applied sciences

MDPI

Review

Best Practices Evidenced for Software Development Based on DevOps and Scrum: A Literature Review

Manuel Pastrana [1,*], Hugo Ordoñez [2,*], Carlos Alberto Cobos-Lozada [2] and Mirna Muñoz [3]

[1] Faculty of Engineering, Institución Universitaria Antonio José Camacho, Cali 760046, Valle del Cauca, Colombia
[2] Faculty of Electronic Engineering and Telecommunications, Cauca University, Popayán 190001, Cauca, Colombia; ccobos@unicauca.edu.co
[3] Software Engineering Unit, Research Centre in Mathematics, Zacatecas 98160, Mexico; mirna.munoz@cimat.mx
* Correspondence: mapastrana@admon.uniajc.edu.co (M.P.); hugoordonez@unicauca.edu.co (H.O.)

Abstract: Very small entities (VSEs) often lack resources and specialized expertise and thus face unique challenges in the adoption of frameworks such as Scrum and DevOps. While these frameworks offer the potential to improve efficiency and quality, their successful integration into VSEs remains a complex undertaking. To address this issue, this study conducts a literature review of studies indexed in Scopus and Web of Science, aiming to consolidate existing knowledge and identify key success factors for the joint adoption of Scrum and DevOps in VSEs. A total of 111 peer-reviewed papers published between 2014 and 2023 were analyzed using a thematic analysis. The findings reveal five critical factors for Scrum (actively involving the stakeholders, early and continuous feedback, transparent communication channel, constant measure quality, and quality deliverable), five essential DevOps practices (versioning, pipeline automation, creation of a collaborative culture, continuous integration, and automated testing), and three common practices that facilitate their integration (early feedback, collaborative culture, and continuous improvement). This research provides actionable insights for VSE practitioners seeking to implement Scrum and DevOps effectively. It underscores the requirement for tailored guidelines and practical models to support the successful adoption of these frameworks, ultimately improving software development processes and business outcomes in these resource-constrained organizations.

Keywords: Scrum; DevOps; very small entities; VSEs; best practices; software development; literature review

Academic Editors: Vito Conforti, Alexander Barkalov and Pedro Couto

Received: 5 February 2025
Revised: 20 March 2025
Accepted: 21 March 2025
Published: 13 May 2025

Citation: Pastrana, M.; Ordoñez, H.; Cobos-Lozada, C.A.; Muñoz, M. Best Practices Evidenced for Software Development Based on DevOps and Scrum: A Literature Review. *Appl. Sci.* **2025**, *15*, 5421. https://doi.org/10.3390/app15105421

1. Introduction

Agile frameworks, particularly Scrum, have become an increasingly popular approach to software development, promising enhanced flexibility, faster delivery cycles, and improved product quality [1]. Complementing Agile frameworks, Development and Operations (DevOps) methodologies aim to streamline the software development lifecycle through automation, collaboration, and continuous feedback [2]. While the benefits of Scrum and DevOps are well documented, their successful adoption can be particularly challenging for very small entities (VSEs) [3,4].

VSEs are typically defined as organizations with fewer than 25 employees [1], often operating with limited resources, restricted budgets, and a lack of specialized expertise [5]. These constraints can make it difficult for VSEs to implement the organizational changes,

invest in the necessary tooling, and provide the adequate training required for effective Scrum and DevOps adoption. Furthermore, the existing literature on Scrum and DevOps often focuses on larger organizations, potentially overlooking the specific requirements and challenges of VSEs [3]. Therefore, a consolidated understanding of the specific practices and successful factors that enable VSEs to leverage Scrum and DevOps effectively is required [5,6].

To address this gap, this study presents a systematic literature review aimed at identifying and consolidating the fundamental practices and success factors associated with Scrum and DevOps implementation in VSEs. This research seeks to answer the following questions: **RQ1: What factors are fundamental to success in implementing Scrum? RQ2: What factors are fundamental to success in implementing DevOps? RQ3: What are the common DevOps practices documented in the literature? RQ4: What practices are common for both Scrum and DevOps?** By synthesizing the existing knowledge, this review aims to provide actionable insights for VSE practitioners seeking to improve their software development processes, which according to [7] is something fundamental for those who want to implement practices suggested by agile frameworks.

The main contributions of this study include the identification of critical success factors for implementing Scrum and DevOps, the main practice of DevOps, the common practices of both frameworks, the challenges and mitigation strategies for implementation, and the empirical and quantitative benefits of Scrum and DevOps. These findings highlight the importance of tailoring Scrum and DevOps practices to the specific context of VSEs.

This review is essential because it provides a consolidated and contextualized understanding of Scrum and DevOps for VSEs, enabling them to make informed decisions and adopt practices that are most likely to lead to success. By addressing the specific challenges and opportunities faced by VSEs, this research contributes to the broader goal of improving software development practices and business outcomes in these organizations.

The remainder of this study is structured as follows. Section 2 describes the research methods used in this systematic review. Section 3 presents the results of our analysis. Section 4 discusses the implications of our findings, and Section 5 concludes with recommendations for future research and practice.

2. Research Method

This study follows the guidelines established by the authors of [8–10] for clearly defining research questions, developing a search strategy, defining inclusion and exclusion criteria, assessing studies' quality, and extracting data. While inspired by Kitchenham's methodology, we did not apply all aspects of her approach but rather adapted it to suit the specific research context and objectives of this study.

As shown in Figure 1, the literature review followed a process comprising five key stages: planning, quality assessment, data extraction, and analysis process, and performing the review. The process implemented at each stage is detailed below.

2.1. Planning

The planning step defines the scope of the review and the research questions. Additionally, the search strategy defined the database selection, the keywords, and the exclusion and inclusion criteria.

2.1.1. Scope of the Review

The literature review aims to identify and consolidate the fundamental practices proposed by Scrum and DevOps, the success factors associated with their implementation,

the challenges and the strategies to solve them, and the possible benefits to be achieved with the implementation of both frameworks.

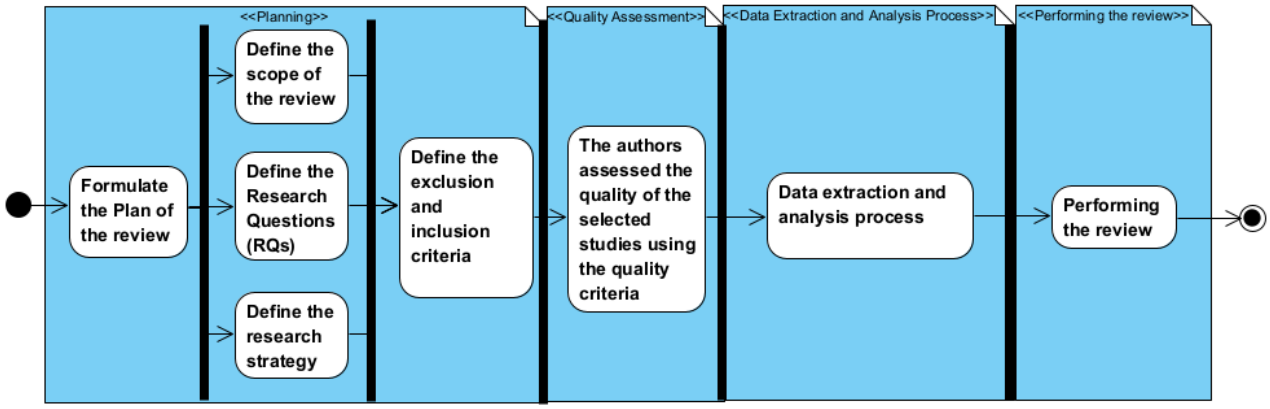

Figure 1. Methodology workflow.

2.1.2. Research Questions

To achieve the review scope, the following research questions were asked:

- **RQ1:** What factors are fundamental to success in implementing Scrum?
- **RQ2:** What factors are fundamental to success in implementing DevOps?
- **RQ3:** What are the common DevOps practices documented in the literature?
- **RQ4:** What practices are common for both Scrum and DevOps?

2.1.3. Search Strategy

To ensure adequate coverage of the relevant literature, a search strategy was designed, drawing upon the methodology outlined by Kitchenham et al. [8]. Specifically, the approach was adopted for string construction, which involves identifying core terms from pilot studies, refining the search string iteratively using a quasi-gold standard of known relevant papers, and combining Boolean operators to balance sensitivity (recall) and specificity (precision). It provides an auditable process endorsed for systematic reviews in software engineering, ensuring minimized selection bias and maximizing reproducibility.

The final strings were executed in Scopus and Web of Science (WoS), which are two of the most recognized electronic research databases. These databases were selected because they index high-quality publications from a wide range of sources, including key databases in software engineering such as IEEE Xplore, Elsevier, and ACM Digital Library. Given their rigorous quality filters and multidisciplinary scope, Scopus and WoS minimize noise from low-impact sources, ensuring that the publications identified are highly relevant and impactful for this study. The search strings were formulated through an iterative process described below.

1. **Initial identification of key terms:** based on research questions and common terms in preliminary studies (e.g., “Scrum”, “DevOps”, and “best practices”);
2. **Validation and refinement:** the search string was validated using a test set of known studies and refined to maximize retrieval of relevant documents;

The final strings were executed in **Scopus** and **Web of Science (WoS)**, which are two of the most recognized electronic research databases. These databases were used to identify publications within their platforms using a quality filter.

- **Scopus:** ALL ((Scrum OR Agile OR agility) AND (DevOps OR “development and operation”) AND (“best practice”);

- **WoS** = (ALL = ((Scrum OR Agile OR agility) AND (DevOps OR "development and operation") AND ("best practices")));

In addition, the **snowballing** technique was applied according to the guidelines of the study in [9], identifying the relevant references in the included studies.

2.1.4. Inclusion and Exclusion Criteria

To ensure the quality and relevance of the selected documents, the following criteria were defined.

Exclusion criteria:

1. Papers not written in English;
2. Presentations, tutorials, and anecdotal opinions;
3. Papers that do not specifically address software development practices;
4. Papers published before 2014 or after 2023.

Inclusion criteria:

1. Peer-reviewed papers published in English;
2. Papers from journals, books, and conference proceedings.

2.2. Quality Assessment

During this step, the authors assessed the methodological quality of the selected studies using quality criteria adopted from Kitchenham et al. [11]. Studies that did not meet the minimum predefined quality threshold were excluded from the review. Next, the exclusion and inclusion criteria were defined.

1. Are the review's inclusion and exclusion criteria described and appropriate?
2. Is the literature search likely to have covered all relevant studies?
3. Did the reviewers assess the quality/validity of the included studies?
4. Were the basic data/studies adequately described?

A process was employed to assess the quality of the selected papers, ensuring consistency and reliability in the evaluation. This process, referred to as the "consensus and independent validation" approach, involved the following steps:

1. To minimize bias in the initial quality assessment, three reviewers were randomly assigned to each study from the pool of reviewers;
2. Each reviewer independently assessed the quality of the paper based on the predefined quality criteria adopted from Kitchenham et al. [11]. They answered the specific quality questions described above for each criterion and provided justifications for their evaluations;
3. The three researchers then compared their assessments, discussed any discrepancies, and collaboratively arrived at a consensus score for each quality criterion;
4. To further enhance reliability, an additional independent reviewer, blinded to the initial assessments, reviewed all quality assessments, providing their answers and justifications. Any discrepancies between the consensus scores and the independent reviewer's scores were resolved through discussion with the initial three reviewers.

2.3. Papers Data Extraction and Analysis Process

Data extraction was carried out based on the study in [9] using a data extraction form that included the following:

1. Identify the article (author, title, abstract, year of publication, category (Scrum, DevOps, or both), contribution, advantage, disadvantage, gap, and DOI/Link);
2. Determine the relevant thematic categories to answer the research questions.

The data extraction and analysis process consisted of randomly assigning the papers identified in the search to three researchers who then read the literature and answered the research questions based on the review findings. The three researchers compared and discussed the results and reached a consensus. At the same time, a fourth researcher conducted the same activity with all of the papers. In the end, the consensus of the first group was discussed with the results of the fourth researcher, and a definitive consensus was reached to provide the result. Figure 2 presents the data extraction and analysis process described above.

