies were identified to be incorporated into the MLR. It partially corresponds to bounded effort stopping rules augmented by an exhaustive subjective stopping criterion. The same stopping criterion was also observed in the studies of Tom et al. [42] and Garouse & Mantyla [19]. The GL results were filtered using the steps described in Fig. 1, obtaining 17 studies. Fig. 1 also illustrates an overview of the entire process. The complete list (G1 to G17) is informed in Artifact Availability Section.

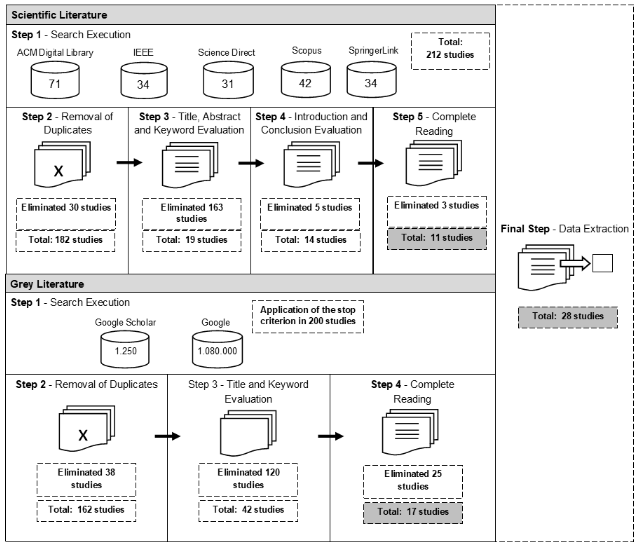

Figure 1: Steps of the MLR selection process.

4.2.1 *Source Quality Assessment* Assessing the quality of sources consists of determining to what extent a source is valid and free from bias. Unlike SL, which normally follows a controlled review and publication process, for processes in GL, quality is more diverse and often more difficult to assess [18]. However, this step was performed in a previous phase, as only those that met the minimum expected quality in terms of source certainty, clarity, details, consistency, and alignment with research focus were included as accepted studies determined by RQ [42]. Results that did not meet a minimum quality threshold were excluded at this stage of the selection process. This limit cannot be predetermined due to the diverse nature of the MLR and assessment and consideration was necessary on a case-by-case basis.

4.2.2 *Data Extraction Procedure* With the aid of the Microsoft Excel and Atlas.ti tools, the extraction process was carried out systematically to answer the RQ through a sheet containing the following fields: (1) study identifier (ID); (2) title; (3) authors; (4) year; (5) definitions; (6) practices; (7) benefits; (8) tools; and (9) future perspectives. After extracting the data, the responses were categorized and grouped to summarize the results.

5 Results

The final phase focuses on documenting the MLR process, findings, and conclusions, ensuring transparency and reproducibility.

5.1 What does the multivocal literature say about SRE?

5.1.1 *Definitions of the term* Initially, we sought definitions of the term SRE according to the following typology: **(i) own:** the study proposes its own definition of SRE, without explicitly attributing it to another source; **(ii) cited:** the study uses an existing definition, explicitly citing the original source; **(iii) interpreted:** the study does not provide an explicit definition, but one can be inferred from the context; and **(iv) none:** the study does not mention the term.

Our findings revealed a variety of studies that address SRE, spanning both SL and GL in some domains, as illustrated in Fig. 2. The horizontal axis represents the publication years, ranging from 2016 to 2024, while the vertical axis captures the organizational domains involved. Among the studies analyzed, 40% (11 of 28) originated from SL and 60% (17 of 28) from GL, highlighting the practical interest in SRE beyond academic environments. The predominance was observed in the software industry domain, accounting for 13 studies (46%). It reflects the origin of SRE practices in large-scale software operations, as well as their role in ensuring system reliability and scalability in cloud-native and service-oriented architectures.

In terms of how the term SRE is defined across the literature, the majority of studies (71%, 20 of 28) cited the seminal work of Beyer et al. [5], reinforcing the centrality of this source in establishing a shared understanding of SRE practices. Additionally, two studies (7%) proposed their own definitions, based on practical experiences in the context of mobile applications. It suggests an attempt to adapt the fundamentals of SRE to specific organizational realities.

Three studies (10%) adopted interpretative definitions, implicitly constructed from the association between SRE and practices such as monitoring, resilience and organizational engineering, evidencing a contextualized appropriation of the concept. Finally, three other studies (10%) did not present any explicit or inferable definition of the term, which may reflect either an assumed familiarity within the audience or a gap in conceptual clarity. This overview consolidates opportunities for an adaptation of the SRE concept. The traceability of the definitions analyzed is informed in Artifact Availability Section for replication purposes.

