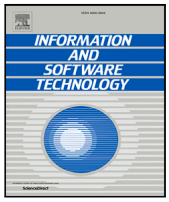
Information and Software Technology 177 (2025) 107583

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Mapping DevOps capabilities to the software life cycle: A systematic literature review

Ricardo Amaro [a,*], Rúben Pereira [a], Miguel Mira da Silva [b]

[a] Instituto Universitário de Lisboa (ISCTE-IUL), INOV, Portugal
[b] Instituto Superior Técnico, Universidade de Lisboa, INOV, Portugal

ARTICLE INFO

Keywords:
DevOps
Metrics
Performance
Adoption
Software development life cycle
Information system

ABSTRACT

Context: Many IT organizations are looking towards DevOps to make their software development and delivery processes faster and more reliable, while DevOps revolutionized the industry by emphasizing collaboration between development and operations teams. Nonetheless, there still exist challenges in harmonizing cultural, technical, measurement and process capabilities for its successful adoption.
Objective: To research improving DevOps adoption, this study explores DevOps Capabilities relevant to the Life Cycle Processes (LCPs) of the IEEE 2675-2021 DevOps standard. Aiming to provide valuable information on increasing efficiency and outcomes by mapping DevOps Capabilities in each phase of the LCPs. Whereas previous research identified and classified 37 DevOps Capabilities, this study aims to determine which capabilities can enhance each of the 30 phases of the LCPs.
Methods: Out of 102 documents identified in the Systematic Literature Review (SLR), relations among DevOps Capabilities and LCPs have been synthesized and organized. An in-depth analysis of data was conducted over the connections across various categories. The mapping revealed how they relate in terms of their application and impact.
Results: The SLR shows technical DevOps Capabilities and technical LCPs strongly correlated. DevOps measurement capabilities have a significant impact on agreement processes. Using an impact scale classification, the study identifies eight capabilities that have exceptional impact on LCPs and eleven capabilities that have a very high impact on the supply process, requirements definition, integration process, and validation process.
Conclusion: The study demonstrates how DevOps Capabilities together with LCPs can improve software delivery, quality, and reliability. It presents a structured approach for improving processes, as well as evidence of DevOps integration in software development and maintenance. The findings help to assess DevOps Capabilities and LCP relations, which is expected to improve successful adoption. Future research should focus on researching practical cases of DevOps integration into LCPs, while overcoming adoption challenges.

Contents

* Corresponding author.
E-mail addresses: ricardo_amaro@iscte-iul.pt (R. Amaro), ruben.filipe.pereira@iscte-iul.pt (R. Pereira), mms@tecnico.ulisboa.pt (M.M.d. Silva).

https://doi.org/10.1016/j.infsof.2024.107583
Received 22 May 2023; Received in revised form 12 September 2024; Accepted 14 September 2024
Available online 19 September 2024

1. Introduction

1.1. Context

Since the early years of Software Development (SD), developers are consistently trying to find the best ways to produce and deliver software. Similarly, companies today also seek to improve methods of creating and implementing software with high quality and return on investment in order to meet the demands of customers and the market [1]. Thus, over the years there has been a dramatic change in SD models, for example, from conventional Waterfall to the Agile methodology. More recently, these organizations are looking to modernize their development environment rapidly by reducing development cycles and improving continuous delivery using a cutting-edge paradigm emphasizing the collaboration of *Developers(Dev) and Operations(Ops)* (DevOps). DevOps helps improve situations where Software delivery is a somewhat risky, complex or lengthy process [2]. Last-minute defects and integration issues frustrate end users, development teams, and business stakeholders. Moreover, coordination issues across teams frequently result in the execution of incorrect functionality, integration, and deployment issues, and finger-pointing. Thus, DevOps is an enabler of software delivery performance [3].

1.2. Problem

The *problem* remains that, while many organizations have been successful in implementing DevOps internally, others are still failing when trying to incorporate the cultural, technical, measurement, and process capabilities of DevOps [4–6]. Therefore, DevOps adoption remains uncertain [7,8], emphasizing the importance of providing managers and teams with appropriate information to successfully support the implementation of DevOps Capabilities and practices [4].

For organizations that do not employ Agile methodologies, it is unclear how DevOps can improve software Life Cycle Processes (LCPs) beyond Agile. The suitability of DevOps for waterfall or other methodologies is questioned [9]. While practices like continuous integration and delivery are said incompatible with waterfall due to its lack of continuity [10], DevOps-specific capabilities can enhance team efficiency even in other environments [9].

From another more formal perspective, the IEEE Standard for DevOps [11] states that DevOps is suitable for most LCP models [12], and "particularly appropriate for teams adopting Agile methodologies". While it is stated that DevOps is suitable for most LCP models, for instance in an iterative waterfall approach [11, p. 32], it is not clear how DevOps will be suitable in other methodologies in a generic way, where each process benefits from one or more DevOps Capabilities. How is it suitable? What DevOps Capabilities can improve each LCP?

1.3. Proposal and objective

The DevOps Standard IEEE Std 2675:2021 [11] is aligned with the Configuration Management IEEE Std 828:2012 [13] and closely adheres to the ISO/IEC/IEEE 15288:2015 [12] and ISO/IEC/IEEE 12207:2017 LCP standards [14]. However, these standards were developed before the systematization of DevOps Capabilities [4], leaving room for improvement. This research proposes a comprehensive literature review to determine which DevOps Capabilities can improve each Software and System LCP presented in the IEEE Standard for DevOps [11]. Previous research has identified and categorized 37 DevOps Capabilities [4]. It will examine how each DevOps Capability relates to each LCP, aiming to enhance existing standards by providing a precise understanding of the impact of DevOps Capabilities on LCPs.

This paper includes an extensive literature review and utilizes a conceptual map seen in Fig. 1 as the framework for analysis in Section 5. Processes consist of interrelated activities that transform inputs into outputs, with the process outcome reflecting the successful attainment of their purpose, which is the high-level objective and expected outcomes of effective process implementation [14].

In Fig. 1 it can be seen how they are interrelated. Each process in the life cycle has a purpose defined in the standard and attains outcomes [15] also defined there, for which it includes specific activities and tasks performed by teams. Teams adopting the process need to learn the skills and knowledge required by DevOps Capabilities in order to enable the activities and tasks of the process.

On the other hand, as proposed, the acquired DevOps Capabilities must be periodically evaluated, through a DevOps assessment, in order to generate and ensure the results intended by the process. The purpose of implementing the process is to provide benefits to the stakeholders [14].

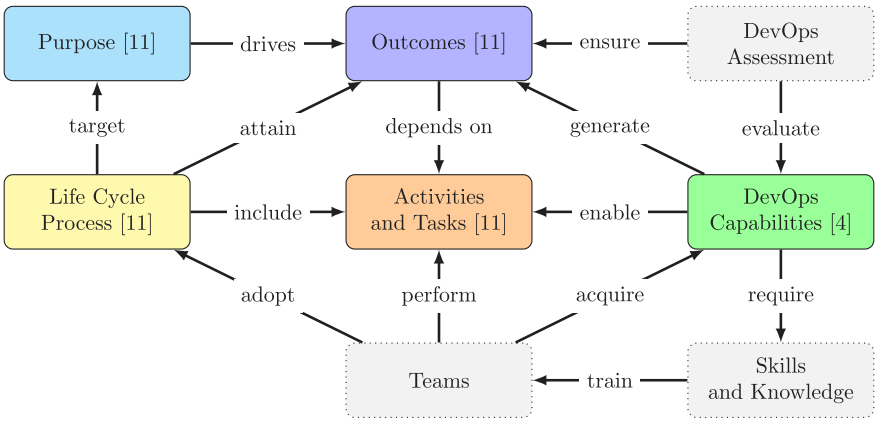


Fig. 1. Relating DevOps Capabilities [4] to LCPs[11] conceptual map.

Based on the original question of how DevOps Capabilities can improve each LCP, this study aims to find and study DevOps Capabilities that are relevant to LCPs from ISO/IEC/IEEE 12207 and IEEE 2675-2021. This is because DevOps is an interdisciplinary field that could use more management-focused research [16, p. 7]. This study conducts a SLR to identify relevant literature that discusses or examines the relationship between DevOps and LCPs. The research questions that will guide this study are:

- **RQ1**. How do authors in scientific literature relate Software Life Cycle Processes to DevOps Capabilities?
- **RQ2**. Which categories of DevOps Capabilities are most relevant to the Software Life Cycle Processes?

This research is grounded in the need to understand how DevOps can improve the software development process by identifying the most relevant capabilities to LCPs.

2. Research background

This section provides a theoretical foundation for the study area of this research. Furthermore, this SLR also gives an overview of other similar studies in Section 4.1. More related work has been previously analyzed in published the studies listed in Section 5 where 37 DevOps Capabilities were extracted from an Multivocal Literature Review (MLR), also mentioned in this section, were harmonized. Although all reviewed papers acknowledge the connection between DevOps and the software development process, none of them explicitly does an SLR to address how authors relate DevOps Capabilities [4] to LCPs from IEEE Standard 2675-2021 [11], which is a key novelty of this paper.

2.1. DevOps capabilities

DevOps comprises capabilities and continuous practices aimed at facilitating rapid software development and delivery through collaborative efforts among development, testing, and operations teams [4,17–19]. It fosters a cultural mindset change, eliminating information silos and promoting higher delivery, quality, and cooperation [5,20]. Automation plays a crucial role in DevOps, leveraging both Free/Libre and Open Source Software (FLOSS) and other tools such as Chef, Puppet, Ansible, Linux, Kubernetes, Jenkins, and Prometheus [21]. Continuous monitoring, feedback, integration (CI), and deployment (CD) are key capabilities that shorten time to market and ensure software correctness and reliability [5,22].

Businesses adopt DevOps to achieve a balance between velocity and system reliability, addressing stakeholder needs and functionality early in the software development cycle [11,15]. This software development process requires identifying, engaging, and collaborating with all stakeholders. This study will utilize the identified DevOps Capabilities seen in Fig. 2 to address the research questions.