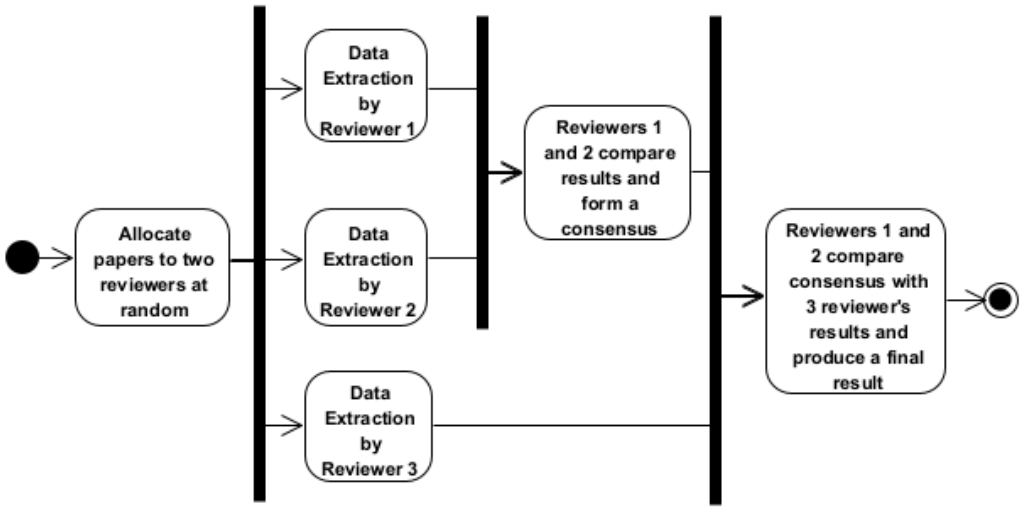

Figure 2. Data extraction and analysis process flowchart.

2.4. Performing the Review

This section presents the search results and the outcome of the filtering process according to the proposed strategy, as well as the search inclusion and exclusion criteria applied.

The selection process was carried out in several stages.

1. **Initial search:** retrieval of 378 documents using the search strings;
2. **Filtering by title and abstract:** exclusion of irrelevant and duplicate documents (267 removed);
3. **Full-text review:** Each article was assigned to four researchers randomly for review, ensuring disagreements were discussed and resolved collaboratively. The process resulted in the selection of 111 papers aligned with the research questions;
4. **Snowballing:** The literature search iteratively expanded through forward snowballing according to the process suggested by Wohlin [10], aiming to uncover publications that cited foundational works in the domain and introduced significant conceptual, methodological, or practical information not already represented in the existing review corpus. By this means, the inclusion of two additional documents identified through the references in the selected studies [7,12] was achieved.

3. Results

Once the planning is defined, the review is executed as described in the method section. Table 1 shows the number of papers found per database and the included and excluded results.

Additionally, Table 2 presents the distribution of papers across the research questions. As RQ1 (Scrum success factors) has fewer papers than RQ3 (DevOps practices), this indicates a potentially greater current research focus on DevOps than on Scrum. This distribution reflects the current research landscape, and the higher interest in DevOps

could be attributed to several factors. First, Scrum is extensively documented. Its core principles, advantages, and challenges are relatively well-established in the literature. As a well-defined framework, Scrum may offer fewer novel research avenues compared with DevOps. Second, DevOps complements Agile frameworks by proposing fundamental practices for software development, presenting ongoing research challenges in exploring the integration of these practices with frameworks such as Scrum. Identifying common practices between both frameworks also creates opportunities to enhance their synergistic application and facilitate broader adoption within organizations. It is important to note that a single article may contribute data relevant to only one research question.

Table 1. Search results by database.

Database	Found	Include	Exclude
Scopus	374	108	266
Web of Science	2	1	1
Snowballing	2	2	0
Total	378	111	267

Table 2. The research question classification results.

Research Question	Results
RQ1	[5,7,12–30]
RQ2	[31–118]
RQ3	[68–118]
RQ4	[119–121]

An analysis of the publication dates of the results obtained, which can be seen in Figure 3, reflects the number of publications in the databases selected after applying the search strings, revealing a trend that began in 2014, with two relevant studies. In 2015, the number of publications increased to three. Subsequently, in 2016, the number continued to increase, with nine relevant studies, and then increased to 12 results in 2017, 15 in 2018, and 18 in 2019. The number of publications peaked in 2020 with 28 results. The community's interest declined in 2021, with only 12 relevant studies, and in 2022, with five results. Finally, 2023 revealed eight results.

The observed decline in the number of relevant publications from 2022 to 2024 (Figure 2) can be analyzed in two ways. First, while the initial surge in research interest, peaking in 2020, may be attributed to increased industry adoption of Scrum and DevOps methodologies, a subsequent stabilization or shift in research focus could explain the recent trend. Second, the observed decline could be influenced by publication lags. The peer-review and publication process for high-quality research can take several months or even years. Therefore, research conducted in 2022 and 2023 may not yet be fully reflected in the published literature available in Scopus and Web of Science (WoS).

Once the literary review was completed, the results were classified by answering each research question by applying the data extraction and analysis process described previously. The classification of the results by research question summarizes the number of relevant papers found in Table 2, which indicates the questions and the number of papers found by each database. For the first question, after filtering the information according to the inclusion, exclusion, and quality criteria, a total of 21 relevant studies were identified, including two studies that were added through the snowballing technique.

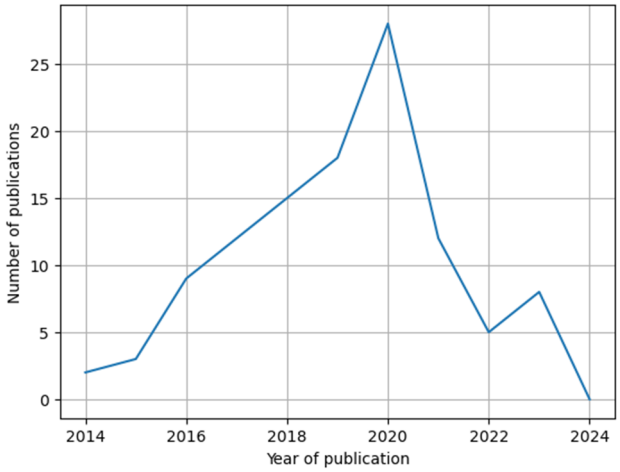

Figure 3. The number of papers per year found in the literature review.

3.1. What Factors Are Fundamental to Success in Implementing Scrum?

To answer this question, twenty-one documents were identified, of which two were added via snowball [7,12]. According to the literature, Table 3 lists the key factors of successful Scrum implementation, identifying the most recurrent in this relationship.

Table 3. Literature found about fundamental Scrum implementation factors.

Practice/Reference	[5]	[11]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[12]	[29]	[30]
Support from management		X																			
Deliver continuous information	X	X					X									X					
Constant quality measurement (concurrent testing)	X	X				X	X				X	X								X	X
Actively involve the stakeholders	X	X				X	X	X		X	X	X			X	X		X		X	X
Transparent communication channels	X	X		X		X	X	X			X			X	X		X	X			
The organizational culture			X						X												
Incremental, iterative development						X	X	X		X						X				X	
Early and continuous feedback	X			X	X	X	X	X		X	X	X	X			X				X	X
Continuous improvement						X	X					X								X	
Shorter delivery cycles					X			X		X	X	X									
Decomposition of the work on smaller lists prioritized by the business value they deliver					X	X								X		X					
Delivery of commitments in time periods no longer than the duration of a sprint											X	X								X	
Collaborative work								X								X	X				
Quality deliverables		X			X			X	X		X	X	X	X							
Self-management															X						X

The findings allow us to identify five practices as critical factors for successfully implementing Scrum. Next, they are listed in the order most referenced in the literature.

1. **Actively involve the stakeholders**. This factor, cited in 61.9% of the analyzed studies, emphasizes the critical role of engaging stakeholders throughout the project. It is highlighted as essential for aligning project objectives with stakeholder expectations and ensuring that deliverables meet their requirements. Key studies, including those by the authors of [5,11,16–18,20–22,25,26,28–30], consistently recognize stakeholder involvement as a cornerstone of project success;
2. **Early and continuous feedback**. Cited in 61.9% of the papers analyzed, this factor underscores the importance of providing timely and ongoing feedback throughout the project lifecycle. It ensures that adjustments can be made promptly, reducing risks and enhancing the quality of deliverables. Studies such as those

by [5,14–18,20–23,26,29,30] highlight this practice as fundamental for aligning project outcomes with evolving requirements and stakeholder expectations;

3. **Transparent communication channel**. Cited in 52.4% of the papers analyzed, this factor emphasizes the importance of maintaining clear, open, and effective communication throughout the project. Transparent communication ensures that all team members and stakeholders are aligned, minimizing misunderstandings and facilitating prompt decision-making. Key studies by the authors of [5,11,14,16–18,21,24,25,27,28] consistently highlight transparent communication as a critical component for project success;
4. **Constant quality measurement or concurrent testing**. Cited in 38.1% of the analyzed papers, this factor highlights the critical role of continuous quality measurement and testing throughout the project. By integrating testing into every phase, teams can identify and address issues early, ensuring that the final product meets the required standards. Studies by the authors of [5,11,16,17,21,22,29,30] emphasize the value of ongoing quality assessment and testing in maintaining high standards and reducing defects;
5. **Delivering high-quality outputs**. A total of 38.1% of the papers emphasize the importance of consistently delivering high-quality outputs throughout the project lifecycle. Achieving quality deliverables ensures that the project meets stakeholder expectations, enhances customer satisfaction, and reduces the requirement for rework. Key studies by the authors of [11,15,18,19,21–24] consistently highlight the central role of quality in successful project execution and delivery.

Although they are not among the top five practices identified in the review, it is essential to remark that incremental, iterative development and shorter delivery cycles are also considered successful factors. However, due to the focus of the papers, not all of them mention this. Additionally, Table 3 indicates that, although there is documentation on the experiences of some companies in which various elements that guarantee success have been measured, the industry does not have a transparent model of what practices to use according to the specific characterization of companies and their cultural conditions, which turns this process into a trial-and-error model that can be costly while the ideal configuration of practices to implement is determined.

3.2. What Factors Are Fundamental to Success in Implementing DevOps?

Thirty-seven papers were reviewed to address the following research question: What factors are fundamental to success in implementing DevOps? Tables 4 and 5 summarize the fundamental factors for successful DevOps implementation, as identified in the literature, highlighting the most frequently mentioned ones. Notably, topics such as operational platform construction, infrastructure as code, and infrastructure management are less mentioned in the findings, so they are only mentioned in Table 4. The findings identify five practices presented below in order of frequency in the literature.

1. **Versioning**. Identified in 59% of the papers analyzed, this practice is highlighted as critical for managing changes in code and ensuring traceability throughout the software development process. By enabling teams to track modifications, resolve conflicts, and maintain a history of changes, versioning contributes to the stability and reliability of projects. Key studies, including those by the authors of [37–42,44,46–51,53–57,63–65], emphasize its foundational role in effective DevOps and Agile practices;
2. **Automation of pipeline operations**. Mentioned in 57% of the papers, this practice is recognized as a cornerstone of DevOps implementation, enabling streamlined and efficient software delivery. Automating pipeline operations, such as code integration,

testing, and deployment, reduces manual effort, minimizes errors, and accelerates development cycles. Key studies, including [37–42,44,46–51,53–57,63–65], highlight how this automation fosters reliability and scalability in software development processes;

3. **Creation of a collaborative culture**. Cited in 51% of the papers, establishing a collaborative culture is fundamental for fostering effective teamwork and ensuring smooth integration between development and operations teams. This practice encourages open communication, knowledge sharing, and mutual support, enhancing problem-solving and innovation. Studies by the authors of [31–35,42–45,50,53–55,59,63–67] underline the importance of nurturing this culture to improve collaboration and achieve successful outcomes in DevOps environments;
4. **Continuous integration**. Mentioned in 49% of the papers, this practice is recognized as key for ensuring high-quality software's seamless and rapid delivery. By continuously integrating code changes into a shared repository and running automated tests, teams can detect errors early, streamline the development process, and maintain consistent software quality. Studies by the authors of [37–42,44–47,49,50,56,57,63–65,67] remark on the role of continuous integration in improving the efficiency and reliability of the software development process;
5. **Automated testing**. Mentioned in 49% of the papers, this practice is highlighted as crucial for improving software quality and accelerating development. By automating repetitive testing tasks, teams can quickly identify defects, ensure consistent test coverage, and maintain high-quality standards throughout the development lifecycle. Key studies, including those by the authors of [37–42,44–47,49,50,56,57,60,64–67], emphasize the role of automated testing in enhancing efficiency, reliability, and scalability within development pipelines.