5.1.2 *New practices* We consider a practice to be new if it has been identified in recent studies from the multivocal literature and is not part of the original set established by Google [6], representing an evolution, adaptation to new contexts, or an innovation within SRE. Thus, our MLR revealed has uncovered a set of innovative practices in SRE, expanding the scope reported in Section 2 (17 practices).

Although Google's pioneering contributions remain key references, the MLR provided an overview of the current state of SRE. Different organizations have different environments, requirements, and particularities. By examining the practices proposed by various sources, we notice adaptations or extensions that make sense in different contexts. These findings provided insights into how to ensure the reliability and scalability of systems in different scenarios.

These new practices not only complement established approaches, but also suggest a continued evolution of the discipline, highlighting the importance of remaining receptive to innovation and adaptation in a dynamic field such as SRE:

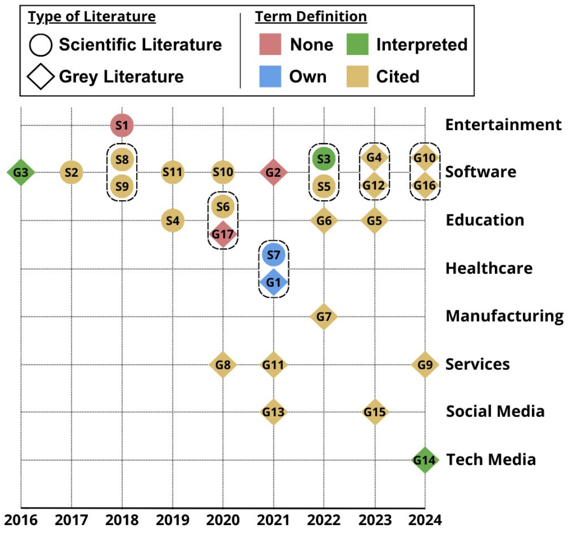

Figure 2: Overview of the MLR.

- **PR18. Measure real user experience (S9):** emphasize the importance of measuring not only server-side latency but also client-side latency. This practice aims to truly understand the user experience, providing a comprehensive assessment of service performance;
- **PR19. Establish chaos engineering (S1, G4):** seek to identify and mitigate risks in complex systems by deliberately introducing failures and disturbances. Experiments, such as server interruptions and injection of corrupted data, allow observation of how the system reacts under extreme conditions, promoting system robustness;
- **PR20. Measure energy efficiency (G5):** acknowledge the importance of sustainability, the practice proposes measuring the energy consumption of servers, data centers, and related infrastructure. The goal is to address the environmental impact crucial for sustainable IT services as defined by green IT policies;
- **PR21. Generate recommendations using artificial intelligence (AI) (G14):** use machine learning to analyze monitoring data and identify patterns is presented as a new effective practice. The machine learning model can generate recommendations to optimize configurations and improve system reliability;
- **PR22. Automate incident response tasks using AI (G14):** using machine learning to identify root causes of problems and implement automatic corrections, highlighting the efficiency of automation in incident response;
- **PR23. Deploy with zero downtime (G17):** ensure that software deployments do not cause interruptions in systems is an essential practice to maintain service continuity. This approach aims to enhance the user experience by avoiding downtime periods; and
- **PR24. Define SRE as a service (G17):** offer an innovative approach to help organizations implement and manage SRE more flexibly and tailored to their specific needs.

5.1.3 Benefits Our work also seek the benefits covered in the selected studies. We pay attention to the tangible results that organizations can obtain when implementing SRE practices. It makes our MLR more applicable and useful to researchers and practitioners who are interested in understanding how SRE can positively impact operations. These findings extend beyond the technical scope, aligning directly with organizational objectives, as follows:

- **S1:** highlights the benefits of chaos engineering, demonstrating the reduction of avoidable interruptions and the strengthening of system resilience. It also emphasizes indirect impacts, such as influencing engineering practices and cultural changes in teams;
- **S3:** emphasizes significant gains from the SRE approach, including increased reliability and resilience of applications, cost savings, and reduction of false alerts or notifications that indicate system problems. The implementation of health check utilities integrated into Platform as a Service (PaaS) solutions is highlighted, as is monitoring critical resources and notifying key stakeholders when thresholds are reached;
- **S6:** focuses on improvements in early detection of system issues and preventing negative impacts on users. It highlights the correlation between SLO and crucial metrics, such as user satisfaction, contributing to the maintenance of high service availability;
- **S7:** underscores the sociotechnical nature of SRE, emphasizing the importance of considering both human and technological aspects in the implementation of reliability practices;
- **S8:** highlights the benefits of SRE in terms of documentation, emphasizing how proper documentation can improve efficiency, reducing Mean Time to Repair (MTTR), and enabling a more scalable approach to service management;
- **G1:** addresses the benefits of applying SRE practices to iOS mobile applications. It highlights improvements in the reliability and quality of applications, including the reduction of failures, optimization of response time, and enhancement of the user experience;
- **G4:** emphasizes the contribution of SRE to improving the user experience, the scalability of systems, and more positive business outcomes. It emphasizes how enhanced reliability and efficiency can lead to increased customer satisfaction and reduced operational costs;
- **G5:** highlights sustainability-related benefits, including the reduction of carbon footprints through the application of SRE practices. It also emphasizes the promotion of energy efficiency in data centers and other IT infrastructures;
- **G6:** focuses on the promotion of a culture of collaboration between development and operations teams, as well as the emphasis on automation for infrastructure and operations management;
- **G7:** explores how SRE enables a proactive approach to identifying and resolving issues before they escalate, contributing to an overall improvement in operational quality;
- **G8:** highlights the automation of operational tasks and the use of SE methods to solve system-related problems. It emphasizes the importance of balancing the launch of new features with maintaining system stability;