The investigation done in this paper also takes into account the processes and definition proposed for DevOps in the IEEE Standard 2675-2021 [11], where it is described as a set of principles and practices that encourage increased communication and collaboration among stakeholders. Majorly involved in designing, developing, and running systems and software products or services, as well as achieving continuous improvement in all aspects of that entity's LCPs.

Finally, DevOps is a fast-growing cultural shift. It stresses building an Agile relationship and collaboration between software development and operations [20], namely with the use of tools to automate the management of software infrastructures, which, over the years, have become complex, heterogeneous, and of large-scale [23].

2.2. Software life cycle processes

For over 60 years, researchers have studied software processes and life cycle models [24]. These models provide an organized and effective approach to software development and delivery, defining roles, activities, and expected results. Initially, the term "software development" replaced "computer system development" as software and hardware were developed together, making code changes time-consuming and costly. Interest in software processes was limited until Benington's work in the 1950s [25], which presented an explicit representation of a Software Development Life Cycle (SDLC). In 1970, Winston W. Royce's paper [26] introduced the concept of a SDLC and described a sequential and interactive approach, now known as the waterfall model.

The term "waterfall model" was later coined by Bell and Thayer in 1976 [27], referring to Royce's work. It became a widely recognized life cycle model, providing a foundation for estimation, project monitoring, and other tasks [28]. Royce's article highlighted the state of the waterfall model at the time and proposed improvements to mitigate the risks of redesign and rework [29]. Since then, there have been many distinct approaches, with different Software Engineering (SE) methodologies and methods. Several other software processes and models like the V-Model [30], Iterative Enhancement [31], Prototyping [32], Spiral [33] all with considerable differences from each other, which have been the object of other studies [24,29], all focused in improving software development and delivery. Today we witness many ways of addressing the whole software product life cycle or portions of it.

Software process models and life cycle models are distinct concepts. A **life cycle model**, as defined in SEVOCAB[1] [11], serves as a framework for the stages and activities involved in the software

[1] https://pascal.computer.org/sev_display

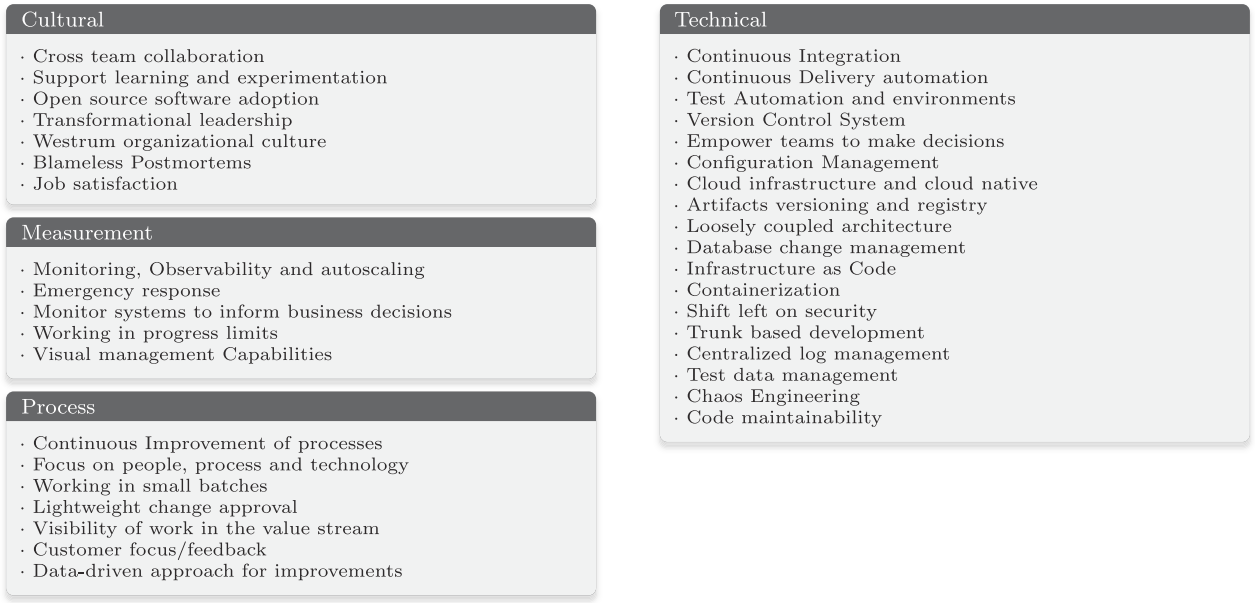


Fig. 2. Categorization of DevOps Capabilities.
Source: Adapted [4].

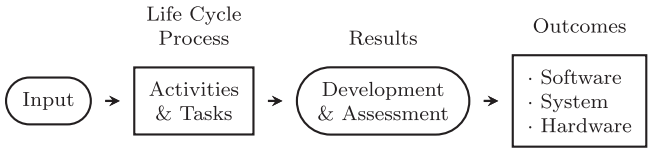


Fig. 3. Processes and Life Cycle Processes in Software Engineering (in Fig. 4).

life cycle, providing a common reference for communication and understanding. On the other hand, software process models offer more detailed information, including sub-steps, outputs, and the roles of individuals involved. As an example, using the Waterfall Life Cycle model outlines the high-level sequential stages of development like Requirements Analysis, Implementation or Maintenance, while in a Software Process Model, employing the Agile practices within the coding phase specifies iterative, collaborative methods for executing the work, like Sprint Planning, Daily Stand-ups or Retrospectives. A **process** refers to a set of interconnected activities that transform inputs into outputs, aiming to achieve a specific result [12]. In contrast, LCPs, illustrated in Fig. 3 and listed in Fig. 4, encompasses the processes involved in the development or evaluation of software, hardware, or system products [14]. Therefore, a process is a broader concept that encompasses the execution of relevant activities [22].

Software processes and life cycle models play a vital role in supporting organizational goals and strategies related to software consumption or development. They are integral to Information Technology (IT) governance, aiming to deliver sustainable, standardized services and achieve desired objectives. Software processes and life cycle models reduce risk, enhance the predictability of LCPs, and align with stakeholders' perspectives, including senior management and external regulatory agencies concerned about reliability, security, and error-free products [11].

The emergence of Agile methodologies created a cultural conflict between plan-driven models like Waterfall and Agile development. While Agile has clear advantages, its adoption was initially slow, with project managers preferring more control through strict, planned approaches [18,23]. However, there is a growing understanding that both approaches share the goal of efficiently building quality software, leading to the adoption of hybrid development models that combine elements from both approaches [24].

The ISO/IEC/IEEE 12207:2017 standard provides a reference model for structuring the software life cycle into different processes, known as Software LCPs. It aligns to the requirements of Waterfall or Agile approaches, accommodating the incremental and iterative nature of Agile development and the detailed specifications and monitoring of Waterfall projects. The standard establishes common terminology and serves as a reference for activities like process definition, modeling, and assessment. It complements other process standards, including IEEE Standard 2675-2021 for DevOps [11]. ISO/IEC/IEEE 12207 helps consolidate and structure various software process categories within its 30 LCPs, as seen in Fig. 4.

- **Agreement processes** are concerned with collaboration and agreements with other organizations.
- **Organizational project-enabling processes** offer the environment required for project execution.
- **Technical management processes** refer to many facets of project management and are therefore executed at the project level.
- **Technical processes** describe the many processes or phases of a software product's life cycle, from defining stakeholder needs to software development.

The life cycle model framework of processes and activities is concerned with the all life cycle, which can be organized into stages, acting as a common reference for communication and understanding between stakeholders [14]. Agile development has proved to save time in market development, while developers and customers can work together rapidly. However, problems arose when the Agile operations team did not get cooperation from the developers to execute the operational processes. The DevOps movement is a mindset shift to solve the problem in Agile operations. *"DevOps is a full life cycle endeavor which gives equal consideration to each stage"* [11, p. 23]. Thus, it is a set of concepts and practices that improve stakeholder communication and collaboration for specifying, producing, improving, and operating software and systems products and services.

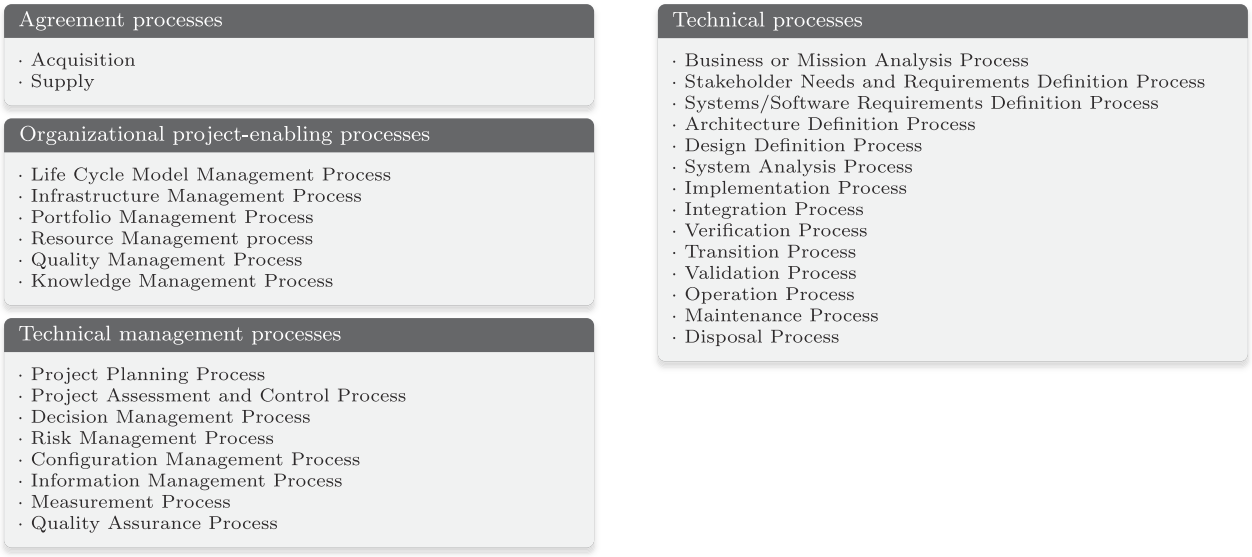


Fig. 4. Software Life Cycle Processes (LCPs).
Source: Adapted [14].