Table 4. Literature found about fundamental DevOps implementation factors.

Practice/Reference	[31]	[32]	[33]	[34]	[35]	[36]	[37]	[38]	[39]	[40]	[41]	[42]	[43]	[44]	[45]	[46]	[47]	[48]	[49]
Step-by-step involvement in the organizational culture.	X	X	X	X								X	X		X				
Understanding the practices and their benefits.	X	X	X	X				X				X	X	X	X				
Creation of a collaborative culture.	X	X	X	X	X							X	X	X	X				
Gradually implementing the practices that allow a culture of continuous improvement.	X	X	X	X															
Constant communication.					X														
Global understanding of the responsibilities of the entire team.					X								X		X				
Continuous generation of information.						X													
Constant feedback.						X	X	X				X		X					
Transparent sharing of the quality of the project's progress.						X	X												
Involving security.									X										
Automation of pipeline operations.							X	X	X	X	X	X		X		X	X	X	X
Continuous integration.							X	X	X	X	X	X		X	X	X	X		X
Continuous deployment.							X	X	X	X	X	X		X		X	X		
Automated testing.							X	X	X	X	X	X		X	X	X	X		X
Monitoring.							X	X	X	X	X	X		X	X		X		
Continuous delivery.							X	X	X	X	X	X		X					
Strategic indicators, factors, and metrics to evaluate associated with tools by models such as Qrapids.																			X
Versioning.							X	X	X	X	X	X		X	X	X	X	X	X

Additionally, it is essential to highlight the importance of understanding the practices and their benefits, as well as the factors of continuous deployment and monitoring.

However, these factors are not included in the top five practices and are next in terms of mention, with a recurrence percentage greater than 38%.

Table 5. The literature found about fundamental DevOps implementation factors.

Practice/Reference	[50]	[51]	[52]	[53]	[54]	[55]	[56]	[57]	[58]	[59]	[60]	[61]	[62]	[63]	[65]	[65]	[66]	[67]
Step-by-step involvement in the organizational culture.		X											X		X	X	X	X
Understanding the practices and their benefits.		X												X	X	X	X	X
Creation of a collaborative culture.	X			X	X	X				X				X	X	X	X	X
Gradually implementing the practices that allow a culture of continuous improvement.							X								X	X	X	
Constant communication.				X	X	X									X			
Global understanding of the responsibilities of the entire team.																		
Continuous generation of information.	X			X	X	X					X							
Constant feedback.	X	X														X		X
Transparent sharing of the quality of the project's progress.				X	X	X							X		X	X		X
Involving security.						X			X	X		X			X			
Construction of operational platforms.							X	X										
Infrastructure as code.							X	X										
Infrastructure management.							X	X							X			
Automation of pipeline operations.	X	X		X	X	X	X	X						X	X	X		
Continuous integration.	X						X	X						X	X	X		X
Continuous deployment.	X						X	X						X	X	X		
Automated testing.	X						X	X			X				X	X		X
Monitoring.	X						X	X							X	X		
Continuous delivery.	X						X	X					X		X	X		
Strategic indicators, factors, and metrics to evaluate associated with tools by models such as Qrapids.																		
Versioning.	X			X	X	X	X	X						X	X	X		X
Actively involve the user.														X	X	X		X

3.3. What Are the Common DevOps Practices Documented in the Literature?

Fifty papers were found for the following question: What are the common DevOps practices documented in the literature? Tables 6–8 present the practices proposed by DevOps for the software development process according to the literature, identifying which are the most mentioned. It is necessary to highlight that the literature shown in Table 6 mentions continuous monitoring, dependency management, continuous planning, code standards, collaborative culture, continuous benchmarking, continuous delivery, continuous information, and continuous feedback.

The results show a list of 10 practices associated with DevOps, which are organized as follows from the most referenced in the literature to the least:

1. **Continuous integration**. Cited in 72.3% of the analyzed papers, continuous integration (CI) is widely recognized as a fundamental practice in modern software development processes. CI involves frequently integrating code changes into a shared repository, followed by automated testing to detect and resolve issues early. This approach enhances software quality, reduces integration risks, and accelerates delivery cycles. Key studies, including [68–70,73–81,84,87–94,96,99–106,109–111,114,115,117,118], emphasize the role of CI in fostering reliable and scalable software delivery pipelines, making it a cornerstone of DevOps and Agile frameworks;
2. **Versioning**. Cited in 68.1% of the analyzed papers, versioning is recognized as a fundamental practice for managing code changes and ensuring traceability throughout the software development lifecycle. By maintaining a system-

atic history of modifications, versioning allows teams to track changes, facilitate collaboration, resolve conflicts, and improve code stability. Studies such as [68–70,73–81,84,87–94,96,99–106,109–111,114,115,117,118] highlight versioning as a fundamental practice for improving the software development process and implementing other practices, such as continuous integration and deployment pipelines, reducing risks associated with code integration, and enhancing software quality;

3. **Continuous deployment**. Cited in 68.1% of the papers analyzed, continuous deployment (CD) is highlighted as a key practice for automating software releases to production environments. CD reduces manual intervention, accelerates delivery cycles, and enhances software reliability by seamlessly deploying tested and validated code changes. Studies such as [68–70,73–81,84,87–91,93,94,96,99–101,103–106,109–111,115,117,118] underscore the importance of CD in enabling a continuous delivery pipeline, ensuring that high-quality software is consistently delivered with minimal disruption;
4. **Cloud computing**. As mentioned in 34% of the papers analyzed, cloud computing is recognized as a key enabler for modern software development and deployment processes. By providing scalable infrastructure, on-demand resources, and flexible computing environments, cloud computing supports the efficient implementation of DevOps practices, such as continuous integration, continuous deployment, and automated testing. Studies including [68–81,102,113] remark on the role of cloud computing in reducing infrastructure costs, improving resource utilization, and enabling rapid provisioning of development and production environments. Cloud computing is, thus, a foundational technology for achieving scalability, flexibility, and efficiency in software delivery pipelines;
5. **Infrastructure as code (IaC).** As mentioned in 34% of the analyzed papers, infrastructure as code (IaC) is recognized as a critical practice for automating the management and provisioning of infrastructure through code. IaC enhances consistency, reduces manual errors, and accelerates deployment processes by treating infrastructure configurations as versioned and executable scripts. Key studies, including [72–81,90,91,96,102,103], focus on how IaC supports integrating DevOps practices, facilitating other practices such as continuous deployment, and ensuring that infrastructure changes are systematically tested and documented;
6. **Automated testing**. As mentioned in 31.9% of the papers analyzed, automated testing is highlighted as a key practice for ensuring software quality and accelerating the software development process. By automating the test execution, teams can efficiently detect defects early, improve test coverage, and reduce manual effort, ultimately leading to faster and more reliable software releases. Studies such as [81,87,88,90,91,99–101,103,105–107,109–112,116] underscore the importance of automated testing in maintaining software stability and enabling teams to deliver high-quality products efficiently;
7. **Unit testing**. As mentioned in 29.8% of the papers, unit testing is highlighted as essential for ensuring the correctness and reliability of individual components or functions within the software. By isolating and testing small units of code, unit testing enables the early detection of errors, supports better code design, and improves the overall maintainability of the system. Key studies, including [81,87–91] and [99–101,103,104,106,110–112,116–118], emphasize unit testing as a key practice for achieving high software quality and facilitating the early identification of issues, which contributes to more efficient and reliable development processes;
8. **Static code analysis (SCA)**. As mentioned in 29.8% of the papers analyzed, unit testing is recognized as a critical practice for validating the functionality of individual components or modules in software development. By isolating and test-

ing small code units, teams can identify defects early, ensure code reliability, and support maintainability throughout the development lifecycle. Key studies, including [68,81,84,87,88,94,99,104–107,109–111], emphasize its role in supporting continuous integration and reducing the likelihood of errors propagating through later stages of development;

9. **Continuous monitoring**. As mentioned in 27.7% of the papers analyzed, continuous monitoring is identified as a key practice for maintaining system performance, reliability, and security throughout the software lifecycle. Continuous monitoring enables teams to detect anomalies early, reduce downtime, and ensure consistent service delivery by providing real-time insights into application behavior, infrastructure health, and potential issues. Studies such as [88,90,91,93,94,99,102,103,105,109–112,114,116] mentioned continuous monitoring as essential for supporting DevOps practices, enabling proactive maintenance, and ensuring the stability of software systems in production environments;
10. **Continuous improvement**. As mentioned in 21.3% of the papers analyzed, continuous improvement is recognized as a vital practice for enhancing processes, products, and services over time. By fostering a culture of ongoing refinement and iteration, teams can adapt to changing requirements, optimize workflows, and enhance overall performance. Studies such as [82–84,90–92,94,97,99,104,107,114–118] emphasize the role of continuous improvement in supporting Agile and DevOps practices, ensuring that teams can deliver higher quality and more efficient solutions through iterative learning and adaptation.

The review shows that the base practices for DevOps are focused on continuous integration, versioning, deployment, automating, and chaining their use through a pipeline. However, although there is a common perspective about the most used practices, the diversity of tools to implement them is very wide, and thus, various configurations and models are suggested in the literature. This trend suggests that there is still no proven guide or standard to define an implementation model for each of these practices or to explain how they can be integrated to create a DevOps-based quality environment determined by the organization's size, indicating a significant research opportunity.

Table 6. The literature found on practices proposed by DevOps for software development.

Practice/Reference	[68]	[69]	[70]	[71]	[72]	[73]	[74]	[75]	[76]	[77]	[78]	[79]	[80]	[81]	[82]	[83]	[84]	[85]	[86]
Versioning	X	X	X			X	X	X	X	X	X	X	X	X			X		
CI	X	X	X			X	X	X	X	X	X	X	X	X			X		
CD	X	X	X			X	X	X	X	X	X	X	X	X			X		
Microservices		X	X	X															
Cloud computing	X	X	X	X	X	X	X	X	X	X	X	X	X	X					
SCA	X													X			X		
IaC					X	X	X	X	X	X	X	X	X	X			X		
DataOps															X	X			
Continuous improvement															X	X	X		
Involving security																	X	X	
TDD																			X
BDD																			X
Unit testing														X					
Integration testing														X					
Automated testing														X					

Table 7. The literature found on practices proposed by DevOps for software development. Cont. imp. = continuous improvement; Cont. monitoring = continuous monitoring; Dependency mgt. = dependency management; Cont. BM. = continuous benchmarking; and Cont. info. = continuous information.

Practice/Reference	[87]	[88]	[89]	[90]	[91]	[92]	[93]	[94]	[95]	[96]	[97]	[99]	[100]	[101]	[102]	[103]	[104]	[105]	[106]
Versioning	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X
CI	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X
CD	X	X	X	X	X		X	X		X		X	X	X		X	X	X	X
Microservices																			
Cloud comp.															X				
SCA	X	X						X				X					X	X	X
IaC				X	X					X					X				
DataOps																			
Cont. imp.				X	X	X		X			X	X					X		
Involving security									X			X				X		X	
TDD													X	X					
BDD																			
Unit testing	X	X	X	X	X							X	X	X		X	X		X
Integration testing	X	X		X	X							X	X	X		X			X
Automated testing	X	X		X	X							X	X	X		X		X	X
Cont. monitoring		X		X	X		X	X				X			X	X		X	
Dependency mgt.												X			X				
Continuous planning												X			X				
Code standard																		X	
Collaborative culture																	X	X	
Cont. BM.														X					
Continuous delivery										X									
Continuous info.										X								X	
Continuous feedback								X	X		X							X	

Table 8. The literature found on practices proposed by DevOps for software development.

Practice/Reference	[107]	[108]	[109]	[110]	[111]	[112]	[113]	[114]	[115]	[116]	[117]	[118]
Versioning			X	X	X			X	X		X	X
CI			X	X	X				X		X	X
CD			X	X	X				X		X	X
Microservices												
Cloud computing							X					
SCA	X		X	X	X							
IaC							X					
DataOps												
Continuous improvement	X							X	X	X	X	X
Involving security												
TDD						X						
BDD												
Unit testing				X	X	X				X	X	X
Integration testing						X				X		
Automated testing	X		X	X	X	X				X		
Continuous monitoring			X	X	X	X		X		X		
Dependency management												
Continuous planning		X										
Code standard												
Collaborative culture		X	X	X	X							
Continuous benchmarking												
Continuous delivery				X								
Continuous information					X			X		X		
Continuous feedback		X	X		X			X		X		

3.4. Challenges and Mitigation Strategies for Integrating Scrum and DevOps in VSEs

Integrating Scrum and DevOps within very small entities (VSEs) presents a multifaceted challenge due to their inherent constraints. Their limited resources, specific organizational structures, and often informal operational practices [1] create unique obstacles to successful integration [14,17,18,31,64,66,110,115,119].