- **G9:** emphasizes improvements in the reliability and availability of services, operational efficiency through automation, and the ability to balance stability and innovation needs. It also discusses how SRE can respond more quickly to incidents and implement updates efficiently; and
- **G10:** highlights the collaboration and communication within teams, the efficient management of operational workload, and the promotion of a culture of continuous improvement and learning.

5.1.4 Tools Our findings also reported the specific tools that SRE teams use in their daily lives and how these tools contribute to the success of SRE practices. The tools cited in the selected studies are described below:

- **T1. Hystrix and Chaos Monkey (S1):** highlights the use of these tools at Netflix, demonstrating the implementation of resilience patterns and chaos experiments to strengthen system reliability;
- **T2. Prometheus and Grafana (S3, G13):** mentions the importance of observability, aligning with the daily preferences of SRE engineers for monitoring and analyzing different platform components;
- **T3. Firebase Crashlytics, Android Vitals, Xcode Instruments, and Firebase Performance (S7, G1):** identifies tools used to support SRE practices for monitoring, diagnosing, and improving the reliability of mobile applications. These tools also track and manage failures beyond performance issues;
- **T4. Cloud Monitoring and gRPC. (S10):** focuses on debugging tools and methods, highlighting the crucial role of logs, remote call latency metrics, and real-time monitoring dashboards in ensuring operational reliability;
- **T5. Terraform. (G13):** highlights the importance of infrastructure as a code (IaaC), monitoring, and alerting systems in the effective practice of SRE.

5.1.5 Synthesis of findings from literature Finally, the analysis of our findings in line with software lifecycle processes as outlined by the ISO 12207 standard revealed a significant and comprehensive integration between SRE practice areas and the established SE processes. The interdisciplinary approach of SRE, which combines elements of SE and systems administration, manifests itself throughout the various phases of the software lifecycle. By aligning the research results with the processes defined in ISO 12207, we observed that activities related to SRE may be identified in different subprocesses established by ISO 12207. This analysis provided a structured view, essential for understanding how SRE practices areas are intertwined and contribute to the reliability and scalability of services throughout the entire software lifecycle.

To illustrate the relationship between the processes of ISO 12207 and the SRE areas, we used a Sankey diagram, a tool for visualizing the flow of connections between different categories [36]. Fig. 3 shows which processes have the greatest impact on each SRE area according to the studies. This visualization highlights the nature of SRE by demonstrating that its practice areas do not exist in isolation but are distributed across different software lifecycle phases.

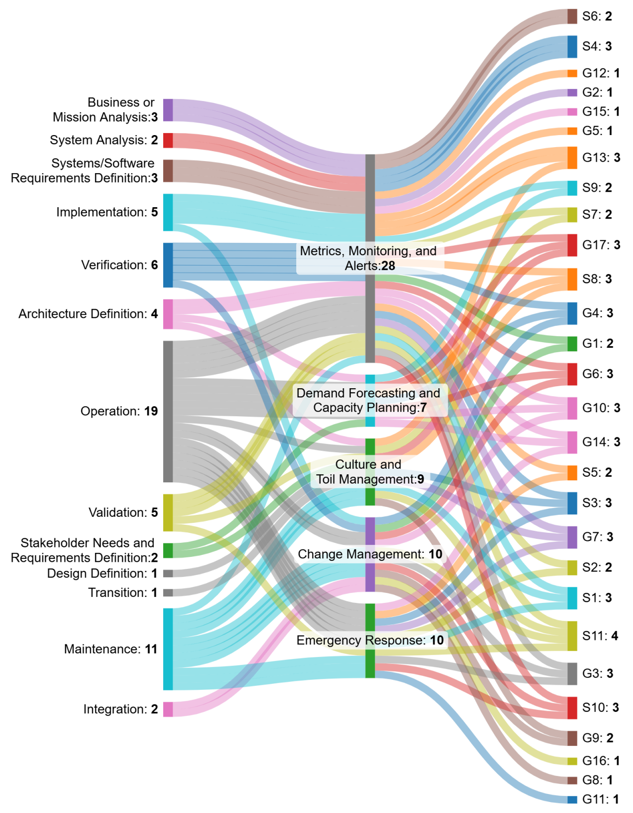

Figure 3: Relationships between ISO 12207 processes, SRE areas, and MLR studies in a Sankey diagram.