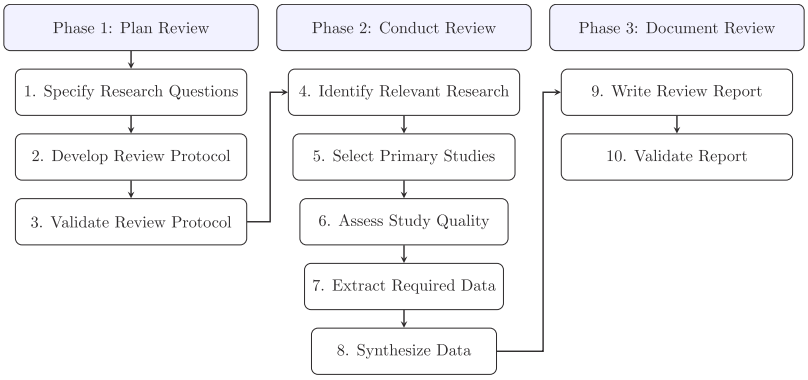


Fig. 5. The SLR process.
Source: Adapted from Kitchenham et al. [35].

3. Systematic literature review

This research focuses on DevOps Capabilities and Life Cycle Processes (LCPs) relying on the guidelines presented by Kitchenham et al. [34] to perform systematic literature reviews in SE. These guidelines are divided into three main phases: planning, conducting, and reporting the review, as shown in Fig. 5. In this section, each step of the SLR performed is detailed.

Since Kitchenham et al. (2004) [35] on SLRs, the use of this kind of review in SE and other scientific communities has become frequent for gathering evidence mainly from primary studies. The development of the research usually consists of three phases, as presented in Fig. 5.

1. **Planing**: Identifying the need for a SLR in order to summarize existing information about some phenomenon in a thorough and unbiased manner. Development and Validation of a Review Protocol presented in Section 3.1.
2. **Conducting**: Is the process of identifying relevant research, election of studies, study quality assessment, data extraction while monitoring progress and data synthesis shown in Section 3.2 and represented in Fig. 6.
3. **Documenting/Reporting**: In the last phase, a review report needs to be written in order to get validation from peer review as seen in Section 4.

3.1. Planning

The initial step of the SLR process involves developing and validating the review protocol. In the introduction, the importance of the study, including the problem, objectives, and research questions, is explained. This section describes how the study is being conducted and the steps taken to develop and validate the review protocol.

3.1.1. Review protocol

In order to find other studies, that may provide answers to the proposed research questions, a search was conducted in November 2022 using various keywords, based on DevOps and the related Life Cycle Processes (LCPs) [11,14].

Following is the resultant search string to be used in the search to retrieve the maximum number of relevant studies. The query is applied to the chosen datasets, which are also listed below.

- **Datasets**: The search engines used were, ACM Digital Library,[2] IEEE Xplore,[3] Science Direct,[4] Springer Link,[5] Wiley Online Library,[6] EBSCO.[7] Scopus[8] and Web of Science.[9]

- **Search String**: The following search string finds the word "DevOps" together with any of the other 31 words.

```
DevOps AND (
   "ISO/IEC 12207" OR
   "Acquisition" OR
   "Supply" OR
   "Life Cycle Model Management" OR
   "Infrastructure Management" OR
   "Portfolio Management" OR
   "Human Resource Management" OR
   "Quality Management" OR
   "Knowledge Management" OR
   "Project Planning" OR
   "Project Assessment and Control" OR
   "Decision Management" OR
   "Risk Management" OR
   "Configuration Management" OR
   "Information Management" OR
   "Measurement" OR
   "Quality Assurance" OR
   "Business or Mission Analysis" OR
   "Stakeholder Needs and Requirements Definition" OR
   "System/Software Requirements Definition" OR
   "Architecture Definition" OR
   "Design Definition" OR
   "System Analysis" OR
   "Implementation" OR
   "Integration" OR
   "Verification" OR
   "Transition" OR
   "Validation" OR
   "Operation" OR
   "Maintenance" OR
   "Disposal"
)
```

In the first phase, a preliminary set of papers is obtained. After the search is complete, inclusion and exclusion criteria shown in Table 1 are applied to refine the search results, during abstract screening. During this step, the abstracts are screened to evaluate the relevance they have to the research. Thereafter, snowballing is done to include any important and relevant material that might be referenced.

Following the process, relevant papers are read and organized in Zotero[10] reference manager to obtain the final selection of studies to perform the review as outlined in Section 3.2. Finally, systematic analysis and qualitative coding are performed in Qualcoder[11] Section 3.3 in order to extract data to spreadsheets[12] and return the answers to the research questions in Section 4.1 and Section 4.2. The document itself is then written in LaTex.[13]

Table 1
Inclusion and exclusion criteria applied in this research.

Inclusion Criteria	Exclusion Criteria
Written in English	Unidentified author
Published between 2017 and 2022	No publication date
Mention an LCP and DevOps Capability	Full-text not accessible
Peer-reviewed	Lack of rigor or validity (unreliable)
Engineering/Software Engineering	Non-engineering related

3.2. Conducting the SLR

This section presents an overview of the SLR process. Fig. 6 illustrates the selection procedure, which consists of four stages: Identification, Screening, Eligibility, and Inclusion, based on the PRISMA statement [36]. These stages also follow the guidelines for conducting an SLR [34,35] and are designed to ensure replicability and adherence to peer-review standards.

3.2.1. Identification of primary documents

In the initial phase, the search process involved querying the selected databases using the search string defined in the planning phase. A total of 25,979 studies were initially identified when searching by full text. To better filter this number, the search query was also applied to the title and abstract, resulting in 3,125 documents for easier identification, as shown in Fig. 6. Applying the inclusion and exclusion criteria seen in Table 1 of Section 3.1.1, the retrieved documents were filtered based on their relevance, publication date (January 2017 to November 2022), only related to software engineering, and have been peer-reviewed. This filtering process excluded 2,524 non-relevant documents. This was done by using the search engine of each database to exclude documents before 2017 (1,186), non-software engineering related (909) and no peer-reviewed documents (429). We remain with 601 documents in Step 3, Relevant after inclusion/exclusion criteria to be imported into the bibliographic reference manager Zotero, for the Quality Assessment and Eligibility process.

In Step 4, Zotero automated deduplication feature removes 131 duplicates using resulting in a final set of 470 unique documents.

3.2.2. Quality assessment and eligibility

This phase of the SLR targets high-quality, relevant studies to be included. After screening, titles and abstracts are reviewed for relevant studies. The Quality Assessment criteria were grounded on recency (after 2017), full-text available, methodological rigor/impartiality/validity (sound), sufficient relevance and data (enough to support results on LCPs and DevOps Capabilities).

In Step 5, a full-text review, when that was available, evaluated studies' methodological rigor, relevance, and quality. Documents without a full text, unsound or without sufficient relevance and data about LCP and DevOps Capabilities were excluded (377). The authors discussed and resolved any quality discrepancies on eligibility decisions through discussion to reach a consensus, captured in a spreadsheet. This step reduced the number to 93 documents after screening abstracts. In order to capture any relevant missing publications, the forward and backward iterative process of *snowballing* [37] was then applied to the references, of the screened literature, in Step 5. This process yielded 9 additional documents to be included, resulting in a final set of 102 full-text documents for assessment. The overall process is illustrated in Fig. 6.

[2] https://dl.acm.org
[3] https://ieeexplore.ieee.org
[4] https://www.sciencedirect.com
[5] https://link.springer.com
[6] https://onlinelibrary.wiley.com
[7] https://search.ebscohost.com
[8] https://www.scopus.com
[9] https://apps.webofknowledge.com
[10] https://zotero.org
[11] https://github.com/ccbogel/QualCoder
[12] https://www.libreoffice.org/discover/calc/
[13] https://www.latex-project.org/

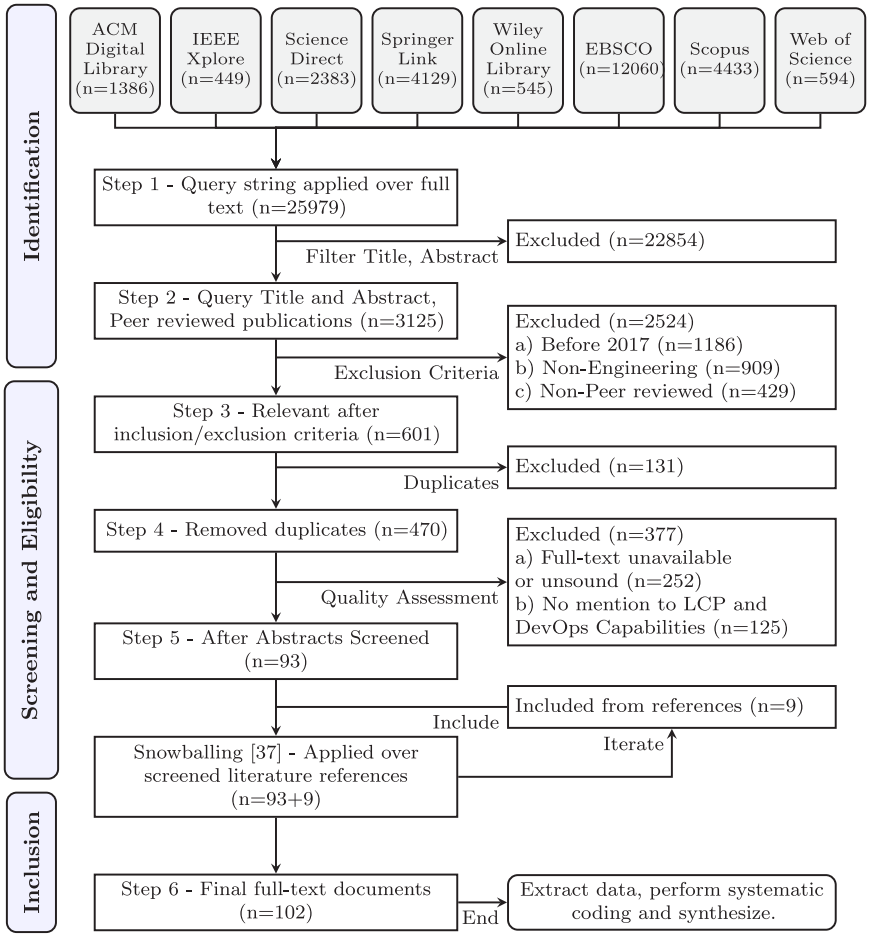


Fig. 6. Systematic review flow of information diagram for this study.
Source: Adapted [36].