This section, grounded in the literature review, examines the common problems encountered during this integration and proposes viable solutions tailored to the VSE context.

3.4.1. Cultural Challenges

Adopting frameworks for software development is not easy, signaling that implementing Scrum and DevOps requires overcoming significant resistance and cultural inertia. These challenges underscore the critical importance of strong organizational commitment, a willingness to embrace change, and the fostering of a collaborative culture that transcends traditional silos [67,82,90,99,107,110]. In many VSEs, established communication patterns, decision-making, and responsibility are often informal and deeply ingrained. Introducing Scrum and DevOps necessitates a shift towards more structured, transparent, and collaborative approaches, which can be met with skepticism or resistance from team members accustomed to more traditional methods. Thus, VSEs should prioritize fostering an environment that encourages synchronization across all areas involved. This means actively promoting cross-functional communication, shared responsibility, and a collective understanding of project goals. Management must champion these values, providing clear direction and consistent support for teams as they adapt to new ways of working. Building trust, celebrating successes, and openly addressing concerns are essential for nurturing a culture that embraces change and collaboration, ultimately leading to enhanced effectiveness and the seamless integration of Scrum and DevOps practices.

3.4.2. Technical Limitations

Some authors, such as those of [90,99,107,110], propose changing how development and operations teams work based on the significant automation of repetitive tasks. When combined with limited financial resources, the skills gap makes it difficult for VSEs to invest in the robust DevOps tooling and infrastructure required for seamless automation, continuous integration, and continuous delivery [14,79]. Implementing and managing these tools can quickly become overwhelming, especially for small teams lacking specialized expertise. As a viable solution to address this, VSEs must prioritize what they require to implement in a staggered manner and then implement cloud-based services to overcome the automation problems [36]. According to the authors of [37], by leveraging cloud-based DevOps platforms, VSEs can access a wide range of pre-configured tools and managed services that significantly reduce the burden of infrastructure management and maintenance and implement the main practices found in the literature as continuous integration (CI), versioning, continuous deployment (CD), cloud computing, IaC, automated testing, unit testing, static code analysis (SCA), continuous monitoring, and continuous improvement.

Cloud solutions also offer the flexibility to scale resources up or down as required, aligning costs with actual usage and reducing upfront investment requirements. Therefore, cloud-based services can empower VSEs to overcome technical limitations and implement essential DevOps practices without straining their limited resources [36]. Another possible solution is to use infrastructure as code (IaC), according to [72–81,90,91,96,102,103].

Instead of managing infrastructure manually, VSEs can define servers, networks, and other components as code to automate provisioning, simplify management, and ensure consistency. Thus, IaC tools facilitate rapid deployment, scaling, and cost optimization to adapt quickly to changing project requirements. Storing IaC configurations in version

control systems improves collaboration, reduces errors, and allows for easy rollback to previous configurations. Additionally, IaC empowers VSEs to streamline their infrastructure operations, improve reliability, and reduce the requirement for specialized infrastructure skills, making it an option for a successful Scrum and DevOps implementation. This presents a powerful solution to overcome technical limitations and resource constraints.

3.4.3. Process-Related Challenges

Integrating Scrum and DevOps within VSEs also requires careful consideration of process-related challenges. While Scrum offers a flexible framework, the work of the authors of [4] rightly points out that not all Agile practices work well in all company configurations. This means that VSEs cannot simply adopt Scrum and DevOps processes wholesale; they must adapt and tailor them to fit their specific context, team dynamics, and project requirements in a staggered manner.

Implementing Scrum and DevOps successfully requires a deep understanding of the company's business model, culture, and team capabilities. Imposing rigid processes that are not well suited to the organization can lead to frustration, inefficiency, and project failure. To overcome these challenges, VSEs should prioritize flexibility and adaptability. This often involves implementing short deliveries that allow the team to react to unexpected situations while providing continuous feedback. This enables early adjustments and reduces significant risks. Additionally, by embracing iterative development cycles and fostering open and transparent communication, VSEs can continuously refine their processes to optimize efficiency and deliver value more effectively. This leads to the implementation of practices associated with communication and collaboration to share the responsibilities between team members and deliver high-quality outputs.

3.4.4. Resource Challenges

Limited financial resources present a significant hurdle for very small entities (VSEs) seeking to implement Scrum and DevOps practices effectively. As highlighted in the introduction of this work, a VSE has a limited budget for reinvestment because its cash flow depends on the number of clients with active projects. This financial constraint severely restricts the ability of VSEs to invest in the necessary tooling, training, and infrastructure for robust Scrum and DevOps workflows. Quality control implementation is mostly manual or absent, thus experimenting with different models and practices is costly and unfeasible. These resource constraints often result in an empirical and expensive software development process characterized by high reprocessing and unpredictable quality, making it difficult for VSEs to compete effectively in the market. Their limited ability to reinvest in process improvements, automation tools, and skilled personnel creates a cycle of inefficiency and limits their growth potential.

VSEs must adopt pragmatic and cost-effective approaches to Scrum and DevOps adoption to address these resource constraints. One way to achieve this is to focus on cloud tools to implement practices across the software development process. VSEs can significantly reduce upfront costs and ongoing configuration and maintenance expenses by leveraging cloud-based services and open-source tools. Cloud platforms offer scalable pricing models that allow VSEs to pay only for the resources they consume, avoiding the large capital investments associated with traditional on-premises infrastructure. Furthermore, open-source tools provide access to various software development and automation capabilities without incurring licensing fees. By strategically combining these approaches, VSEs can effectively manage their limited resources and build a solid foundation for Scrum and DevOps success.

Table 9 summarizes the above and shows the relationship between the challenges and practices proposed by Scrum and DevOps, according to the literature review, that companies can implement.

Table 9. Relationship between challenges and solutions proposed by Scrum and DevOps practices.

Challenges	Scrum Practices	DevOps Practices
Cultural challenges	Actively involved stakeholders Transparent communication channels	Creation of a collaborative culture Continuous improvement
Technical limitations	Constant quality measurement or concurrent testing Early and continuous feedback	Versioning Automation of pipeline operations Cloud computing IaC
Process-related challenges	Early and continuous feedback Delivering high-quality outputs	Continuous improvement Continuous integration Continuous deployment Automated testing Unit testing
Resource challenges	Transparent communication channels	Cloud computing Continuous integration Continuous deployment

3.5. What Practices Are Common for Both Scrum and DevOps?

Integrating Scrum and DevOps necessitates a holistic approach that leverages the practices common to both frameworks. Based on the literature review, the three practices that appear common to both frameworks are actively involved stakeholders, early and continuous feedback, and transparent communication channels, with each element contributing uniquely to the success and streamlining of the joint implementation according to the authors of [119–121].

Table 10 presents the following common practices proposed by Scrum and DevOps for the software development process according to the literature:

1. Actively involving the stakeholders;
2. Early and continuous feedback;
3. Transparent communication channels.

Table 10. The literature found on common practices for Scrum and DevOps.

Practice/Reference	[119]	[120]	[121]
Actively involving the stakeholders	X	X	X
Early and continuous feedback	X	X	X
Transparent communication channels	X	X	X

The review of the selected studies reveals that actively involving stakeholders is a paramount success factor in Scrum and DevOps implementations. This prevalence underscores the critical role of engaging stakeholders throughout the project lifecycle, not merely as passive recipients of deliverables but as active participants in shaping project outcomes. It is essential to align project objectives with stakeholder expectations and ensure that project deliverables demonstrably meet articulated requirements, mitigating the risk of misalignment and promoting project success.

These findings are further substantiated by the recurring emphasis on communication and collaboration, indicating a consensus within the research community regarding the importance of stakeholder integration in achieving the intended benefits of implementing Scrum and DevOps frameworks. In summary, stakeholders must be actively engaged to make better software.

Additionally, early and continuous feedback underscores the importance of providing timely and ongoing feedback throughout the project lifecycle. This practice ensures that adjustments can be made promptly, reducing risks and enhancing the quality of deliverables.

The literature review results suggest that, in a Scrum and DevOps context, early feedback loops (e.g., from automated testing, continuous integration, and direct stakeholder involvement) are not merely about identifying defects; they represent a mechanism for ensuring that the software being built aligns with the actual requirements of the business and the end-users.

Frequent integration and testing, which are hallmarks of DevOps, provide rapid insights into the impact of code changes. This allows developers to catch integration issues and performance bottlenecks before they escalate into major problems.

Furthermore, incorporating feedback from stakeholders throughout the development process, which is a key principle of Scrum, ensures that the software meets their evolving requirements and expectations. This proactive approach minimizes the risk of producing the wrong product, reducing costly rework and ensuring that the final project result delivers maximum value. It allows stakeholders to play a part in project delivery. For this reason, integrating stakeholder feedback into each iteration and demonstrating working software frequently promotes transparency, alignment, and, ultimately, greater project success.

Equally important, in both frameworks, namely Scrum and DevOps, transparent communication channels are not merely about relaying information. Rather, this practice fosters a shared understanding of project goals, progress, and challenges among all team members and stakeholders. Transparent communication is crucial because it makes stakeholders and development teams more aligned and closer to each other. Ultimately, transparent communication fosters trust and collaboration within the team, leading to faster problem-solving, improved decision-making, and a greater sense of shared ownership for the project's success. Moreover, transparent communication creates a supportive environment for continuous improvement through an open discussion of successes and failures, where the team can identify areas for improvement and implement changes to optimize processes.

Additionally, some relevant findings to highlight are discussed in the following sections.

3.5.1. About the Agile Software Development Approach

In 2020 [119], the authors deepened a previous work [120], where they identified that the industry is rapidly transforming its Agile software development process by nurturing it with DevOps thinking and approaches. In both works, the authors reflect that Agile thinking, specifically Scrum, proposes a high level of communication and interaction between its various roles and likewise between the multiple roles and project stakeholders, allowing early feedback on the progress of the project, discovering opportunities for improvement, and adjusting the various artifacts that document the project. Although each role has specific functions, such as maintaining the product vision and increasing the value of the delivery, for example, in the case of the product owner, this does not limit the roles for constantly sharing information with the teams that can facilitate continuous improvement. Thus, continuous information comes from various events, such as the daily Scrum meeting, Scrum board, sprint-planning meeting, sprint review, and sprint retrospective, which enriches project progress management, risk management, and coordination for meeting sprint goals. While this has helped companies improve development processes by increasing quality, improving delivery, improving production start-up times, and increasing opportunities for improvement, it still needs to catch up to the results required by the industry. The main reason is that many manual steps are still required to improve the software development process.

3.5.2. Suggestions for the Implementation of Both Frameworks

To solve the above, the work of [119,120] suggested using Scrum with DevOps, proposing a transitional change in three phases: first, to understand and apply Agile thinking; second, to understand how continuous integration works (versioning, continuous deployment, unit testing, integration, and automated, among others), its tools, and how to apply them; and finally, an approach to development and operations that should be organized based on continuous delivery.

It is essential to highlight that the authors of the study in [119] mentioned that DevOps increased quality controls compared with Agile models, adding continuous reviews that generate highly relevant information for decision-making. This approach is called Smart because it has a clear purpose: to reduce the gap between development and operations. The Smart approach seeks to perform and control two operations that traditionally have been working independently, and its results are easily perceived by all involved. In addition, its process is based on automation, and all of the resulting information is put in an application context to improve the results of the final deliverable.

3.5.3. The Future of Software Development

As indicated in [121], the current trend is to adopt the best practices proposed by frameworks such as Scrum, DevOps, Lean, and Kanban, among others, and use them to build their configurations within companies. The purpose of this approach is to develop a model to follow in order to identify what to do and how to do it according to a specific type of company. Likewise, Saltz et al. [122] state that, although Scrum is still the most used framework, with 56% adoption in the industry, companies are looking to achieve automated cycles of continuous delivery to be more competitive against the competition. Therefore, DevOps is an appropriate extension of Scrum-based development processes. It is essential to highlight that the common elements that make them compatible are collaborative work, continuous feedback, controlled deliveries with constant quality measurements, and information transparency. Additionally, the current lack of research on the subject presents an opportunity for those who wish to develop this area of knowledge.