We used a coding approach to establish these relationships, inspired by the initial procedures (open and axial coding) for the Grounded Theory [8]. The coding of literature findings, mapped to ISO 12207 software lifecycle processes and linked to corresponding SRE practice areas, is provided as supplementary material in Artifact Availability Section.

In summary, Fig. 3 shows that the predominance of the *Maintenance* and *Operation* processes as the most frequently mapped within the selected studies. It reflects the critical nature of these phases in the software lifecycle to ensure the reliability and scalability of systems. The *Maintenance* process, with 11 mappings, stands out as a stage in which SRE practices are frequently applied to deal with changes, bug fixes, and optimizations, directly contributing to the stability and reliability of software. In turn, the emphasis on the *Operation* process, with 19 mappings, suggests practices related to system availability, highlighting the commitment to ensuring operational resilience, effective monitoring, and proactive incident resolution. These results indicate that software lifecycle processes are in line with the principles of the SRE approach.

5.2 What are the research and practice needs related to SRE?

Although the adoption of SRE has demonstrated significant benefits, such as increased reliability and efficient automation, there are still gaps to be addressed, both in academic research and in practice. Given this scenario, we structured a research agenda that organizes the main opportunities and challenges of SRE throughout the software lifecycle, as defined by ISO 12207. The agenda is also structured around the major lifecycle processes of ISO 12207 as described in Table 2.

6 Discussion

We discuss our results in light of related works and previous studies, triangulating them with the existing SRE literature. This analysis is structured as follows: (i) the prioritization of practices based on their complexity and predictability, using the Stacey matrix; (ii) the evolution of operational reliability as a cyclical and iterative process, represented by the Flywheel model; and (iii) the implications of our findings for researchers and practitioners.

The adoption of SRE in organizations does not occur uniformly. Some practices are quickly implemented due to their maturity and widespread market adoption, while others still face technical, cultural, or organizational challenges [43]. To distinguish these initiatives, we use the concept of *low-hanging fruit*, which is often employed to differentiate easily implementable initiatives from those requiring investments and a longer maturation period [28].

Low-hanging fruit practices provide quick and predictable gains for companies seeking to enhance the reliability of their systems. **PR3** and **PR9** are typical examples, as their implementation can leverage well-established tools such as Prometheus and Grafana [38]. Similarly, **PR1**, **PR12**, **PR13**, and **PR14** follow widely adopted standards, as mentioned in seminal SRE literature [6].

These practices echoed in empirical studies on continuous integration (CI) and quality assurance. For instance, Soares et al. [40] identify CI patterns that facilitate rapid feedback and error detection, reinforcing the value of these SRE practices in enabling predictable, high-quality software delivery pipelines. Furthermore, Axelsson and Skoglund [2] corroborate that quality assurance depends heavily on practices that can cross organizational boundaries with low complexity.

To systematically structure these practices, we use the Stacey matrix [41], a model that classifies initiatives based on the degree of certainty regarding requirements and technology, as illustrated in Fig. 4. This categorization helps identify which practices can be prioritized for quick benefits and which require experimentation and deeper organizational changes before implementation.

Simple practices are those that have a high level of predictability and extensive documentation, making their adoption relatively straightforward. This category is directly related to the *low-hanging fruit* practices, which represent easier starting points for organizations to begin their SRE journey.

However, some emerging practices require further validation or face technical and organizational challenges that hinder their adoption. For instance, **PR22** demands a high degree of integration with existing systems, as well as trust in predictive models [44]. Similarly, **PR20** is a growing concern but still lacks standardized metrics and industry consensus [34].

Complicated practices require greater technical expertise, but their implementation tends to follow a predictable path. **PR2** depend on a detailed understanding of the organization's reliability goals and require accurate SLO monitoring. **PR5** and **PR6** demands predictive analysis based on historical data to define scalable strategies [6]. **PR7** and **PR11** require technical adjustments and careful integration, despite the availability of tools such as ArgoCD [35]. **PR10** relies on robust versioning practices [30]. Additionally, **PR16** and **PR8** offer value in minimizing operational risks, although their adoption may require internal process restructuring [1].