3.2.3. Extraction of data

In Step 6, the flow ends with the extraction of data, performing systematic coding and synthesizing results over the 102 final documents. For the extraction process, the methodology involved systematic study within Zotero, as mentioned before, highlighting and extracting relevant parts of text and sections to export them and performing qualitative coding of papers within Qualcoder. This leads to identifying the relations in the study and common themes which are used in the discussion of this article. Data synthesis and results provide a comprehensive answer to the research question.

3.3. Data extraction analysis

The extraction phase involves locating and identifying relevant data for analysis. It allows for the combination of different categories of data to be synthesized. The systematic analysis phase follows qualitative coding.

3.3.1. Literature number of contributions

The number of contributions gathered from literature towards quality assessment derives from several databases, as seen in Table 2. This process is important to achieve the necessary quality for the data extraction phase.

This approach tries to reach the most databases possible and still maintain a feasible and large scope of academic publications in order to answer the research questions using qualitative coding analysis.

Fig. 7 shows the publications gathered for the SLR, categorized according to several ranking factors sourced from reputable academic sites, including Conference Ranks.[14] and Scimago Journal & Country Rank[15] The data reveals a focus on strong publications in high-quality outlets, including Q1 and Q2 journals, A/B conferences, and books.

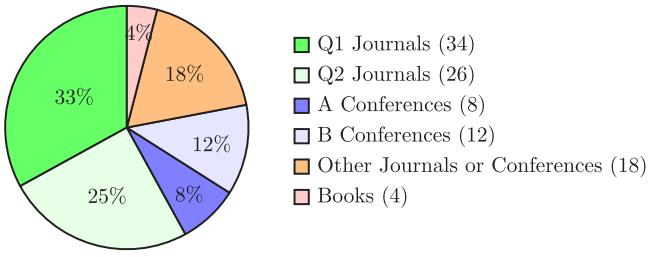


Fig. 7. Break down of publication quality based on ranking.

This is a reliable indicator of the research quality and impact achieved in this domain, given the substantial number of publications produced by these prominent institutions. Overall, the publications selected for review meet the required high standards of academic rigor and excellence, contributing to the advancement of the field.

[14] https://www.conferenceranks.com
[15] https://www.scimagojr.com

Table 2
Databases and steps used in the Systematic Literature Review (SLR) protocol.

Database	Step 1	Step 2	Step 3	Step 4	Step 5	Snowballing	Step 6
ACM Digital Library	1,386	140	85	85	13	0	13
IEEE Xplore	449	350	117	117	21	3	24
Science Direct	2383	110	38	38	14	0	14
Springer Link	4129	225	34	13	2	2	4
Wiley Online Library	545	17	14	14	5	0	5
EBSCO	12,060	736	87	65	11	1	12
Scopus	4433	1138	198	123	20	0	20
Web of Science	594	409	28	15	7	3	10
Total	25,979	3125	601	470	93	9	102

Step 1 = Query All fields, All documents
Step 2 = Query Title and Abstract, Peer reviewed publications
Step 3 = Relevant (inclusion/exclusion criteria) Table 1
Step 4 = After Removing duplicates
Step 5 = After Abstracts Screened
Snowballing = Applied over screened literature [37]
Step 6 = Full-text Document Assess.

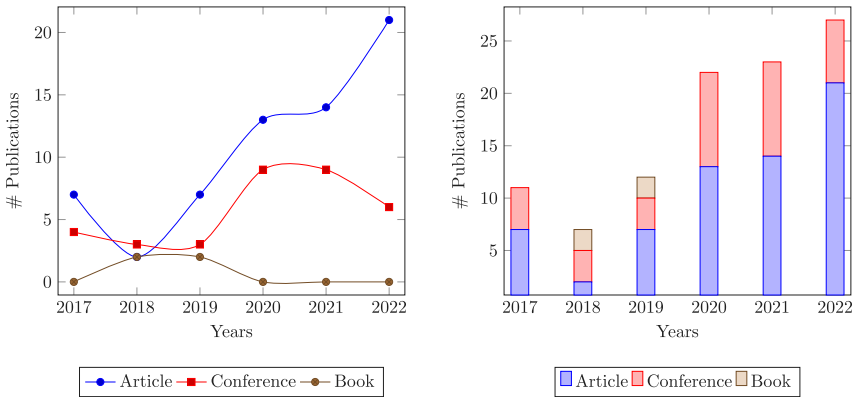


Fig. 8. Distribution of publications per type over the years.

3.3.2. Distribution of publications over the years

As can be seen in Fig. 8, it is also noted that there is an upward trend concerning publication numbers, with articles being the predominant types of them. On the other hand, conferences are other forms, while books occur less frequently. The growth in publication numbers indicates more interest in DevOps which could be a catalyst to improve both the software lifecycle as well as processes associated with it.

In summary, the analysis of the figures indicates a consistent growth in DevOps-related publications, with a significant increase in 2022. These publications mainly focus on the application of DevOps principles and practices to different stages of the SDLC. These findings suggest that DevOps is a crucial and expanding area of knowledge in software development, and ongoing research is valuable to explore its applications and benefits further.

4. Reporting the literature review

After extracting data and performing systematic qualitative coding, this reporting section answers this study's research question.

4.1. RQ1 - How do authors in scientific literature relate Software Life Cycle Processes to DevOps Capabilities?

This section addresses the first research question and discusses how authors relate Life Cycle Processes (LCPs) to DevOps Capabilities in literature.

Several papers propose ways to link DevOps with LCPs. For instance, Ali (2020) mentions that a hybrid DevOps process incorporating systematic reuse-based software development and management can reduce rework effort and costs while increasing productivity [38]. Similarly, Sánchez-Gordón (2018) suggests that new organizational structures and highly automated processes can connect LCPs with DevOps [39]. Although all the papers acknowledge the connection between DevOps and the software development process, none of them explicitly address how authors relate LCPs to DevOps Capabilities. Leite et al. (2019) highlight that DevOps involves collaborative and multidisciplinary efforts to automate software development [5], while Senapathi et al. (2018) found that the adoption of DevOps practices improves deployment frequency and communication between IT development and operations personnel [17]. To address this gap, Tables 3, 4, 5, and 6 present the identified relations between DevOps Capabilities and Life Cycle Processes [4,11].

This SLR utilizes the Life Cycle Processes (LCPs) from the Standard for DevOps [11] and categories of DevOps Capabilities proposed by Amaro et al. (2022) [4] as follows:

Cultural capabilities are those that focus on the people and teams involved in software development and delivery. They include things like cross-team collaboration and communication, a culture of learning and experimentation, and Free/Libre and Open Source Software (FLOSS) adoption.

Measurement capabilities are those that focus on collecting and analyzing data about software development and delivery. They include things like proactive monitoring, observability, auto-scaling, emergency response, proactive failure notification, monitoring systems to inform business decisions, working in progress limits, and visual management capabilities.

Process capabilities are those that focus on the way that software development and delivery are done. They include things like continuous improvement of processes and workflows, focus on people, process, and technology, working in small batches, lightweight change approval,

visibility of work in the value stream, customer focus and feedback, and a data-driven approach for improvements.

Technical capabilities are the ones that focus on the tools and technologies used in software development and delivery. They include things like continuous integration, continuous delivery/deployment automation, test automation, and environments, Version Control System (VCS), empowering teams to make decisions and changes, Configuration Management (CM), cloud infrastructure and cloud-native, artifacts versioning and registry, loosely coupled architecture, database change management, infrastructure as code, containerization, shift left on security, trunk based development, centralized log management, test data management, chaos engineering, and code maintainability.

For the remainder of this section, the relationship between each LCP (listed in Fig. 4) and the categories of DevOps Capabilities (as in Fig. 2) is detailed according to the analysis of the retrieved literature.

4.1.1. Agreement processes

LCP01. *Acquisition Process:* Obtaining a product or service that meets the acquirer's requirements [11].

Cultural capabilities: DevOps requires teams to work together [11], use FLOSS [40–42], and have a strong organizational culture that invests in tools and technologies to facilitate knowledge sharing and collaboration. [43,44].

Measurement capabilities: The acquisition of tools for proactive monitoring, observability, and autoscaling [11,45], emergency response, and failure notification [11], as well as visual management capabilities such as dashboards to support DevOps and the acquisition process [46] itself. Likewise, Risk Management should be considered to analyze, treat, and monitor risks [46] in these acquisitions.

Process capabilities: Focus on people, process, and technology [11,14,40], with a customer-centric approach [47] and data-driven approach for continuous improvement [24]. The acquisition process should align with DevOps principles, improving acquisition, in the life cycle [11,47].

Technical capabilities: Such as Continuous Integration & Continuous Delivery or Deployment (CI/CD) [48,49], Quality Assurance (QA) [14], VCSs [50], should empower teams [45], by acquiring flexible and decoupled software, supporting microservices [45], containers [51], and security best practices [11,14,43,49]. The process should be supported by operational data to assess benchmarks [49].

LCP02. *Supply Process:* Providing a product or service that meets the requirements agreement [11].

Cultural capabilities: While DevOps improves the technical and quality aspect of supply, it does so also by leveraging cross-team collaboration [52], communication, and experimentation [53,54] important for job satisfaction [5,21], with key enablers like FLOSS tools like Kubernetes [41,55,56], while reducing conflicts, failures [57], and deployment times.

Measurement capabilities: Supply products and services with evidence of DevOps measurement capabilities. Namely, observability, autoscaling, emergency response [5], to monitor systems and inform business decisions [58], integrated with visual management capabilities [59]. Moreover, cloud computing services offer productive, efficient, and reliable infrastructure with vulnerability scans [41,45,53,60], monitoring mechanisms and links to interconnect data centers with high performance, using FLOSS software on the cloud [40,57,61].