Based on these three references, understanding practices and their benefits, creating a collaborative culture, and having a clear and transparent communication channel are the common practices for Scrum and DevOps.

Furthermore, when comparing the results obtained from the first research question (what factors are fundamental to success in the implementation of Scrum?) and the second (what factors are fundamental to success in the implementation of DevOps?), the following practices were identified as common success factors in the implementation of Scrum and DevOps: actively involve the stakeholders, early and continuous feedback, transparent communication channels, and deliver continuous information.

3.6. Empirical Evidence for Scrum and DevOps Benefits

While the previous sections identified key factors and practices for successful Scrum and DevOps implementation, this section examines the empirical evidence supporting these claims, drawing upon the case studies, experimental validations, and quantitative data reported in the reviewed literature.

Several studies found in the review provide real-world examples of the benefits of adopting Scrum and DevOps. The authors of [46] present a case study examining the productivity gains of DevOps adoption by an IT team. While the abstract does not detail specific metrics, the study suggests a positive correlation between DevOps implementation and team productivity. Similarly, the authors of [49] offer a theory, model, and case study

of DevOps adoption, providing a more in-depth analysis of the factors contributing to successful implementation.

Additionally, the impact of specific DevOps practices is explored by the authors of [94], who characterize the influence of continuous integration based on empirical results from over 250 open-source and proprietary projects. This large-scale study provides valuable insights into the benefits of continuous integration on software quality and delivery speed. In addition, the authors of [100] further investigate this topic, presenting a large-scale empirical study on the impact of continuous integration on other software development practices. Their findings highlight the interconnectedness of various DevOps practices and the synergistic effects of their combined implementation.

On the other hand, the authors of [118] examine the shift-left testing approach in DevOps. While the abstract focuses on the benefits, challenges, and best practices, the study likely contains empirical data related to the impact of shift-left testing on software quality and defect reduction. Additionally, the authors of [112] select DevOps' best test practices that are significantly related to those presented in this study, which provides a solid foundation and validation of the selection of DevOps practices.

However, it is important to acknowledge that the existing empirical evidence has some limitations. The focus of the study in [46] is the impact of DevOps alone. In addition, there is a lack of specific data on the implementation of DevOps in VSEs. Therefore, future research should focus on conducting rigorous quantitative studies in VSEs to address these gaps and provide more conclusive evidence of the benefits of Scrum and DevOps in these organizations.

3.7. Quantitative Evidence of Scrum and DevOps Benefits

The literature review provides evidence supporting the benefits of Scrum and DevOps adoption in the industry, especially for improving software development productivity, reducing delivery times, enhancing software quality, and optimizing operational efficiency. The quantitative results of multiple studies demonstrating the impact of adopting some of its practices are detailed below.

3.7.1. Productivity Gains and Delivery Speed

The integration of DevOps and Agile methodologies has consistently shown improvements in software development efficiency according to the authors of [45], who conducted a case study on an IT team transitioning to DevOps by merging development and operations teams. The work analyzed six DevOps capabilities by measuring team productivity before and after the transition. The results found a 20% increase in productivity and a 30% reduction in deployment time.

Likewise, the authors of [48] examined DevOps adoption in various real-world companies using Grounded Theory and a case study at a government institution. Their research emphasized that automation alone is insufficient for DevOps success and that collaboration plays a crucial role. The same result was found using Scrum. Therefore, actively involving stakeholders, creating transparent communication channels, and creating a collaborative culture will help companies achieve continuous improvement. While exact quantitative measures were not provided, the case study reported a 25% faster release cycle and a 40% improvement in incident resolution time.

Additionally, the authors of [20] demonstrated substantial benefits in real-world settings by examining Portuguese IT companies migrating from waterfall to Agile, reporting a 25% reduction in time to market and an 18% increase in project success rates, indicating that constant quality measurement or concurrent testing and early and continuous feedback proposed from Scrum is necessary; moreover, versioning and automation of pipeline opera-

tions from DevOps are also required. Adding to their findings, the authors of [29] studied Agile adoption in China's global software development industry, demonstrating a 20% improvement in project delivery speed and a 30% increase in customer satisfaction scores.

These findings align with industry-wide observations, where continuous feedback loops, automated deployment, and improved cross-team collaboration have been highlighted as key drivers of efficiency gains [40].

3.7.2. Software Quality Improvements

Scrum, as a key Agile framework, promotes continuous delivery, iterative development, and strong collaboration, according to the author of [4]. His findings highlight that Scrum enables teams to maintain steady velocity, improve predictability, and enhance product quality through shorter feedback loops and frequent releases.

DevOps also contribute significantly to improved software quality. The authors of [93] analyzed over 250 open-source and proprietary projects, finding that CI adoption led to a 12% reduction in defect density and a 15% increase in developer productivity. Reinforcing these findings, the authors of [99] conducted a large-scale empirical study on GitHub projects and found that versioning and CI adoption increased the number of merged pull requests by 22% and reduced pull request resolution time by 18%, leading to better code quality and defect management, pointing to early and continuous feedback to generate deliverable quality.

Additionally, automated testing practices further enhance software quality according to the authors of [111], who demonstrated that integrating automated testing frameworks improved test execution speed by 35% and increased defect detection rates by 18% compared with manual testing, as demonstrated using automated tests.

3.7.3. Cost and Risk Reduction Through Early Testing and Security Measures

The implementation of shift-left testing, which is the inclusion of testing as early as possible in the development process using practices such as versioning, automated build through CI, unit testing, and SCA (static code analysis), has been found to significantly reduce defect costs. The authors of [117] found that early defect detection improved by 40%, and bug-fix costs were reduced by 25%, demonstrating the financial benefits of proactive quality assurance.

From a security perspective, the authors of [106] investigated security concerns and best practices for automating software deployment processes, finding that adopting secure automation practices reduced security vulnerabilities by 30% and cut response time to security incidents by 50%. These results are aligned with the requirement for integrated security practices in CI/CD pipelines to prevent security breaches late in the development process [40].

3.7.4. Scaling Agile in Large Organizations

The authors of [30] conducted a mixed-methods study on an Agile transformation in a mission-critical government project, reporting a 15% decrease in development cycle time and a 10% improvement in team collaboration.

In addition to this, the authors of [4] provided a detailed practitioner-oriented account of Scrum adoption, emphasizing how cross-functional teams, self-organization, and iterative releases contribute to improved software quality and faster project delivery. Scrum's structured approach to backlog management, sprint planning, and frequent retrospectives ensures continuous learning and process improvement. The above findings are summarized in Table 11.

Table 11. Scrum and DevOps implementation benefits.

Benefits	Improvement Observed	Supporting Studies
Scrum adoption	Actively involved stakeholders. Transparent communication channels. Increased team collaboration and improved predictability. Creation of a collaborative culture. Continuous improvement.	[4]
DevOps adoption	Constant quality measurement or concurrent testing. Early and continuous feedback. Productivity increased by 20%, and deployment time decreased by 30%.	[45]
Faster release cycles	Time to market decreased by 25%, and incident resolution time decreased by 40%. Quality deliverable.	[29,48]
Continuous integration	Early and continuous feedback. Quality deliverable. Time to market decreased by 25%, and incident resolution time decreased by 40%.	[40,93,99]
Automated testing	Transparent communication channels. Test execution speed increased by 35%, and defect detection increased by 18%.	[111]
Shift-left testing	Early defect detection increased by 40%, and bug-fix costs increased by 25%.	[117]
Security automation	Security vulnerabilities decreased by 30%, and response time decreased by 50%.	[106]
Agile transformation	The development cycle decreased by 25%, and project success rates increased by 18%.	[20,30]

4. Discussion

4.1. Context of the Results

This RSL provides a broad perspective on the fundamental practices and success factors related to Scrum and DevOps, as well as their points of convergence. In the case of Scrum, five critical success factors were identified, including active stakeholder engagement, transparent communication, and early and continuous feedback. The literature review recognized these practices as essential to ensuring quality deliverables and fostering a collaborative culture in Agile projects.

Regarding DevOps, the findings underscore the value of technical practices, such as version control, continuous integration, and pipeline operations automation, which have proven instrumental in accelerating development cycles and improving software quality. In addition, the complementary nature of DevOps concerning Scrum reinforces the idea that its integration can significantly enhance development processes in VSEs, organizations with limited resources, and simple structures that require solutions adapted to their context.

A key finding was the identification of three common practices between both frameworks: early feedback, transparent communication, and creating a collaborative culture. These practices make it easier to integrate Scrum and DevOps and help overcome traditional barriers, such as the disconnect between development and operations teams. This is especially relevant for VSEs, where the lack of resources to experiment with multiple configurations makes these common practices strategic starting points for joint adoption.

The results of this study also highlight that, although Scrum is well-documented and its benefits are widely recognized in the literature, its implementation still needs to present challenges related to the absence of clear guidelines on the application of its principles in different contexts. On the other hand, with a more technical focus, DevOps offers a robust set of practices that can be complemented by Scrum's organizational flexibility to address specific problems, such as reducing rework and improving the predictability of deliverables.

Finally, the analysis reveals a growing trend in the literature towards DevOps research, driven by its ability to optimize Agile frameworks and its emphasis on automation. This trend suggests an opportunity to explore innovative combinations of practices between both frameworks, intending to provide more comprehensive solutions to the specific challenges of VSEs.

4.2. Comparison with Previous Work

In line with previous research, Scrum is confirmed to be a well-documented framework, with extensive guides on its advantages and challenges. However, this review aims to identify a gap in the literature related to the integration of Scrum with DevOps. Studies such as those by the authors of [44] highlight the benefits of DevOps in terms of automation and software quality but rarely explore how these practices can be harmonized with Scrum, especially in the context of VSEs.

This work also complements the findings of the authors of [30], who identify resistance to change and lack of knowledge as key barriers to DevOps adoption. The results of our literature review expand this perspective by identifying factors such as version control, continuous integration, and collaborative culture as pillars to overcome these challenges.

4.3. Interpretation of the Results

Identifying five critical success factors for Scrum, such as active stakeholder participation and transparent communication, reflects the requirement for approaches focused on collaboration and managing expectations. In contrast, key DevOps practices, such as pipeline automation and version control, underscore the importance of technical tools to optimize processes.

The convergence in common practices, such as early feedback and collaborative culture, suggests that integrating both frameworks is feasible and beneficial. However, effective implementation requires a contextualized approach that considers the characteristics of VSEs, such as their small size and limited resources.

Additionally, the lack of a standardized approach for implementing the practices of both frameworks, coupled with the challenges posed by the wide variety of practices and tools available and the understanding of the common obstacles faced during their implementation, provides a foundation that highlights the importance of identifying the most frequently cited practices in the literature and suggests that maybe the creation of a guide can enhance the existing knowledge and benefits of these practices. This guide would offer a structured set of recommendations, supporting the gradual adoption of the framework's practices while overcoming the challenges typically encountered during implementation.

4.4. Threats to Validity

The validity of the results obtained in this systematic review of the literature may be affected by several threats, which are discussed in the following sections.

4.4.1. Construct Validity

This threat refers to whether the research questions, inclusion/exclusion criteria, and search strings adequately reflect this study's objectives. Although the search strings were iteratively validated and refined, there is a risk that relevant terms should have been included, limiting the retrieval of key papers. In addition, some concepts, such as "fundamental practices" or "success factors", may be interpreted differently in primary studies, affecting data selection and analysis consistency.

Additionally, it is important to acknowledge that the search strategy focused on the Scopus and Web of Science (WoS) databases, which are recognized for their stringent quality filters and comprehensive coverage of the scholarly literature. The selection of these

databases ensures the inclusion of rigorously peer-reviewed research, which may exclude some studies published in less prominent or specialized venues.

4.4.2. Internal Validity

The decisions made during the study selection and analysis process could introduce bias, such as in the following examples:

- The inclusion and exclusion of studies depended partly on interpreting titles and abstracts, which could have led to the exclusion of relevant papers;
- The qualitative analysis was based on a thematic process that, although systematic, can be influenced by the subjectivity of the researchers. We employed a peer review of the identified categories and patterns to mitigate this, although bias was not eliminated.

4.4.3. External Validity

The ability to generalize findings to organizations or contexts other than those reported in the studies reviewed is limited. Since this literature review focused on very small organizations (VSEs), the success factors and practices identified might not directly apply to larger companies or those with complex organizational structures. In addition, including studies conducted in different regions and cultures could lead to results that are not homogeneous.