On the other hand, some practices present significant technical and organizational uncertainties, being classified as complex. For instance, **PR4** relies on correlation algorithms and automated responses, requiring more sophisticated integration. **PR18** aims to correlate technical metrics with end-user perception, which is still an evolving are a within SRE [22]. **PR19** is well established in theory, but its practical implementation demands organizational maturity to conduct failure experiments in production environments without compromising operations [4]. Similarly, **PR21** and **PR22** represents an advancement, but depends on trustworthy predictive models and the team's willingness to delegate critical decisions to automated systems. **PR23** involves multiple orchestration techniques and requires a CI environment that is tightly aligned with the organization's infrastructure [16].

Finally, some initiatives are classified as chaotic, as they involve a high degree of uncertainty and require deep cultural change to be successfully implemented. **PR15**, although widely advocated in the SRE literature, demands a cultural shift that many organizations are not yet prepared to undertake [5]. **PR17** is influenced by subjective factors such as organizational climate, sense of purpose, and opportunities for professional growth [11]. **PR20**, while an emerging trend toward sustainable IT infrastructure, still lacks standardized metrics and widespread industry adoption [31]. Lastly, **PR24** remains in an early stage of adoption, with no consolidated models to support large-scale implementation [22].

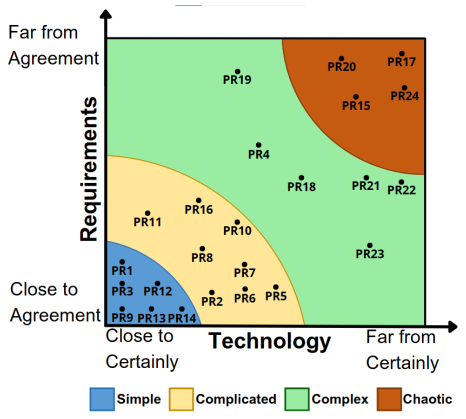

Figure 4: SRE practices mapped to the Stacey matrix.

This categorization of practices within the Stacey matrix provides a strategic roadmap for organizations looking to adopt an SRE approach. By identifying which initiatives can be implemented more directly, companies can structure a planning, balancing quick wins with long-term investments.

However, operational reliability does not evolve linearly. Instead, SRE adoption follows an iterative cycle, where each advancement

Table 2: Overview of the proposed research agenda for SRE.

Life-cycle process	SRE practice area	Research topics	Study
Implementation	Demand Forecasting and Capacity Planning	Implementation of SRE practices in startups is challenging due to the challenging nature of distributing reliable mobile applications at scale.	S7
Implementation	Emergency Response	Challenges in adapting SRE practices, traditionally used in cloud and server environments, to the mobile environment, which presents unique requirements and restrictions.	G1
Implementation	Metrics, Monitoring, and Alerts	The use of robust monitoring metrics when adopting cloud infrastructure, aiming to guarantee the reliability and availability of the system.	G13
Implementation	Metrics, Monitoring, and Alerts	Developing training programs that teach SRE teams about AI techniques.	G14
Operation	Change Management	Contrary to common belief, the most significant information demands for SRE teams are not related to source code changes but rather to understanding changes in the production environment, including system updates and configuration changes. Opportunities for future research lie in effective collaboration between DEV teams in integrating human factors into tool support, i.e., developing tools to better align with how professionals interact and make decisions.	S2
Operation	Emergency Response	Natural language processing to validate response patterns in chats for faster incident response.	S10
Operation	Emergency Response	Barriers and challenges faced by SRE teams during the debugging process, such as the complexity of the systems and the need for rapid problem resolution to minimize the impact on users.	S4
Transition	Emergency Response	Adopting the practice of chaos engineering requires a cultural change within organizations in addition to technical challenges. Reluctance to experiment in production becomes a barrier to adopting this practice.	S1
Transition	Change Management	There are opportunities to use advanced services in areas such as artificial intelligence, machine learning, and big data to manage changes, ensuring that the reliability of the platform is not compromised.	S11
Verification	Metrics, Monitoring, and Alerts	Natural language processing to analyze user engagement data.	S6
Design Definition	Culture and Toil Management	Research opportunity to balance technical, operational, and environmental aspects to meet green IT trends.	G5

creates new opportunities for optimization and learning. To capture this dynamic, we draw inspiration from the Flywheel model, which illustrates how the progressive adoption of SRE can generate a cycle of continuous improvement, as shown in Fig. 5. We propose a customized Flywheel model for SRE, organized in five interconnected steps. Each step reflects a practical layer of maturity and can be supported by practices and tools identified in our study:

1. Metrics and Monitoring: The first step involves establishing reliable indicators to assess system reliability and identify areas for improvement. Practices such as **PR1**, **PR3**, and **PR18** are key starting points because they provide the quantitative foundation required to assess system behavior, user impact, and reliability goals. Tools such as **T2** and **T5** help implement observability and alerting systems, while **T3** offer visibility in mobile contexts. This foundation supports not only awareness of system health but also alignment with user expectations;