Process capabilities: Working in small batches [56] is essential for supplying change and updating systems. Visibility of the value stream helps control the life cycle [11,14], divided into four value streams: creating value in the "Dev" space, downstream delivery, and value creation in "Ops" monitoring and upstream feedback [58,62]. The lightweight change approval process is used by development teams to manage changes and supply customers with faster updates [46,63,64].

Technical capabilities: For the supply process, it is mentioned that DevOps Capabilities and tools such as continuous integration [48,65] using trunk-based development [42], continuous delivery [45,66], test automation [23,67], VCSs [41,64], and configuration management [14,21] are discussed. Security [17,68,69], cloud practices [40,70], microservices [57,71,72], database change management [66], containers [45,57,65,73], monitoring and logging services, empowering teams [23,63,74] and machine learning are also discussed. Finally, infrastructure is suggested as code [16], artifacts [65], and management of test data and centralized logs [45,61].

4.1.2. Organizational project-enabling processes

LCP03. *Life Cycle Model Management process:* Ensure the definition, maintenance, and availability of policies, processes, models, and procedures aligned with organizational objectives. Emphasizes the integration, collaboration, automation, and feedback systems crucial for DevOps [11].

Cultural capabilities: The adoption of FLOSS in DevOps facilitates continuous integration, delivery, and testing [85]. Chen et al. propose a platform that integrates FLOSS components to support research and development life cycle management [84]. The combination of DevOps and FLOSS promotes a culture of learning and experimentation in software development [79].

Measurement capabilities: Establishing Life Cycle Model Management aligned with DevOps Capabilities, project needs, and organizational policies are crucial [11]. Real-time analytics play a vital role in tracking application health and usage metrics, diagnosing problems, and integrating monitoring into application life cycle management using tools like Chef, Ansible, and Puppet [11,66].

Process capabilities: DevOps emphasizes continuous improvement of processes and workflows, particularly in alignment with security considerations and infrastructure management [11]. Lightweight change approval is a critical practice for managing software and system configurations in DevOps, involving configuration identification, status accounting, change control, and configuration audit [11].

Technical capabilities: DevOps relies on continuous integration, delivery, testing, and monitoring to ensure software reliability, availability, and security. Configuration Management is employed to control system elements and configurations throughout the product life cycle [11,46]. Database change management and containerization are also significant in DevOps [84,85]. Security considerations should align with life cycle model management, encompassing accountability, continuous tracking and evaluation, monitoring, and improvement of security performance [14,66,84].

LCP04. *Infrastructure Management process:* Provides and maintains the necessary infrastructure and services to support objectives throughout the lifecycle [11].

Cultural capabilities: Cultural capabilities in infrastructure management include cross-team collaboration, communication [75], and transformational leadership [88]. These capabilities support a learning culture, experimentation [5,17,56], and the adoption of FLOSS [41], enabling effective infrastructure management [80–82].

Measurement capabilities: Effective infrastructure management in DevOps requires proactive monitoring, observability, autoscaling [21,52,54,56], emergency response [5], visual management [45], and continuous delivery. Key aspects include artifacts and configuration management, infrastructure as code, containerization, and cloud services. Continuous monitoring of runtime performance, availability, scalability, resilience, reliability, metrics, alerting, and log management is essential [5,51,61].

Process capabilities: Legacy infrastructure systems pose challenges in DevOps due to factors like lack of automation, source code quality, and monitoring. Continuous delivery can help overcome these barriers, and successful DevOps implementation relies on frequent releases [5,40,63]. Effective configuration management of code and infrastructure and customer feedback play important roles [52].

Technical capabilities: Automation of infrastructure and software development is a critical aspect of DevOps [49,66,95]. Infrastructure

Table 3
Papers relating DevOps Capabilities [4] to Life Cycle Processes [11] 01–08.

		Agreement processes		Organizational project-enabling processes					
		LCP01	LCP02	LCP03	LCP04	LCP05	LCP06	LCP07	LCP08
C01	Cultural Capabilities	[11]	[52]		[75]	[52,75–77]	[5,46]	[78]	[11]
C02			[11,14,46,53,54]	[79]	[5,17,56,80–82]		[83]	[79]	[58,79]
C03		[40–42]	[41,55,56,59,69,73]	[84,85]	[41]	[80,86]		[40,42]	[87]
C04					[88]	[11]	[89]		
C05		[43,44]				[80]	[90]		
C06			[5,57]						
C07			[5,21]						
C08	Measurement	[11,45]	[5,14,21,40,41,45,51,53,57,60–62,69–71,91]	[11,66]	[5,21,51,52,54,56,61]	[46,86,92]	[44,93]	[14,23,77]	[17,21,87]
C09		[11]	[5]		[5]				
C10		[46]	[58]	[11]		[86]		[14,77,94]	
C11									
C12		[45]	[59]		[45]	[14,86]		[23]	
C13	Process Capabilities			[11]		[58]	[11,68]	[22,78]	[11]
C14		[11,14,40]					[40]		
C15			[56]					[95]	
C16			[11,14,46,63,64]	[11]	[5,40,63]	[11,14,86]		[63,77,96]	[22]
C17			[58,62]						
C18		[47]			[52]			[11]	
C19		[24]					[97]	[24]	
C20	Technical Capabilities	[48]	[5,21,23,48,65]	[84,85]	[5,49,66,71,95]	[5,66,77,95]	[68]	[11,22,42,77,78,96,98]	[49,75,80]
C21		[49]	[21,45,62,66]	[66,84,85]	[5,17,49,52,63,71]	[5,49,52,77,96]	[89,99]	[5,11,22,42,63,77,78,96]	[22,75,80]
C22		[14]	[23,67]		[5,49,67,75,95]	[75,95]	[67]	[22,100]	[14]
C23		[50]	[41,64]		[81,82,101]	[64]		[22,98]	
C24		[45]	[23,63,74]		[45,95,102]	[11,76,89,102]	[89,95]		
C25			[14,21,41,66]	[11,14,46]	[14,21,23,56,65,82]	[11,14,46,75,89]	[11]	[14]	[21]
C26			[21,40,53,56,61,70]		[17,51,56,88,95]	[41]			
C27			[65]						
C28		[45]	[5,21,40,45,50,51,57,70–73,103,104]	[84]	[45,63,105]	[105]		[63,106]	
C29			[66]	[66]					
C30			[16,52]		[11,17,23,56,63,65,66,80,82,88,101]	[64,66]		[63,79]	
C31		[51]	[5,40,45,50,56,57,65,71,73,103]	[85]	[5,41,65,80,82,86,101]	[41]		[79]	[75]
C32		[11,14,43,49]	[11,14,17,40,41,45,46,55,60,66,68–70,104,107]	[11,14,66,84]	[5,16,40,60,67,88,108]	[11,40,49,106]	[99,109]	[11,14,22]	[11,14,22]
C33			[42]						
C34			[40,45,61]					[110]	
C35			[66]						[111]
C36									
C37		[5]		[14]					

Legend: DevOps Capabilities C01 - Cross-team collaboration; C02 - Support learning and experimentation; C03 - Open source software adoption; C04 - Transformational leadership; C05 - Westrum organizational culture; C06 - Blameless Postmortems ; C07 - Job satisfaction; C08 - Monitoring, Observability, and autoscaling; C09 - Emergency response; C10 - Monitor systems to inform business decisions; C11 - Working in progress limits; C12 - Visual management Capabilities; C13 - Continuous Improvement of processes; C14 - Focus on people, process, and technology; C15 - Working in small batches; C16 - Lightweight change approval; C17 - Visibility of work in the value stream; C18 - Customer focus/feedback; C19 - Data-driven approach for improvements; C20 - Continuous Integration; C21 - Continuous Delivery automation; C22 - Test Automation and environments; C23 - Version Control System; C24 - Empower teams to make decisions; C25 - Configuration Management; C26 - Cloud infrastructure and cloud-native; C27 - Artifacts versioning and registry; C28 - Loosely coupled architecture; C29 - Database change management; C30 - Infrastructure as Code; C31 - Containerization; C32 - Shift left on security; C33 - Trunk-based development; C34 - Centralized log management; C35 - Test data management; C36 - Chaos Engineering; C37 - Code maintainability.
Life Cycle Processes LCP01 - Acquisition process; LCP02 - Supply process; LCP03 - Life Cycle Model Management process; LCP04 - Infrastructure Management process; LCP05 - Portfolio Management process; LCP06 - Human Resource Management process; LCP07 - Quality Management process; LCP08 - Knowledge Management process.

as Code (IaC) is a key practice that manages infrastructure components as programmable artifacts [11,23,63,101]. It automates system provisioning, deployment, and orchestration. DevOps utilizes various automation and management technologies for frequent and reliable releases [49,95], including containerization [41,80,82,86], configuration management [14,21,56], and microservices technologies [51,88].

LCP05. *Portfolio Management process:* Uses a central portfolio for continuous assessment, managing, and redirected investment to maintain strategic projects and ensure the organization's success [11].

Cultural capabilities: Organizations that use reusable assets should collaborate [76] and explore potential reuse opportunities [52,75,77]. FLOSS adoption improves portfolios and brings new tools [80,86]. Transformational leadership and portfolio risk management are key for managing changes during the transition to DevOps [11,80].

Measurement capabilities: Proactive monitoring, observability, and visual management improve Portfolio Management decision-making, resource efficiency, and strategic alignment [14,86]. These capabilities provide real-time insights, anticipate issues, optimize resources, and provide clear data visualization to ensure the portfolio meets organizational goals and adapts to future changes and challenges [46,92].

Process capabilities: Manage product lines to meet organizational or customer needs and objectives while supporting technology changes [11]. Implement lightweight changes supported by configuration management to maintain a product's life cycle [14,86]. Improve processes and workflows to achieve non-functional requirements and improve software [58].