4.4.4. Validity of Conclusion

The results presented may be influenced by limitations in the quality of primary studies. Although a quality assessment process was used to prioritize robust papers, some studies require more details about their methodology, which could have affected the reliability of their conclusions. In addition, grouping common factors and practices can mask important contextual variations.

4.4.5. Other Limitations

An additional limitation is the time range of the selected studies. Although the search was comprehensive, the results may reflect only some recent studies published after the initial search. This aspect is relevant, given the dynamic nature of the Scrum and DevOps fields.

4.4.6. Mitigation

To address these threats, the following measures were taken:

- Refine search strings iteratively and apply multiple recognized databases (Scopus and Web of Science);
- Establish clear inclusion and exclusion criteria complemented by the snowballing technique;
- Cross-review among researchers to minimize bias in data selection and analysis;
- Explicitly document the context of primary studies to interpret the results according to their particularities.

5. Conclusions and Future Directions

The results of the literature review demonstrate that there is a marked trend in the industry in the adoption of Agile frameworks, especially Scrum. This trend is particularly evident in VSEs (with up to 25 employees) due to the benefits it brings within the development process, such as simple planning of activities, continuous deliveries that allow the customer to visualize the return on their investment in a short time, constant communication, and transparency of results. Likewise, the practices recommended by the

Scrum framework, such as actively involving stakeholders, having a clear and transparent communication channel, generating continuous information, measuring quality, and shorter delivery cycles to receive valuable feedback as soon as possible, are oriented to make quality a transversal axis of the entire development process.

In this context, DevOps marks a solid tendency to complement development processes by automating and implementing certain practices oriented toward a cleaner, maintainable, and standardized code of sufficient quality to accelerate deployment times through continuous integration, testing, monitoring, deployment, and analyzing the information generated to improve the process. All this goes hand in hand with a philosophy that transforms the organizational culture to bring the development and infrastructure areas closer together, eliminating natural barriers to development processes and controlled environments.

Therefore, the success of DevOps implementation in companies is related to elements such as collaborative culture and clear, transparent, and constant communication, in addition to the implementation of a set of practices supported by tools that ensure regular feedback on quality. Although there are many models of the possible practices that can be implemented, the potential order, and the effects they have on the development process, there is no guide that allows us to determine precisely, according to the company size and maturity level of the development process, which practices should be adopted and which path to follow to implement the recommendations one by one. This creates a significant opportunity for further research.

In addition, it can be stated that the practices identified as proposed by DevOps to improve the software development process are versioning, continuous integration, continuous testing (unit, integration, system, and acceptance), static code analysis (review of the international code standard for the programming language of the project, cyclomatic complexity, and OWASP top 10, among others), continuous monitoring, deployment, and continuous delivery. Although the documentation mentions several ways to configure these practices with various tools—including on-premises, on-cloud, or hybrid—the issue is the same as with Agile models, in that it can be achieved, but there is a lack of guidance on how to facilitate their adoption.

The same challenge exists with Scrum, where documentation on the experience of implementing the framework in companies of all sizes exists. However, a model or general guide on implementing the set of practices needs to be developed, as suggested by several review authors. This shows that adopting Agile thinking is challenging because it becomes a costly trial-and-error process for VSEs due to its characteristics. It is essential to highlight that Agile thinking fosters a collaborative culture, includes quality filters that generate continuous feedback through the participation of all stakeholders, and improves communication channels as crucial success factors in its implementation.

On the other hand, despite the diverse literature on Scrum and DevOps regarding their definitions and recommended practices, and even companies that have implemented them, no papers refer to a process of integrating common practices. This indicates that research regarding building a guide for VSEs to implement Scrum and DevOps is required and would provide a relevant contribution.

Finally, based on the results of the literature review, it can be understood that the common practices between Scrum and DevOps are focused on feedback and continuous improvement, complementing various practices. According to some authors, the trend in research and the industry is to take the best practices proposed by frameworks such as Scrum, DevOps, Lean, and Kanban, among others, and assemble methodologies according to the organizational culture, size, and maturity of their development processes. Therefore, a consensus on which practices to implement and a guided model to identify what and how to achieve a gradual implementation process is required.

The authors with the most publications in this review have produced four papers at most, indicating a significant opportunity to contribute further to this area of knowledge. It is also important to highlight that, according to the systematic review of the literature, the issue of the characterization of the companies, highlighting the common points they have in their configuration, organizational culture, infrastructure, size of work teams, types of projects they undertake, etc., has not been sufficiently addressed. Therefore, a specific characterization of VSEs that allow the implementation of Scrum and DevOps is necessary.

Author Contributions: M.P.: Conceptualization, Methodology Resources, Writing—Original draft preparation. H.O.: Conceptualization, Methodology, Resources, Writing—Original draft preparation, Supervision, Project administration, Funding acquisition. C.A.C.-L.: Supervision, Writing—Reviewing and Editing. M.M.: Supervision, Writing—Reviewing and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: For datasets that are not publicly available, requests can be made to the corresponding author, who will provide access upon approval, particularly for the data subject to ethical or privacy restrictions. This statement ensures that readers know where and how to access the data that supports the results and analyses presented in this article.

Acknowledgments: The authors express their gratitude to the University of Cauca, Popayán, Colombia; the Antonio José Camacho University Institution, Cali, Colombia; and the Mathematics Research Center, Zacatecas, Mexico, for participating in the Ph.D. research project entitled "Guide for the Implementation of Harmonized Software Development Practices Based on Scrum and DevOps in Very Small Companies" and for supporting the group of researchers in executing this project.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study.

Abbreviations

The following abbreviations are used in this manuscript:

VSEs	Very small entities
CI	Continuous integration
CD	Continuous deployment
DevOps	Development and operations
IaC	Infrastructure as code
SCA	Static code analysis
TDD	Test-driven development
BDD	Behavior-driven development

References

1. Pastrana, M.A.; Ordoñez, H.A.; Cobos, C. ISO 29110 En Colombia: De La Teoría a La Práctica. *Rev. Guillermo De Ockham* **2019**, *17*, 71–80. [CrossRef]
2. Hastie, S.; Wojewoda, S. Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch. Available online: http://www.infoq.com/articles/standish-chaos-2015 (accessed on 20 March 2025).
3. Annous, H.; Livadas, L.; Miles, G. OffshoreQA: A Framework for Helping Software Development Outsourcing Companies Comply with ISO 9001:2008. In Proceedings of the 5th International Conference on Global Software Engineering, ICGSE, Princeton, NJ, USA, 23–26 August 2010; pp. 313–315.
4. Schwaber, K.; Sutherland, J. La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas Del Juego. 2017. Available online: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf (accessed on 20 March 2025).
5. Pastrana Pardo, M.A.; Ordóñez Erazo, H.A.; Cobos Lozada, C.A. Approach to the Best Practices of Software Development Based on DevOps and SCRUM Used in Very Small Entities. *Rev. Fac. Ing.* **2022**, *31*, e14828. [CrossRef]
6. Ordóñez, H.A.; Buchelli, V. Lineamientos Para La Implementación Del Modelo CALMS de DevOps En Mipymes Desarrolladoras de Software En El Contexto Sur Colombiano. *Rev. Guillermo Ockham* **2020**, *18*, 81–91. [CrossRef]

7. Kniberg, H. *Scrum and XP from the Trenches: How We Do Scrum*; Lulu Press, Inc.: Morrisville, NC, USA, 2007. Available online: https://www.agileleanhouse.com/lib/lib/People/HenrikKniberg/ScrumAndXpFromTheTrenchesonline07-31.pdf (accessed on 20 March 2025).
8. Kitchenham, B.; Charters, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering Version 2.3. *Engineering* **2007**, *45*, 1051. [CrossRef]
9. Inayat, I.; Salim, S.S.; Marczak, S.; Daneva, M.; Shamshirband, S. A Systematic Literature Review on Agile Requirements Engineering Practices and Challenges. *Comput. Hum. Behav.* **2015**, *51*, 915–929. [CrossRef]
10. Wohlin, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, UK, 13–14 May 2014. [CrossRef]
11. Kitchenham, B.; Pretorius, R.; Budgen, D.; Pearl Brereton, O.; Turner, M.; Niazi, M.; Linkman, S. Systematic Literature Reviews in Software Engineering–A Tertiary Study. *Inf. Softw. Technol* **2010**, *52*, 792–805. [CrossRef]
12. Schwaber, K.; Sutherland, J. La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas Del Juego. 2020. Available online: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf (accessed on 20 March 2025).
13. Matook, S.; Maruping, L.M. A Competency Model for Customer Representatives in Agile Software Development Projects. *MIS Q. Exec.* **2014**, *13*, 77–95.
14. Wawryk, V.J.; Krenn, C.; Dietinger, T. Scaling a Running Agile Fix-Bid Project with near Shoring: Theory vs. Reality and (Best) Practice. In Proceedings of the 2015 IEEE 8th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2015-Proceedings, Graz, Austria, 13–17 April 2015.
15. Muñoz, M.; Mejia, J.; Corona, B.; Calvo-Manzano, J.A.; San Feliu, T.; Miramontes, J. Analysis of Tools for Assessing the Implementation and Use of Agile Methodologies in SMEs. In *Software Process Improvement and Capability Determination: 16th International Conference, SPICE 2016, Dublin, Ireland, 9–10 June 2016, Proceedings*; Communications in Computer and Information Science Series; Springer: Cham, Switzerland, 2016; Volume 609, pp. 123–134.
16. Chen, H.-M.; Kazman, R.; Haziyev, S. Agile Big Data Analytics Development: An Architecture-Centric Approach. In Proceedings of the IEEE 2016 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA, 5–8 January 2016; pp. 5378–5387.
17. Hobbs, B.; Petit, Y. Agile Methods on Large Projects in Large Organizations. *Proj. Manag. J.* **2017**, *48*, 3–19. [CrossRef]
18. Spurrier, G.; Topi, H. When Is Agile Appropriate for Enterprise Software Development? In Proceedings of the 25th European Conference on Information Systems, ECIS 2017, Guimarães, Portugal, 5–10 June 2017; pp. 2536–2545.
19. Carroll, N.; O'Connor, M.; Edison, H. The Identification and Classification of Impediments in Software Flow. In Proceedings of the Americas Conference on Information Systems 2018: Digital Disruption, AMCIS 2018, New Orleans, LA, USA, 16–18 August 2018.
20. Klünder, J.; Hohl, P.; Küpper, S.; Krusche, S.; Lous, P.; Fazal-Baqaie, M.; Prause, C.R. Towards Understanding the Motivation of German Organizations to Apply Certain Software Development Methods. In *Product-Focused Software Process Improvement: 19th International Conference, PROFES 2018, Wolfsburg, Germany, 28–30 November 2018, Proceedings*; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2018; Volume 11271, pp. 449–456.
21. Almeida, F.; Simões, J. Moving from Waterfall to Agile: Perspectives from IT Portuguese Companies. *Int. J. Serv. Sci. Manag. Eng. Technol.* **2019**, *10*, 30–43. [CrossRef]
22. Bajpai, S.; Eppinger, S.D.; Joglekar, N.R. The Structure of Agile Development Under Scaled Planning and Coordination. In Proceedings of the 21st International DSM Conference, Monterey, UK, 25 September 2019; The Design Society: Glasgow, UK; pp. 25–34.
23. López-Alcarria, A.; Olivares-Vicente, A.; Poza-Vilches, F. A Systematic Review of the Use of Agile Methodologies in Education to Foster Sustainability Competencies. *Sustainability* **2019**, *11*, 2915. [CrossRef]
24. Fronza, I.; Corral, L.; Pahl, C. End-User Software Development: Effectiveness of a Software Engineering-Centric Instructional Strategy. *J. Inf. Technol. Educ. Res.* **2020**, *19*, 367–393. [CrossRef]
25. Bajpai, S.; Eppinger, S.D.; Joglekar, N.R. Enhancing Visibility in Agile Program Increment DSMs. In Proceedings of the 22nd International Dependency and Structure Modeling Conference, DSM 2020, Cambridge, MA, USA, 13–15 October 2020; pp. 155–164.
26. Gerster, D.; Dremel, C.; Brenner, W.; Kelker, P. How Enterprises Adopt Agile Forms of Organizational Design. *ACM SIGMIS Database Database Adv. Inf. Syst.* **2020**, *51*, 84–103. [CrossRef]
27. Jana, D.; Pal, P. ESSENCE Kernel in Overcoming Challenges of Agile Software Development. In Proceedings of the 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 10–13 December 2020; pp. 1–8.
28. Mahajan, R.A.; Mahajan, S.A. Development of Scrum-Tree-KNN Algorithm for Distributed Agile Development. In Proceedings of the 2020 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 12–14 March 2020; pp. 17–21.