2. Incident Response Automation: Once monitoring is in place, organizations can gradually implement scripts to reduce response time and minimize manual intervention. This step includes practices such as **PR4**, **PR12**, **PR13**, **PR14**, and **PR22** supported by the tools **T3** and **T4**. These practices and tools are grouped here because they enable faster and more consistent incident handling, reduce downtime through automation, and promote organizational learning by ensuring that failures generate actionable insights rather than recurring problems;

3. SRE Culture and Toil Reduction: Cultural transformation begins when operational burdens are reduced and teams can focus on engineering work. Practices such as **PR15**, **PR16**, and **PR17** are essential because they create a healthier work environment, increase team engagement, and free engineers to focus on tasks that improve reliability over time. Organizations that promote these values build stronger collaboration between Dev and Ops roles;

4. Feedback and Continuous Improvement: As systems become more observable and stable, feedback loops gain strength. Practices such as **PR5**, **PR6**, **PR7**, and **PR14** reinforce a continuous improvement mindset. **PR8** and **PR10** support adaptive change management. This stage depends heavily on tools that provide actionable insights, such as **T2** and **T4**; and

5. Innovation with AI and Advanced Automation: In mature environments, teams start experimenting with advanced solutions (e.g., machine learning models to prevent failures before they occur). Practices such as **PR19**, **PR20**, and **PR23** illustrate this phase. Tools such as **T1** enable resilience testing and fault injection in production, helping teams validate system behavior under pressure. AI models further enhance this step by predicting risks and automating complex tasks (**PR21** and **PR22**).

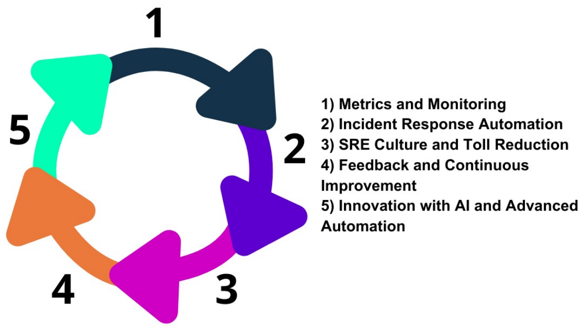
1
2
3
4
5
1) Metrics and Monitoring
2) Incident Response Automation
3) SRE Culture and Toll Reduction
4) Feedback and Continuous Improvement
5) Innovation with AI and Advanced Automation

Figure 5: Iterative cycle for SRE adoption.

The transition between these steps follows an iterative approach, where continuous learning drives incremental improvements. Companies that adopt this evolutionary cycle may build a more reliability infrastructure, reducing operational overhead and enabling teams to focus on innovation. By organizing our findings into this model, we aim to support practitioners in gradually adopting SRE with strategy, while also offering researchers a structured lens to investigate the evolution of reliability practices in software systems.

For **researchers**, the categorization of practices within the Stacey matrix highlights critical gaps in the SRE field. While some practices are widely accepted and well-documented, others, such as AI-driven reliability and energy efficiency measurement still require further study. Additionally, the Flywheel analysis of SRE reinforces the need for further investigations into iterative reliability adoption strategies across organizations of different sizes and domains.

For **practitioners**, prioritizing practices based on complexity and predictability can serve as a strategic roadmap for progressive SRE adoption. Organizations looking to implement SRE can start with simple, well-established practices before advancing to more complex and innovative initiatives. Furthermore, the Flywheel model emphasizes that reliability is not a static goal but an ongoing process of evolution and adaptation, requiring continuous monitoring and adjustments.

The practices and tools identified support not only operational improvements but also cultural and strategic shifts in engineering teams. If the proposed research agenda is progressively addressed, we may witness a future in which SRE becomes more aligned with different organizational contexts. By filling current gaps such as integrating AI into incident response, SRE can evolve into a discipline not limited to large-scale systems.

7 Threats to Validity

Every work has threats that should be addressed together with the results, considering the classification proposed in [37]. Regarding **internal validity**, our concern is the researcher's bias. The results may be affected by the researcher's bias in study selection. To mitigate this threat, while the main coding was done by the two authors of this work, a third author with more than 15 years of SE research and ESE double-checked the results.

The definition of SRE varied between sources, which could influence the interpretation of the results, as well as the proposed practices and tools. The bias introduced, as some focused on established Google practices, while others introduced novel approaches, may be a threat to **external validity**. We also used a stopping criterion for the GL recommended by the MLR guidelines. This criterion may influence the size of the sample, which, in turn, may be a threat to the **conclusion validity**. To mitigate this threat, we described in detail the criteria and procedures adopted.

8 Conclusion

The present work aimed to investigate what the multivocal literature is saying about the SRE approach. Thus, an MLR was performed to identify the practices, benefits, barriers, and future perspectives from the point of view of academia and industry. As a result, 28 studies were selected after applying the review procedures.