Technical capabilities: In portfolio management, continuous integration [66,95], continuous delivery [49,77], test automation [75,95], VCSs [64], configuration management [14,46], cloud infrastructure [41], loosely coupled architecture [105], infrastructure as code [64,66], containerization [41], and shift left on security [11,40,49,106] are highlighted. These capabilities require automation, collaboration, and closer feedback between teams, product, and customers [75,86]. Challenges related to continuous deployment adoption include team coordination, customer adoption, feature discovery, plugin management, and scaling CI tools [5,77].

LCP06. *Human Resource Management process:* Ensures the provision and maintenance of human resources and competencies [11].

Cultural capabilities: Implementing cross-functional teams in DevOps is important, but challenges exist due to the shortage of operators with development skills [5,46]. Transformational leadership and a culture that supports learning and experimentation are needed for DevOps adaptation [83,89,90]. Reallocation of resources and adjusting organizational cultures and processes are challenges and benefits of changing work structures [90].

Measurement capabilities: Alignment between HR management and DevOps measurement capabilities is crucial, including identifying necessary skills and providing performance data [44]. Monitoring provides insights for resource management, but expertise in metrics monitoring is important for successful DevOps adoption [93].

Process capabilities: Continuous improvement in software development and related organizational functions is emphasized in DevOps [11,68]. Continuous integration and reducing organizational silos are key aspects [40]. HR management should support teams and ensure personnel has the flexibility and capacity to adopt new technologies and methods [97].

Technical capabilities: Continuous integration, continuous delivery, test automation, and configuration management provide insights for managing HR and development time in DevOps [11,67,68,97]. Hiring skilled professionals with DevOps knowledge is essential. Adapting HR and automating infrastructure is necessary for operational contexts and aligning with DevOps processes [40,89,95]. Security shift left is also important [99,109].

LCP07. *Quality Management process:* Assures that products, services, and implementations meet quality objectives and achieve customer satisfaction [11].

Cultural capabilities: DevOps improves software development practices and faces challenges with FLOSS adoption, including license, quality, and security concerns. Cross-team collaboration, learning culture, and open-source tool adoption like SonarQube address these challenges and support quality management [40,42,78,79].

Measurement capabilities: Monitoring, measuring customer satisfaction, continuous monitoring, and visualization of product and process quality inform business decisions in quality management [14,23,77,94].

Process capabilities: Continuous improvement, small batches, lightweight change approval, data-driven decisions, and managing corrections, lessons learned, and customer feedback drive software development and release management [11,22,24,63,77,78,95,96].

Technical capabilities: Standardizing quality management with *Continuous Integration* (CI), using FLOSS tools like Jenkins, Cucumber, JUnit, GIT, and Selenium, practicing QA and testing, and addressing log management, code quality analysis, and security contribute to successful technology-driven transformation in DevOps [11,22,42,98,100].

LCP08. *Knowledge Management process:* Enables the organization to leverage existing knowledge for opportunities [11].

Cultural capabilities: Knowledge management can support cross-functional communication, practical lessons learned, experimentation, learning culture, and FLOSS software adoption support DevOps [11,58,79,87].

Measurement capabilities: Upskilling the team, training, and hiring for observability, alert handling, autoscaling, and monitoring tools are crucial for DevOps adoption [17,21,87].

Process capabilities: Sharing lessons and artifacts, updating policies and processes, securing knowledge management environments, and implementing CI/CD with lightweight change approval improve workflows and artifact delivery [11,22].

Technical capabilities: Implementing CI/CD, test automation, CM, containerization, shift left on security, and test data management improves software development, infrastructure, quality, security, and operations while promoting team knowledge sharing [14,22,49,75,80,111]. Safeguarding and securing knowledge and skills is also important [14].

4.1.3. Technical management processes

LCP09. *Project Planning:* Produces workable plans identifying outputs, tasks, schedules, acceptance criteria, and resources [11].

Cultural capabilities: DevOps roles emphasize improved cross-team collaboration, project planning [77] with FLOSS and a business-focused plan, transformational leadership, blameless postmortems, and a culture of feedback and improvement [14,40,47,79,112].

Measurement capabilities: Defining common objectives, management priorities, roles, and responsibilities for cross-functional DevOps teams, and using proactive monitoring, observability, and autoscaling to inform business decisions and system adjustments [14,40].

Process capabilities: Continuous improvement, integrating lessons learned from QA processes, project planning, lightweight change approval, and customer feedback support system adoption and project success [11,47,112,113].

Technical capabilities: Collaboration, continuous integration [21], adherence to the project plan for testing and quality processes, automated testing, CM, security planning and coordination, transition planning, coordination of CM plans, and loosely coupled architecture are important technical aspects of DevOps [11,14,40,47].

LCP10. *Project Assessment and Control:* Monitors project alignment, performance, and provides corrective actions [11].

Cultural capabilities: Supporting learning and experimentation in project assessment and control, particularly with VCS, enables tracking project changes and learning from past mistakes [81].

Measurement capabilities: Proactive monitoring, observability, autoscaling, and visual management aid project control by providing infrastructure provisioning, validation, monitoring, and presenting project tracking information through dashboards [14,64].

Process capabilities: Improvement of project tracking, assessment, and control processes based on data and project needs [11], data-driven approach, lightweight change approval processes, and continuous project monitoring and assessment are essential for effective project management and control [14].

Technical capabilities: Automation in continuous integration and delivery [21], continuous testing, VCSs like Git, loosely coupled architecture, and shift left on security contribute to improved project assessment and control processes and the overall quality of the project [11,14,41,64,81,120].

LCP11. *Decision Management:* Structured framework for making informed decisions throughout the lifecycle [11].

Cultural capabilities: Cross-team collaboration [11,77], learning culture, experimentation, and a performance-oriented organizational culture are crucial for decision management in DevOps, facilitating knowledge sharing and the development of microservices at scale [14,23,40].

Measurement capabilities: Proactive monitoring, observability, autoscaling, and visual management support informed business decisions and communication of results to stakeholders in decision management processes [14,23,40,41].

Process capabilities: Data-driven decision-making, lightweight change approval processes, and standardization of tasks and processes drive continuous improvement and enhance the quality of decisions in DevOps [11,14,23,119].

Technical capabilities: Effective decision management is crucial for successful automation in Continuous Integration and Delivery, Test Automation, CM, cloud infrastructure utilization, and adopting a loosely coupled architecture. Shifting left on security ensures secure software delivery [11,14,49,53,66,102].

LCP12. *Risk Management:* Continuously identifies and manages risks associated with acquisition, development, maintenance, or operation [11].

Cultural capabilities: Transformational leadership is crucial for managing culture, tools, processes, and practices during the DevOps transition [11]. DevOps Risk Management, an automated and continuous process, empowers leaders to identify, analyze, and mitigate risks that may affect project success [71].

Measurement capabilities: Risk management in DevOps involves continuous monitoring, vulnerability scanning, and testing within a risk management framework [53]. Proactive monitoring, observability, autoscaling, and visual management capabilities inform business decisions, while working in progress limits manage workflows effectively [11,46,108].

Process capabilities: DevOps risk management focuses on continuous process improvement, lightweight change approval, and customer feedback [108]. Change management is crucial for DevOps adoption, and stakeholder feedback loops ensure continuous improvement and timely delivery [11,14].

Technical capabilities: Automation is essential in DevOps for continuous integration, delivery, deployment, and operations [47,48]. Risk management, QA, testing, and CM play vital roles in building secure and verifiable systems. Empowering teams to make decisions and adhering to risk management frameworks and processes minimize adverse effects on the organization and stakeholders [11,14,43,53,79,90,108,109].

LCP13. *Configuration Management:* Control and manage system elements and configurations across the lifecycle [11].

Cultural capabilities: Cross-team collaboration, communication, a learning culture, and FLOSS adoption are crucial for improving DevOps. CM reduces errors, improves security, and standardizes environments.

Table 4
Papers relating DevOps Capabilities [4] to Life Cycle Processes [11] 09–16.

		Technical management processes (1 of 2)							
		LCP09	LCP10	LCP11	LCP12	LCP13	LCP14	LCP015	LCP16
C01	Cultural Capabilities	[40,77]		[11,40,77]		[75,89,114]	[11]	[11]	[17,48,76,87,106]
C02			[81]	[14,23]		[5,17,80,82,115]	[42,58]	[53,83]	[106]
C03		[14,112,113]				[14,66]	[42,110,116]	[38]	[14]
C04		[79]			[11,71]				[50,71]
C05				[40]					[11,22]
C06		[47]							
C07									
C08	Measurement	[40]	[14,64]	[14,40]	[11,53,108]	[5,14,21,41,61,77,84,114]	[11,21,41,91]	[11,14,57,61,75]	[11,21,22,39,44,50,79,117]
C09					[11]	[5]			[17]
C10		[14]		[14,41]	[46]	[14,21,41]	[14,41]	[11,14,47,99]	[14]
C11					[11]			[11,118]	[11,118]
C12			[11]	[23]	[108]	[66]		[11]	
C13	Process Capabilities	[11]	[11]	[11,23,119]	[11,108]		[42,68]	[16]	[11,69]
C14								[14]	
C15									[11]
C16		[11,112,113]	[14]	[11,14]	[11,14]	[5,11,14]	[11,14,41,42]	[11]	[24,63,79,98]
C17							[23]	[11]	[47]
C18		[47]			[11]				[63,114]
C19			[11]	[11,23,119]		[115]			[11,24]
C20	Technical Capabilities	[21]	[11,21]	[49,102]	[11,48]	[5,21–23,69,72,98]	[42]		[11,16,22,23,48,50,71,120]
C21		[21]	[11,21]	[5,66,102,119]	[11]	[5,11,17,21,66,72,87]	[11,42,44,116]		[11,16,22,39,66,69,82,106,114,120]
C22		[11]	[120]	[11]	[47]	[5,14,22,75,87,100]		[14]	[11,17,22,23,71,92,100,106,121]
C23			[64,81]			[21]	[116]		[22,106]
C24				[11,77]	[11,79,108]		[11]	[99]	[11,17]
C25		[11,14]	[14]	[14]	[14,46]	[5,11,14,21,46,65,82,89,93]	[11,14,21]	[14,61]	[11,14,21,48,69]
C26				[53,66]	[53]	[41,56,93]			[66]
C27						[66]			
C28		[21]	[21]	[40]		[5,21]	[42]	[50,57]	[21,50,106,117]
C29						[66]	[41]		
C30						[5,21,23,41,56,65,66,69,80]	[91]		[50,79,91]
C31						[5,41,65,69,77,80,82]	[116]	[78]	[50,79,120]
C32		[11,40,47]	[11,41]	[11,14]	[11,14,43,47,90,109]	[5,11,14,21,41,66,114]	[14]	[11,99]	[11,21,22,50,69,107]
C33									[91]
C34						[110]	[42,110]		
C35									[11]
C36					[11]				[11]
C37						[23]			

Legend: DevOps Capabilities C01–C37 mentioned in Table 3.
Life Cycle Processes LCP09 - Project Planning; LCP10 - Project Assessment and Control; LCP11 - Decision Management; LCP12 - Risk Management; LCP13 - Configuration Management; LCP14 - Information Management; LCP15 - Measurement; LCP16 - Quality Assurance.