29. Almeida, F.; Simões, J. Leadership Challenges in Agile Environments. *Int. J. Inf. Technol. Proj. Manag.* **2021**, *12*, 30–44. [CrossRef]
30. Khan, A.A.; Shameem, M.; Nadeem, M.; Akbar, M.A. Agile Trends in Chinese Global Software Development Industry: Fuzzy AHP Based Conceptual Mapping. *Appl. Soft Comput.* **2021**, *102*, 107090. [CrossRef]
31. Russo, D. The Agile Success Model: A Mixed-Methods Study of a Large-Scale Agile Transformation. *ACM Trans. Softw. Eng. Methodol.* **2021**, *30*, 1–46. [CrossRef]
32. Fitzgerald, B.; Stol, K.-J. Continuous Software Engineering and beyond: Trends and Challenges. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering-RCoSE 2014, Hyderabad, India, 3 June 2014; ACM Press: New York, NY, USA, 2014; pp. 1–9.
33. de Bayser, M.; Azevedo, L.G.; Cerqueira, R. ResearchOps: The Case for DevOps in Scientific Applications. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 1398–1404.
34. Jabbari, R.; bin Ali, N.; Petersen, K.; Tanveer, B. What Is DevOps?: A Systematic Mapping Study on Definitions and Practices. In Proceedings of the Scientific Workshop Proceedings of XP2016, Edinburgh, UK, 24 May 2016; ACM: New York, NY, USA, 2016; Volume 24, pp. 1–11.
35. Mohan, V.; Othmane, L. Ben SecDevOps: Is It a Marketing Buzzword?-Mapping Research on Security in DevOps. In Proceedings of the IEEE 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 542–547.
36. Zheng, J.; Liu, Y.; Lin, J. Exploring and Enabling DevOps for Data Analytical System with Essential Demands Elicitation. *Int. J. Softw. Eng. Knowl. Eng.* **2016**, *26*, 1453–1472. [CrossRef]
37. Rajkumar, M.; Pole, A.K.; Adige, V.S.; Mahanta, P. DevOps Culture and Its Impact on Cloud Delivery and Software Development. In Proceedings of the 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA), Dehradun, India, 8–9 April 2016; pp. 1–6.
38. Airaj, M. Enable Cloud DevOps Approach for Industry and Higher Education. *Concurr. Comput.* **2017**, *29*, e3937. [CrossRef]
39. Bucena, I.; Kirikova, M. Simplifying the Devops Adoption Process. In Proceedings of the CEUR Workshop Proceedings, Bangalore, India, 8–10 December 2017; Volume 1898.
40. Huijgens, H.; Lamping, R.; Stevens, D.; Rothengatter, H.; Gousios, G.; Romano, D. Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, Paderborn, Germany, 4–8 September 2017; ACM: New York, NY, USA, 2017; pp. 866–871.
41. Fazal-Baqaie, M.; Güldali, B.; Oberthür, S. Towards DevOps in Multi-Provider Projects. In Proceedings of the CEUR Workshop Proceedings, Bangalore, India, 8–10 December 2017; Volume 1806, pp. 18–21.
42. Díaz, J.; Almaraz, R.; Pérez, J.; Garbajosa, J. DevOps in Practice: An exploratory case study. In Proceedings of the 19th International Conference on Agile Software Development: Companion, Porto, Portugal, 21–25 May 2018; ACM: New York, NY, USA, 2018; pp. 1–3.
43. Raibulet, C.; Arcelli Fontana, F. Collaborative and Teamwork Software Development in an Undergraduate Software Engineering Course. *J. Syst. Softw.* **2018**, *144*, 409–422. [CrossRef]
44. Jabbari, R.; bin Ali, N.; Petersen, K.; Tanveer, B. Towards a Benefits Dependency Network for DevOps Based on a Systematic Literature Review. *J. Softw. Evol. Process* **2018**, *30*, e1957. [CrossRef]
45. Stocker, W. From Agile to Continuous Development in the Healthcare Domain: Lessons Learned. In Proceedings of the International Conference on Software Engineering, Beijing, China, 23–25 November 2018; pp. 211–212.
46. Silva, M.A.; Faustino, J.P.; Pereira, R.; da Silva, M.M. Productivity Gains of DevOps Adoption in an IT Team: A Case Study. In Proceedings of the 27th International Conference on Information Systems Development: Designing Digitalization, ISD 2018, Lund, Sweden, 22–24 August 2018.
47. Elberzhager, F.; Naab, M. High Quality at Short Time-to-Market: Challenges Towards This Goal and Guidelines for the Realization. In *Software Quality: Methods and Tools for Better Software and Systems—10th International Conference, SWQD 2018, Vienna, Austria, 16–19 January 2018, Proceedings*; Lecture Notes in Business Information Processing; Springer: Cham, Switzerland, 2018; Volume 302, pp. 121–132.
48. Martinez-Fernandez, S.; Vollmer, A.M.; Jedlitschka, A.; Franch, X.; Lopez, L.; Ram, P.; Rodriguez, P.; Aaramaa, S.; Bagnato, A.; Choras, M.; et al. Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study. *IEEE Access* **2019**, *7*, 68219–68239. [CrossRef]
49. Luz, W.P.; Pinto, G.; Bonifácio, R. Adopting DevOps in the Real World: A Theory, a Model, and a Case Study. *J. Syst. Softw.* **2019**, *157*, 110384. [CrossRef]
50. Zulfahmi Toh, M.; Sahibuddin, S.; Mahrin, M.N. Adoption Issues in DevOps from the Perspective of Continuous Delivery Pipeline. In Proceedings of the PervasiveHealth: Pervasive Computing Technologies for Healthcare, Trento, Italy, 20–23 May 2019; Volume F1479, pp. 173–177.

51. Siewruk, G.; Mazurczyk, W.; Karpiński, A. Security Assurance in Devops Methodologies and Related Environments. *Int. J. Electron. Telecommun.* **2019**, *65*, 211–216. [CrossRef]
52. Badshah, S.; Khan, A.A.; Khan, B. Towards Process Improvement in DevOps: A Systematic Literature Review. In Proceedings of the PervasiveHealth: Pervasive Computing Technologies for Healthcare, Atlanta, GA, USA, 18–20 May 2020; pp. 427–433.
53. Khan, A.A.; Shameem, M. Multicriteria Decision-making Taxonomy for DevOps Challenging Factors Using Analytical Hierarchy Process. *J. Softw. Evol. Process* **2020**, *32*, e2263. [CrossRef]
54. Macarthy, R.W.; Bass, J.M. An Empirical Taxonomy of DevOps in Practice. In Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Portoroz, Slovenia, 26–28 August 2020; pp. 221–228.
55. Rafi, S.; Yu, W.; Akbar, M.A. Towards a Hypothetical Framework to Secure DevOps Adoption: Grounded Theory Approach. In Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering, Trondheim, Norway, 15–17 April 2020; ACM: New York, NY, USA, 2020; pp. 457–462.
56. Rafi, S.; Yu, W.; Akbar, M.A.; Alsanad, A.; Gumaei, A. Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE. *IEEE Access* **2020**, *8*, 105426–105446. [CrossRef]
57. Wang, Y.; Mäntylä, M.V.; Demeyer, S.; Wiklund, K.; Eldh, S.; Kairi, T. Software Test Automation Maturity: A Survey of the State of the Practice. In Proceedings of the ICSOFT 2020-Proceedings of the 15th International Conference on Software Technologies, Online, 7–9 July 2020; pp. 27–38.
58. Zaydi, M.; Nassereddine, B. DevSecOps Practices for an Agile and Secure IT Service Management. *J. Manag. Inf. Decis. Sci.* **2020**, *23*, 134–149.
59. Raj, P.; Sinha, P. Project Management in Era of Agile and Devops Methodlogies. *Int. J. Sci. Technol. Res.* **2020**, *9*, 1024–1033.
60. Sliep, C.; Marnewick, C. The Quest in Delivering Quality IT Services: The Case of a Higher Education Institution. *Educ. Inf. Technol.* **2020**, *25*, 4817–4844. [CrossRef]
61. Leite, L.; Pinto, G.; Kon, F.; Meirelles, P. The Organization of Software Teams in the Quest for Continuous Delivery: A Grounded Theory Approach. *Inf. Softw. Technol.* **2021**, *139*, 106672. [CrossRef]
62. Rafi, S.; Yu, W.; Akbar, M.A.; Mahmood, S.; Alsanad, A.; Gumaei, A. Readiness Model for DevOps Implementation in Software Organizations. *J. Softw. Evol. Process* **2021**, *33*, e2323. [CrossRef]
63. Mahdavi-Hezaveh, R.; Dremann, J.; Williams, L. Software Development with Feature Toggles: Practices Used by Practitioners. *Empir. Softw. Eng.* **2021**, *26*, 1. [CrossRef]
64. Toh, M.Z.; Sahibuddin, S.; Bakar, R.A. A Review on DevOps Adoption in Continuous Delivery Process. In Proceedings of the 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), Pekan, Malaysia, 24–26 August 2021; pp. 98–103.
65. Rajapakse, R.N.; Zahedi, M.; Babar, M.A.; Shen, H. Challenges and Solutions When Adopting DevSecOps: A Systematic Review. *Inf. Softw Technol.* **2022**, *141*, 106700. [CrossRef]
66. Nisha, T.N.; Khandebharad, A. Migration From DevOps to DevSecOps: A Complete Migration Framework, Challenges, and Evaluation. *Int. J. Cloud Appl. Comput. (IJCAC)* **2021**, *12*, 1–15. [CrossRef]
67. Akbar, M.A.; Rafi, S.; Alsanad, A.A.; Qadri, S.F.; Alsanad, A.; Alothaim, A. Toward Successful DevOps: A Decision-Making Framework. *IEEE Access* **2022**, *10*, 51343–51362. [CrossRef]
68. Jayakody, J.A.V.M.K.; Wijayanayake, W.M.J.I. Critical Success Factors for DevOps Adoption in Information Systems Development. *Int. J. Inf. Syst. Proj. Manag.* **2023**, *11*, 60–82. [CrossRef]
69. Kumar, R.; Goyal, R. Modeling Continuous Security: A Conceptual Model for Automated DevSecOps Using Open-Source Software over Cloud (ADOC). *Comput. Secur.* **2020**, *97*, 101967. [CrossRef]
70. Baškarada, S.; Nguyen, V.; Koronios, A. Architecting Microservices: Practical Opportunities and Challenges. *J. Comput. Inf. Syst.* **2020**, *60*, 428–436. [CrossRef]
71. Ntentos, E.; Zdun, U.; Plakidas, K.; Meixner, S.; Geiger, S. Metrics for Assessing Architecture Conformance to Microservice Architecture Patterns and Practices. In *Service-Oriented Computing: 18th International Conference, ICSOC 2020, Dubai, United Arab Emirates, 14–17 December 2020, Proceedings*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2020; Volume 12571, pp. 580–596.
72. Daneva, M.; Bolscher, R. What We Know About Software Architecture Styles in Continuous Delivery and Devops? In *Software Technologies: 14th International Conference, ICSOFT 2019, Prague, Czech Republic, 26–28 July 2019, Revised Selected Papers*; Communications in Computer and Information Science Series; Springer: Cham, Switzerland, 2020; Volume 1250, ISBN 9783030529901.
73. Henkel, J.; Bird, C.; Lahiri, S.K.; Reps, T. Learning from, Understanding, and Supporting Devops Artifacts for Docker. In Proceedings of the Proceedings-International Conference on Software Engineering, Limerick, Ireland, 9 June 2020; pp. 38–49.
74. Hills, M. Introducing DevOps Techniques in a Software Construction Class. In Proceedings of the 2020 IEEE 32nd Conference on Software Engineering Education and Training, CSEE and T 2020, Munich, Germany, 9–12 November 2020; pp. 85–89.
75. Hemon, A.; Fitzgerald, B.; Lyonnet, B.; Rowe, F. Innovative Practices for Knowledge Sharing in Large-Scale DevOps. *IEEE Softw.* **2020**, *37*, 30–37. [CrossRef]