Our work revealed convergence in the core definition of the SRE approach, indicating significant influence from the widely accepted source, Google. The analysis of the studies highlighted the predominance of references to this source as a conceptual basis for SRE, indicating a consolidation of the definition of the term SRE in the academic community and among practitioners. In addition, we have seen the introduction of new SRE practices, although the practices established by Google remain widely adopted.

Based on the ISO 12207 standard for software lifecycle processes, the operation process emerged as the most cited, highlighting its centrality in discussions about SRE. The reason can be attributed to the role that it plays in ensuring the ongoing reliability and availability of systems. The operation process, which encompasses SRE practice areas, is essential to the daily activities of SRE teams.

Our work categorized SRE practices using the Stacey matrix, providing a structured approach for organizations to prioritize their adoption based on complexity and predictability. Additionally, we introduced a Flywheel model to illustrate the iterative and continuous nature of SRE implementation. As future work, further empirical validation of the Flywheel model in diverse organizational contexts is needed, along with an in-depth analysis of the impact of AI-driven automation on SRE effectiveness.

ARTIFACT AVAILABILITY

The authors declare that the data supporting the findings of this work are available at https://doi.org/10.5281/zenodo.15204643.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and Grant 88887.955223/2024-00, CNPq (Grant 316510/2023-8), FAPERJ (Grant E-26/204.404/2024), and UNIRIO.

REFERENCES

[1] Sudheer Amgothu. 2024. Innovative CI/CD Pipeline Optimization through Canary and Blue-Green Deployment. *International Journal of Computer Applications* 186, 50 (2024), 1–5.

[2] Jakob Axelsson and Mats Skoglund. 2016. Quality assurance in software ecosystems: A systematic literature mapping and research agenda. *Journal of Systems and Software* 114 (2016), 69–81.

[3] Victor R Basili. 1992. *Software modeling and measurement: the Goal/Question/Metric paradigm.* University of Maryland at College Park.

[4] Ali Basiri, Niosha Behnam, Ruud De Rooij, Lorin Hochstein, Luke Kosewski, Justin Reynolds, and Casey Rosenthal. 2016. Chaos engineering. *IEEE Software* 33, 3 (2016), 35–41.

[5] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site reliability engineering: How Google runs production systems.* " O'Reilly Media, Inc.".

[6] Betsy Beyer, Niall Richard Murphy, David K Rensin, Kent Kawahara, and Stephen Thorne. 2018. *The site reliability workbook: practical ways to implement SRE.* " O'Reilly Media, Inc.".

[7] David N Blank-Edelman. 2018. *Seeking SRE: conversations about running production systems at scale.* " O'Reilly Media, Inc.".

[8] Kathy Charmaz. 2006. *Constructing grounded theory: A practical guide through qualitative analysis.* Sage.

[9] Tarun Kumar Chawdhury. 2024. *Beyond the Falcon: A Generative AI Approach to Robust Endpoint Security.* Technical Report. Technical report.

[10] Felipe Cordeiro, Aline Vasconcelos, Rodrigo Pereira dos Santos, and Patricia Lago. 2024. Investigating Accountability in Business-intensive Systems-of-Systems. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 35–46.

[11] Luiz Alexandre Costa, Edson Dias, Danilo Ribeiro, Awdren Fontão, Gustavo Pinto, Rodrigo Pereira Dos Santos, and Alexander Serebrenik. 2024. An Actionable Framework for Understanding and Improving Talent Retention as a Competitive Advantage in IT Organizations. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 290–291.

[12] Vijay Datla. 2023. Site Reliability Engineering A Modern Approach to Ensuring Cloud Service Uptime and Reliability. *International Journal of Computer Engineering and Technology (IJCET)* 14, 03 (2023), 181–186.

[13] Breno B Nicolau de França, Helvio Jeronimo, and Guilherme Horta Travassos. 2016. Characterizing DevOps by hearing multiple voices. In *Proceedings of the XXX Brazilian Symposium on Software Engineering*. 53–62.

[14] Sayed Mehdi Hejazi Dehaghani and Nafiseh Hajrahimi. 2013. Which factors affect software projects maintenance cost more? *Acta Informatica Medica* 21, 1 (2013), 63.

[15] Tore Dybå and Torgeir Dingsøyr. 2008. Strength of evidence in systematic reviews in software engineering. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 178–187.

[16] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. DevOps. *Ieee Software* 33, 3 (2016), 94–100.

[17] Fernando Vedoin Garcia, Jean Carlo Rossa Hauck, and Adriano Borgatto. 2024. How do Agile Organizations Manage Risks: An Analysis of the State of Practice in Brazil. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 80–91.

[18] Vahid Garousi, Michael Felderer, and Mika V Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology* 106 (2019), 101–121.