Continuous experimentation and automation are essential for successful DevOps [5,14,17,66,75,80,82,89,114,115].

Measurement capabilities: CM ensures project integrity using tools like Puppet and Chef for configuration automation and monitoring [61,84]. DevOps requires technical skills in log analysis, containerization, and management skills in planning and time monitoring. Continuous planning, feedback, and monitoring are essential [14,21,41,77,114].

Process capabilities: Source code management, build, release, deployment engineering, and application lifecycle management are crucial. Configuration audits verify authorized changes, and lightweight, data-driven change approvals are implemented [115]. CM also includes managing changes to organizational procedures and business workflow [5,11,14].

Technical capabilities: CM tools like Chef, Ansible, and Puppet Labs support platform configuration in cloud computing environments. Teams improve accessibility to resources for automatic testing, server configuration, and DevOps practices such as CI/CD [41,80]. CM is intrinsic to managing and controlling system elements and configurations throughout the lifecycle [5,11,14,21,23,46,56,65,66,69,82,89,93,114].

LCP14. *Information Management:* Generate, obtain, confirm, transform, retain, retrieve, disseminate and dispose of information, to designated stakeholders [11].

Cultural capabilities: Effective information management requires appropriate tool selection, communication channels [11], and comprehensive log analysis platforms. FLOSS like ELK facilitates runtime information collection and analysis [110,116]. Cross-team collaboration, a learning culture that supports experimentation, and communication are essential for successful implementation [42,58].

Measurement capabilities: Proactive monitoring, observability, and autoscaling ensure peak system performance [11,21,91]. Real-time system monitoring and data collection enable issue identification and informed decision-making. These practices improve organizational performance [14,41].

Process capabilities: Continuous improvement, lightweight change approval, and work visibility are critical for information management [23]. Continuous integration between software development and operational deployment and continuous assessment and improvement of the link between business strategy and software development are vital [42,68]. Effective log management and continuous integration processes adapt to changing requirements [11,14].

Technical capabilities: Information management establishes procedures for handling information products, manages configuration changes, and implements central logging for runtime behavior analysis [11,41,110]. It utilizes tools like VCSs, continuous deployment capabilities, and unified environments [14,21,42,44,91,116].

LCP15. *Measurement:* Collects and analyzes data to support management decisions and demonstrate quality [11].

Cultural capabilities: Effective communication, collaboration, and feedback loops are essential for embedding Quality Management into DevOps [11]. FLOSS solutions and deterministic metric measurement enhance performance. Challenges include experiment pre-processing and ensuring external validity [38,53,83].

Measurement capabilities: Automated checks, tests, and monitoring measure Service Level Indicators (SLIs) and assess deliverable readiness [57,61,75]. Real-time feedback and measurement-driven design enable continuous improvement, progress tracking, and meeting Service Level Agreements (SLAs). Proper measurement procedures, instrumentation, and data integrity support evidence-based decision-making and quality improvement [14,99,118].

Process capabilities: Measurement is essential for continuous improvement in DevOps [16]. Organizations must ensure data quality and integrity, design and configure instrumentation for metrics collection, and use measurements to manage changes in the application life cycle. Measurements should be implemented across all roles to support problem analysis and process improvement [11,14].

Technical capabilities: Measurement processes using test automation and calibrated verification environments enable system suitability [11,14]. Bidirectional traceability is important for requirements management, architecture, design, CM, and information management. Measurement and observability systems enable data collection and processing for decision-making and change management processes [50,57,61,78,99].

LCP16. *Quality Assurance:* Ensure the application of the Quality Management process, as well as the organization's policies and procedures [11].

Cultural capabilities: Collaboration, communication, and a learning culture enhance QA in DevOps [17,48,76,87,106]. Adopting FLOSS, transformational leadership, and constant experimentation contribute to higher quality and increased transparency [14,50,71]. Test automation is crucial to meet QA requirements in dynamic DevOps cycles [11, 22].

Measurement capabilities: Real-time monitoring and automated testing play crucial roles in DevOps QA. Usage and measurement data enable continuous assessment, test case generation, metric computation, and result visualization [11,21,22,39,44,50,79,117]. Automated monitoring and testing identify irregularities, variances, and information gaps to ensure SLOs are met and quality is assured [14,17, 118].

Process capabilities: Continuous improvement, customer feedback, and a data-driven approach are crucial in QA [11,69]. DevOps practices reduce rework and errors, leading to faster time-to-market and improved product quality [24,79,98]. Incorporating new tools and methods addresses risk areas and process gaps [47,63,114].

Technical capabilities: CI and DevOps automation tools enable QA in software development [16,23,39,48,50,66,69,82,114]. Test automation, risk management, and testing are emphasized in DevOps to achieve accelerated velocity and continuous delivery. QA oversees CI, adapts test automation to DevOps cycles, and ensures code quality. Continuous QA and testing are essential to reduce rework and waste [22,79,91,106,107,117,120].

4.1.4. Technical processes

LCP17. *Business or Mission Analysis:* Focuses on defining problems, characterizing solutions, and identifying potential solutions [11].

Cultural capabilities: A learning culture and experimentation are critical in business analysis. Feature analytics evaluate cost, usage, and return on investment [68]. FLOSS adoption solves performance bottlenecks, and integrating processes is essential for implementation [40, 84].

Measurement capabilities: Monitoring and mapping software nonfunctional requirements to business objectives to inform decision-making [11,46,58]. Visual management and proactive monitoring guide business decisions and adapt to system changes. Feedback from operational monitoring aids customer support [14].

Process capabilities: Capability models drive continuous improvement in Business/Mission Analysis processes. Effective monitoring of system performance and customer support is crucial. Operational gaps inform process changes [47,75]. Lightweight change approval supports continuous improvement [46].

Technical capabilities: Business or mission analysis benefits organizations by identifying key goals and strategies [11,14]. It explores internal and external factors impacting performance, improves efficiency, effectiveness, and success [47,49,75,108]. Analysis considers market trends, customer needs, and competitive pressures. The goal is to develop a clear understanding of the organization's mission and identify growth opportunities [40,45,68,84,89].

LCP18. *Stakeholder Needs and Requirements Definition:* Identifies requirements for a system to meet users and stakeholders' needs [11].

Cultural capabilities: Agile methods prioritize teamwork and iterative delivery. Continuous experimentation through small field experiments learns about customer needs. FLOSS and adapting to changing customer needs to ensure stakeholder satisfaction [14,86]. Organizational culture affects job satisfaction and quality [44,90].

Measurement capabilities: Identifying stakeholder needs and requirements is critical [77,86]. Effective communication ensures an understanding of stakeholder expectations, leading to solutions that meet requirements [14]. Gathering and analyzing stakeholder feedback informs decision-making and resource prioritization, increasing stakeholder satisfaction [14,23].

Process capabilities: Prioritizing people and interactions, customer collaboration, and responsiveness to change are essential in meeting stakeholder needs and improving processes [23,62]. Data-driven decision-making and rapid experimentation guide product development and foster innovation [14,46,62].

Technical capabilities: Understanding stakeholder needs and requirements is crucial for project success. Identifying expectations and concerns of stakeholders and incorporating them into planning and development processes [23,49]. Effective stakeholder engagement improves outcomes, builds trust, and fosters relationships [11,14]. Regularly reviewing stakeholder needs ensures ongoing satisfaction [22,41,67, 77].

LCP19. *System/Software Requirements Definition:* Transforms stakeholder views into technical solutions, creating measurable system requirements [11].

Cultural capabilities: Clear communication, collaboration, and attention to detail are crucial in defining system and software requirements [11,40,44,90]. Effective requirements definition ensures project success and avoids costly errors and delays [23,56,83].

Measurement capabilities: Incorporating measurement capabilities early in requirements design helps identify and mitigate risks and discrepancies [14,41,77,86,98,123]. Keeping the project on schedule and within scope prevents costly changes and rework later in the development cycle [11,21,51,57]. AI and ML with runtime data are used in recent projects [119] to translate stakeholder opinions into measurable system requirements. Levy (2022) suggests that using this data ensures that needs are based on real operational facts [86].

Process capabilities: Properly defining requirements involves a continuously improving process of analyzing stakeholders' needs, setting realistic goals, and facilitating communication among team members [23,86,98]. It ensures successful project outcomes and reduces the risk of errors and misunderstandings [11,14,46].

Technical capabilities: Like the need for CI/CD, proper testing, or CM are important for successfully transforming views to requirements software development, Technical identifying, analyzing, and prioritizing stakeholders' needs and constraints, and documenting them clearly [48, 66,68,69,79,124]. Effective requirements definition ensures meeting user needs, on-time and within-budget delivery of high-quality products. It requires collaboration, communication, and appropriate techniques and tools throughout the development lifecycle [22,23,67,70, 72,84,97,98,104,114]

LCP20. *Architecture Definition:* Generates system architecture alternatives to satisfy requirements [11].

Cultural capabilities: A learning culture, cross-team collaboration, and FLOSS adoption enhance the Architecture Definition process [14, 67,77]. Proof-of-Concept experiments and mathematical modeling provide confidence in system requirements and design [11,41,46,51].