76. Leite, L.; Kon, F.; Pinto, G.; Meirelles, P. Platform Teams: An Organizational Structure for Continuous Delivery. In Proceedings of the 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020, Seoul, Republic of Korea, 27 June–19 July 2020; pp. 505–511.
77. Rangnau, T.; Buijtenen, R.V.; Fransen, F.; Turkmen, F. Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines. In Proceedings of the 2020 IEEE 24th International Enterprise Distributed Object Computing Conference, EDOC 2020, Eindhoven, The Netherlands, 5–8 October 2020; pp. 145–154.
78. João, N.A.; Faustino, O.; Pereira, R.; Alturas, B.; Silva, M.M. da Agile Information Technology Service Management with DevOps: An Incident Management Case Study. *Int. J. Agil. Syst. Manag.* **2020**, *13*, 339. [CrossRef]
79. Lim, G.; Ham, M.; Moon, J.; Song, W. LightSys: Lightweight and Efficient CI System for Improving Integration Speed of Software. In Proceedings of the International Conference on Software Engineering, Limerick, Ireland, 9 June 2021; pp. 91–100.
80. Muñoz, A.; Farao, A.; Correia, J.R.C.; Xenakis, C. P2ise: Preserving Project Integrity in Ci/Cd Based on Secure Elements. *Information* **2021**, *12*, 357. [CrossRef]
81. Railic, N.; Savic, M. Architecting Continuous Integration and Continuous Deployment for Microservice Architecture. In Proceedings of the 2021 20th International Symposium INFOTEH-JAHORINA, INFOTEH 2021-Proceedings, Jahorina, Republic of Srpska, Bosnia and Herzegovina, 17–19 March 2021.
82. Castellanos, C.; Varela, C.A.; Correal, D. ACCORDANT: A Domain Specific-Model and DevOps Approach for Big Data Analytics Architectures. *J. Syst. Softw.* **2021**, *172*, 110869. [CrossRef]
83. Rodriguez, M.; de Araújo, L.J.P.; Mazzara, M. Good Practices for the Adoption of DataOps in the Software Industry. *J. Phys. Conf. Ser.* **2020**, *1694*, 012032. [CrossRef]
84. Batra, P.; Jatain, A. Hybrid Model for Evaluation of Quality Aware DevOps. *Int. J. Appl. Sci. Eng.* **2021**, *18*, 1–11. [CrossRef]
85. Tomas, N.; Li, J.; Huang, H. An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps. In Proceedings of the 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Oxford, UK, 3–4 June 2019; pp. 1–8.
86. Alnaim, A.A. Using Rules Engine in the Automation of System Security Review. In Proceedings of the 2019 IEEE Cybersecurity Development (SecDev), IEEE, McLean, VA, USA, 25–27 September 2019; p. 142.
87. Meixner, K.; Winkler, D.; Biffl, S. Towards Combined Process & Tool Variability Management in Software Testing. In Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems, Leuven, Belgium, 6–8 February 2019; ACM: New York, NY, USA, 2019; pp. 1–6.
88. Boehm, B.; Behnamghader, P. Anticipatory Development Processes for Reducing Total Ownership Costs and Schedules. *Syst. Eng.* **2019**, *22*, 401–410. [CrossRef]
89. Bolscher, R.; Daneva, M. Designing Software Architecture to Support Continuous Delivery and DevOps: A Systematic Literature Review. In Proceedings of the 14th International Conference on Software Technologies, SCITEPRESS-Science and Technology Publications, Prague, Czech Republic, 2–4 May 2019; pp. 27–39.
90. Oliinyk, B.; Oleksiuk, V. Automation in Software Testing, Can We Automate Anything We Want? In Proceedings of the CEUR Workshop Proceedings, Kryvyi Rih, Ukraine, 20 December 2019; Volume 2546, pp. 224–234.
91. Jennings, R.A.K.; Gannod, G. DevOps-Preparing Students for Professional Practice. In Proceedings of the Frontiers in Education Conference, FIE, Lincoln, NE, USA, 13–16 October 2019.
92. Guseila, L.G.; Bratu, D.-V.; Moraru, S.-A. Continuous Testing in the Development of IoT Applications. In Proceedings of the 2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI), Lisbon, Portugal, 29–30 August 2019; pp. 1–6.
93. Ferry, N.; Solberg, A.; Song, H.; Lavirotte, S.; Tigli, J.-Y.; Winter, T.; Muntés-Mulero, V.; Metzger, A.; Rios Velasco, E.; Castelruiz Aguirre, A. ENACT: Development, Operation, and Quality Assurance of Trustworthy Smart IoT Systems. In *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, 5–6 March 2018, Revised Selected Papers*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2019; Volume 11350, pp. 112–127.
94. Rahman, A.; Agrawal, A.; Krishna, R.; Sobran, A. Characterizing the Influence of Continuous Integration: Empirical Results from 250+ Open Source and Proprietary Projects. In Proceedings of the 4th ACM SIGSOFT International Workshop on Software Analytics, Lake Buena Vista, FL, USA, 5 November 2018; ACM: New York, NY, USA, 2018; pp. 8–14.
95. Ivanov, V.; Smolander, K. Implementation of a DevOps Pipeline for Serverless Applications. In *Product-Focused Software Process Improvement: 19th International Conference, PROFES 2018, Wolfsburg, Germany, 28–30 November 2018, Proceedings*; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2018; Volume 11271, pp. 48–64, ISBN 9783030036720.
96. Erder, M.; Pureur, P. *Continuous Architecture: Sustainable Architecture in an Agile and Cloud-Centric World*; Morgan Kaufmann: Waltham, MA, USA, 2015; ISBN 9780128032848.

97. Di Nitto, E.; Jamshidi, P.; Guerriero, M.; Spais, I.; Tamburri, D.A. A Software Architecture Framework for Quality-Aware DevOps. In Proceedings of the 2nd International Workshop on Quality-Aware DevOps, Saarbrücken, Germany, 21 July 2016; ACM: New York, NY, USA, 2016; pp. 12–17.
98. Adams, B.; McIntosh, S. Modern Release Engineering in a Nutshell–Why Researchers Should Care. In Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Osaka, Japan, 14–18 March 2016; pp. 78–90.
99. Perez-Palacin, D.; Ridene, Y.; Merseguer, J. Quality Assessment in DevOps: Automated Analysis of a Tax Fraud Detection System. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion, L'Aquila, Italy, 22–26 April 2017; ACM: New York, NY, USA, 2017; pp. 133–138.
100. Zhao, Y.; Serebrenik, A.; Zhou, Y.; Filkov, V.; Vasilescu, B. The Impact of Continuous Integration on Other Software Development Practices: A Large-Scale Empirical Study. In Proceedings of the 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA, 30 October–3 November 2017; pp. 60–71.
101. Lenhard, J. Improving Process Portability Through Metrics and Continuous Inspection. In *Advances in Intelligent Process-Aware Information Systems: Concepts, Methods, and Technologies*; Intelligent Systems Reference Library Series; Springer: Cham, Switzerland, 2017; Volume 123, pp. 193–223.
102. Fitzgerald, B.; Stol, K.-J. Continuous Software Engineering: A Roadmap and Agenda. *J. Syst. Softw.* **2017**, *123*, 176–189. [CrossRef]
103. Spinellis, D. State-of-the-Art Software Testing. *IEEE Softw.* **2017**, *34*, 4–6. [CrossRef]
104. Meng, F.J.; Wegman, M.N.; Xu, J.M.; Zhang, X.; Chen, P.; Chafle, G. IT Troubleshooting with Drift Analysis in the DevOps Era. *IBM J. Res. Dev.* **2017**, *61*, 6:62–6:73. [CrossRef]
105. Qumer Gill, A.; Loumish, A.; Riyat, I.; Han, S. DevOps for Information Management Systems. *VINE J. Inf. Knowl. Manag. Syst.* **2018**, *48*, 122–139. [CrossRef]
106. Mascheroni, M.A.; Irrazábal, E. Continuous Testing and Solutions for Testing Problems in Continuous Delivery: A Systematic Literature Review. *Comput. Y Sist.* **2018**, *22*, 1009–1038. [CrossRef]
107. Mohan, V.; ben Othmane, L.; Kres, A. BP: Security Concerns and Best Practices for Automation of Software Deployment Processes: An Industrial Case Study. In Proceedings of the 2018 IEEE Cybersecurity Development (SecDev), Cambridge, MA, USA, 30 September–2 October 2018; pp. 21–28.
108. Luz, W.P.; Pinto, G.; Bonifácio, R. Building a Collaborative Culture: A grounded theory of well succeeded devops adoption in practice. In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Oulu, Finland, 11–12 October 2018; ACM: New York, NY, USA, 2018; pp. 1–10.
109. Steffens, A.; Lichter, H.; Moscher, M. Towards Data-Driven Continuous Compliance Testing. In Proceedings of the CEUR Workshop Proceedings, Copenhagen, Denmark, 14 October 2018; Volume 2066, pp. 78–84.
110. Cruzes, D.S.; Melsnes, K.; Marczak, S. Testing in a DevOps Era: Perceptions of Testers in Norwegian Organisations. In *Computational Science and Its Applications—ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, 1–4 July 2019, Proceedings, Part IV*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2019; Volume 11622, pp. 442–455.
111. Mumbarkar, P.; Prasad, S. Adopting DevOps: Capabilities, Practices, and Challenges Faced by Organizations. *AIP Conf. Proc.* **2022**, *2519*, 030029. [CrossRef]
112. Rafi, S.; Akbar, M.A.; Mahmood, S.; Alsanad, A.; Alothaim, A. Selection of DevOps Best Test Practices: A Hybrid Approach Using ISM and Fuzzy TOPSIS Analysis. *J. Softw. Evol. Process* **2022**, *34*, e2448. [CrossRef]
113. Bonet Faus, J.; Le Masson, P.; Pelissier, U.; Jibet, N.; Bordas, A.; Pajot, S. Design Methods for Diagnosing and Locating Entangled Technical Debt in Devops Frameworks. *Proc. Des. Soc.* **2023**, *3*, 1267–1276. [CrossRef]
114. Hernández, R.; Moros, B.; Nicolás, J. Requirements Management in DevOps Environments: A Multivocal Mapping Study. *Requir. Eng.* **2023**, *28*, 317–346. [CrossRef]
115. Kumar, A.; Nadeem, M.; Shameem, M. Machine Learning Based Predictive Modeling to Effectively Implement DevOps Practices in Software Organizations. *Autom. Softw. Eng.* **2023**, *30*, 21. [CrossRef]
116. Pandiyavathi, T.; Sivakumar, B. DevOps Challenges and Practices in Software Engineering. In *Intelligent Sustainable Systems: Proceedings of ICISS 2023*; Lecture Notes in Networks and Systems; Springer: Singapore, 2023; Volume 665, pp. 49–57. [CrossRef]
117. Sravani, D.; Reddy, J.R.; Viswas, P.S.; Jyothi, N.M.; Chandukiran, P. Python Security in DevOps: Best Practices for Secure Coding, Configuration Management, and Continuous Testing and Monitoring. In Proceedings of the 2023 4th International Conference on Electronics and Sustainable Communication Systems, ICESC 2023–Proceedings, Coimbatore, India, 6–8 July 2023; pp. 514–520. [CrossRef]
118. Rani, V.S.; Babu, A.R.; Deepthi, K.; Reddy, V.R. Shift-Left Testing in DevOps: A Study of Benefits, Challenges, and Best Practices. In Proceedings of the 2nd International Conference on Automation, Computing and Renewable Systems, ICACRS 2023–Proceedings, Pudukkottai, India, 11–13 December 2023; pp. 1675–1680. [CrossRef]

119. Narasimhulu, M.; Mounika, D.V.; Varshini, P.; Amarendra, K.; Rao, T.K.R.K. Investigating the Impact of Containerization on the Deployment Process in DevOps. In the Proceedings of the 2nd International Conference on Edge Computing and Applications, ICECAA, Namakkal, India, 19–21 July 2023; pp. 679–685. [CrossRef]
120. Hemon, A.; Lyonnet, B.; Rowe, F.; Fitzgerald, B. From Agile to DevOps: Smart Skills and Collaborations. *Inf. Syst. Front.* **2020**. [CrossRef]
121. Hemon, A.; Lyonnet, B.; Rowe, F.; Fitzgerald, B. Conceptualizing the Transition from Agile to DevOps: A Maturity Model for a Smarter Is Function. In Proceedings of the IFIP Advances in Information and Communication Technology, Lisbon, Portugal, 25–27 June 2019.
122. Saltz, J.; Sutherland, A. Ski: A New Agile Framework That Supports Devops, Continuous Delivery, and Lean Hypothesis Testing. In Proceedings of the Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 7–10 January 2020; pp. 6217–6226.