[19] Vahid Garousi and Mika V Mäntylä. 2016. When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology* 76 (2016), 92–117.

[20] Robert Glass. 2006. *Software Creativity 2.0.* developer.* Books.

[21] Robert L Glass. 2002. *Facts and fallacies of software engineering.* Addison-Wesley Professional.

[22] Jayanna Hallur. 2024. The Future of SRE: Trends, Tools, and Techniques for the Next Decade. *International Journal of Science and Research (IJSR)* 13, 9 (2024), 1688–1698.

[23] Jez Humble and David Farley. 2010. *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson Education.

[24] ISO/IEC/IEEE. 2017. ISO/IEC/IEEE International Standard - Systems and software engineering–Software life cycle processes–Part 2: Relation and mapping between ISO/IEC/IEEE 12207:2017 and ISO/IEC 12207:2008. *ISO/IEC/IEEE 12207-2:2020(E)* (2017), 1–278. https://doi.org/10.1109/IEEESTD.2020.9238529

[25] Muhammad Shoaib Khan, Abudul Wahid Khan, Faheem Khan, Muhammad Adnan Khan, and Taeg Keun Whangbo. 2022. Critical challenges to adopt DevOps culture in software organizations: A systematic review. *IEEE Access* 10 (2022), 14339–14349.

[26] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report EBSE-2007-01* (2007).

[27] Barbara Kitchenham, Lech Madeyski, and David Budgen. 2022. How should software engineering secondary studies include grey material? *IEEE Transactions on Software Engineering* 49, 2 (2022), 872–882.

[28] Eriks Klotins, Tony Gorschek, and Magnus Wilson. 2023. Continuous software engineering: Introducing an industry readiness model. *IEEE Software* 40, 4 (2023), 77–87.

[29] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A survey of DevOps concepts and challenges. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–35.

[30] Ajay Mahimkar, Carlos Eduardo de Andrade, Rakesh Sinha, and Giritharan Rana. 2021. A composition framework for change management. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 788–806.

[31] Javier Mancebo, Félix García, and Coral Calero. 2021. A process for analysing the energy efficiency of software. *Information and Software Technology* 134 (2021), 106560.

[32] Rodney T Ogawa and Betty Malen. 1991. Towards rigor in reviews of multivocal literatures: Applying the exploratory case study method. *Review of educational research* 61, 3 (1991), 265–286.

[33] Shravan Pargaonkar. 2023. Cultivating Software Excellence: The Intersection of Code Quality and Dynamic Analysis in Contemporary Software Development within the Field of Software Quality Engineering. *International Journal of Science and Research (IJSR)* 12, 9 (2023), 10–13.

[34] Gustavo Pinto and Fernando Castor. 2017. Energy efficiency: a new concern for application software developers. *Commun. ACM* 60, 12 (2017), 68–75.

[35] Saikiran Reddy, A Catharine, J Jeslin Shanthamalar, et al. 2024. Efficient Application Deployment: GitOps for Faster and Secure CI/CD Cycles. In *2024 International Conference on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE)*. IEEE, 1–7.

[36] Patrick Riehmann, Manfred Hanfler, and Bernd Froehlich. 2005. Interactive sankey diagrams. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*. IEEE, 233–240.

[37] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons.

[38] Vivek Sharma. 2022. Managing multi-cloud deployments on kubernetes with istio, prometheus and grafana. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1. IEEE, 525–529.

[39] Francisco Silva, Valéria Lelli, Ismayle Santos, and Rossana Andrade. 2022. Towards a fault taxonomy for microservices-based applications. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*. 247–256.

[40] Eliezio Soares, Gustavo Sizilio, Jadson Santos, Daniel Alencar Da Costa, and Uirá Kulesza. 2022. The effects of continuous integration on software development: a systematic literature review. *Empirical Software Engineering* 27, 3 (2022), 78.

[41] Ralph D Stacey. 2007. *Strategic management and organisational dynamics: The challenge of complexity to ways of thinking about organisations.* Pearson education.

[42] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516.

[43] Vladyslav Ukis. 2022. *Establishing SRE Foundations: A Step-by-Step Guide to Introducing Site Reliability Engineering in Software Delivery Organizations.* Addison-Wesley Professional.

[44] Joseph Uzoma, Olakunle Falana, Callistus Obunadike, Kunle Oloyede, and Echezona Obunadike. 2023. Using artificial intelligence for automated incidence response in cybersecurity. *International Journal of Information Technology (IJIT)* 1, 4 (2023).

[45] Affan Yasin, Rubia Fatima, Lijie Wen, Wasif Afzal, Muhammad Azhar, and Richard Torkar. 2020. On using grey literature and google scholar in systematic literature reviews in software engineering. *IEEE Access* 8 (2020), 36226–36243.