Measurement capabilities: Cloud application architecture design requires integration of coding, testing, packaging, and monitoring activities [11,38,41,71,77]. Refactoring monolithic applications based on DevOps principles improves collaboration, scalability, and observability [51,77,86].

Process capabilities: Modularity, scalability, and upgradability address stakeholder needs in the Architecture Definition process [14,77]. Customer feedback, data-driven approaches, and integrated processes facilitate efficient change management [1,47].

Technical capabilities: DevOps practices like CI/CD, Test Automation [67], CM, Cloud infrastructure, and loosely coupled architecture improve Architecture Definition [5,11,63,64]. They enhance collaboration, scalability, flexibility, and alignment with business planning [14, 51,56,77,112,113].

LCP21. *Design Definition:* Provides detailed system and element data for implementation [11].

Cultural capabilities: Experimentation, FLOSS adoption, blameless postmortems, and job satisfaction support a learning culture, innovation, agility, and high-quality design and development [23,43,51,

Table 5
Papers relating DevOps Capabilities [4] to Life Cycle Processes [11] 17–23.

		Technical processes						
		LCP17	LCP18	LCP19	LCP20	LCP21	LCP22	LCP23
C01	Cultural Capabilities		[75]	[14,40,75,77]	[77]			[11,47]
C02		[68]	[23,83]	[11,14,23,46,51,53,56,83]	[11,14,41,46,51]	[11,14,23,46,66,88,115]	[11,14,23,42,46,51,73,88]	[38]
C03		[40,84]	[14,86]	[55,86]	[14,67]	[43,51,107,111,116,122]		[11]
C04				[11]			[122]	
C05			[44,90]	[40,44,90]				[47,69,89]
C06						[71]		
C07			[23]	[23]		[23]		
C08	Measurement	[11,46,58]	[11,14,23,77,86]	[11,14,21,38,40,41,51,57,58,77,86,98,119,123]	[11,38,41,51,71,77,86]	[11,14,21,46,51,57,61,71,119,123]	[14,50,53,73,110,115,119]	[11,38,47]
C09						[11]		
C10		[14]	[14]	[14,40]	[77]	[21]		
C11			[23]					
C12		[14]	[14,23]	[14]		[23]		
C13	Process Capabilities	[47,75]	[62]	[5,62,86,98]		[62,69]	[46]	[69]
C14			[23]	[23]				
C15				[41]			[22]	[38,47,90]
C16		[46]	[14,23,46,86]	[11,14,46,86]	[14,77]	[111]		[11,89]
C17				[11,23]		[44,62]		
C18				[11]	[47]			[47,52]
C19			[23]	[23]	[1]	[23]		[1]
C20	Technical Capabilities	[49,108]	[23,49,72]	[11,49,65,66,70,72,84,120,123]	[64]	[21,44,70]	[42,71,120,124,125]	[11,48,49,95]
C21		[47,75,105,108]	[49,72,97,124]	[5,23,49,58,65,70,72,79,97,98,120,123,124]	[5,11,63]	[21,40,44,49,70,71]	[42,58,71,115,120,124,125]	[11,48,105]
C22		[23]	[48,67]	[11,48,65,69,122]	[67]	[23,115,123]	[120]	[14,50,67]
C23				[66]				[59]
C24			[48,75]	[40,48,68,75]		[49]		
C25			[23,41]	[14,41]	[14,56,69]	[14]		[11,14,46]
C26				[40,53,57]	[95,110]	[21,88,119]	[53]	
C27				[11]				[11]
C28		[45,105]		[21,40,45,59,104,117]	[5,21,51,59,84,86,104]	[21,45,49–51,59,73,86,105]	[22,42,57,73,117]	[105]
C29								[50,95,120]
C30				[11,41,65]	[56]		[71,82]	[17,56]
C31		[105]		[65,104]	[21,41,51]	[21,41,49,51,54,73,119]	[51,73,124]	[54,95]
C32		[11,14,40,47,68,75,84,89]	[11,14,22,41,67,77]	[11,14,21,22,40,41,60,65,67,70,77,84,114,123]	[11,14,21,77,112,113]	[11,14,21,40,109,112,113]	[14,74,109,124]	[11,14,24,40,47,54,99,107,114,120,125]
C33								[95]
C34				[11,110]	[51,110]		[42,110]	[110]
C35			[11]	[11]		[11]		
C36								
C37							[42]	[121]

Legend: DevOps Capabilities C01-C37 mentioned in Table 3.
Life Cycle Processes LCP17 - Business or Mission Analysis; LCP18 - Stakeholder Needs and Requirements Definition; LCP19 - System/Software Requirements Definition; LCP20 - Architecture Definition; LCP21 - Design Definition; LCP22 - System Analysis; LCP23 - Implementation.

71,111]. Specifically, postmortems are well documented Root Cause Analysiss (RCAs) that help identify design failures without assigning blame, promoting innovation, agility, and quality improvement. They encourage collaboration between designers, developers, and operations teams.

Measurement capabilities: Monitoring, data collection, and automated telemetry points are crucial in Design Definition. They support continuous software and system engineering [21,51,57,61,71,123], and enable control systems to maintain system goals based on measurements obtained. Smart health monitoring systems enable ubiquitous monitoring [21,23].

Process capabilities: Continuous improvement, DevOps principles, lightweight change approval, data-driven approaches, and rapid experimentation are essential in Design Definition [69,111,119]. Organizational and technical metrics track continuous improvement, while work visibility in the value stream defines system design. Rapid and continuous experimentation accelerates innovation [23,44,62].

Technical capabilities: Continuous integration, deployment practices, architecture agility, CM, security shifting left, system design, and testing are crucial in Design Definition. They enable rapid continuous delivery, functionality improvement, and transformation of system design [14,21,88,119]. CM maintains project integrity, and system design predicts cloud infrastructure properties [11,41,45,54,73,86,105,109,113].

LCP22. *System Analysis:* Supports decision-making with rigorous data and information [11].

Cultural capabilities: System analysis encompasses mathematical an analysis, modeling, simulation, and experimentation [11,14,42,73]. It provides confidence in system requirements, architecture, and design, and aims to understand the trade space and critical quality characteristics of a system [23,51,88,122].

Measurement capabilities: Monitoring enables dynamic observation and analysis of system data for runtime and design time [50,73,115]. It provides metrics for system status, and future cloud solution impact, and facilitates debugging and problem-solving for distributed systems [14,53,110,119].

Process capabilities: Microservices architecture in DevOps and *Continuous Delivery or Deployment* (CD) enables frequent software feature delivery and improves quality attributes. ISO/IEC 29110 reinforces DevOps processes [46]. Attention is given to quality attributes like ease of deployment, security, modifiability, and ability to be monitored in CD architectures [22].

Technical capabilities: System analysis involves various tools and methodologies, including CI/CD, Test Automation, and Containerization [51,58,115,120,125]. Centralizing log management, maintaining code quality, and implementing early security measures are crucial. Challenges and specialized analysis techniques for microservices architecture are discussed [22,42,57,73,74,82,109,124].

LCP23. *Implementation:* Realizes specified system elements from requirements and design [11].

Cultural capabilities: Implementing DevOps software and systems requires a collaborative approach across different teams, adoption of FLOSS, conducting experiments and fostering a learning culture. However, success can be hindered by communication problems, team frustrations and uncoordinated activities can undermine success [11,38,47]. Including such challenges like organizational culture, infrastructure, legacy systems and lack of automation. [47,69,89].

Measurement capabilities: Implementing involves several Quality Management (QM) procedures like consistent policies, the practice of evidencing quality in everything and automated code monitoring to detect non-compliant code [47]. Planning for monitoring, allocation of resources are relevant activities for developing good quality and evading issues. Traceability, feedback mechanisms and measurement are important elements [11,47].

Process capabilities: The tools and teams are integrated by the incorporation of all the necessary components for continuous integration and deployment [69], making them work hand in hand from the start to the end of an application process. The repetitive tasks are processed automatically [38,90] and customer feedback contributes to improvements [47,52]. Data-driven approaches, testing practices, and change control protocols assure quality and efficiency [1,11,89].

Technical capabilities: A rigorous process is required for DevOps implementation, which includes continuous integration, testing, and

Table 6
Papers relating DevOps Capabilities [4] to Life Cycle Processes [11] 24–30.

		Technical processes						
		LCP24	LCP25	LCP26	LCP27	LCP28	LCP29	LCP30
C01	Cultural Capabilities	[14,92,100,114]	[11,47]		[48,52,58,75,77,82,94,114]	[11,52,66,99]	[52,63,66]	[11,89]
C02					[23,38,46,51,58,68,81,87,115,128,129]	[88,115]	[14,38,81,88,115]	[101]
C03		[11,40,41,68,116]			[14,40,81]	[40,41]	[24,84,126]	[101]
C04						[40,87]		
C05						[75,89]	[43,89]	
C06				[11,47]	[11,14]	[14]		
C07		[49]					[50]	
C08	Measurement	[11,17,23,41,45,57,61,71,114,123,126]	[11]	[57,71]	[11,51,57,61,64,69,95,114,115,119,122,123,128]	[5,14,40,41,57,60,72,91,106,108,119,123,128]	[11,45,46,57,61,119]	[54]
C09						[5,40]		
C10		[14,94]			[94,119]	[47,75,106]		
C11					[89,127]	[118]		
C12						[108]		
C13	Process Capabilities	[11,69]	[11]		[23,62,94]	[43]	[11,23]	[11]
C14		[40,103]				[71]	[14,23]	
C15							[38,82]	
C16		[11,14,42,96]	[14,89]	[14]	[14,41,66,79,98]	[40,86,120]	[11,14]	[11]
C17		[11]			[62]			
C18		[23,47,83]		[47]	[58]	[11]	[38,83,90]	
C19		[11]						
C20	Technical Capabilities	[11,23,40,48,49,68,69,72,91,96,106,118,126,127]	[11]		[11,23,40,54,65,68,71,81,87,89,94,97,123,125,127,129]	[72,118]	[11,16,45,72,90,95,97,104,108,126]	[66,101]
C21		[11,17,23,48,67,83,91,96,97,103,115,118]			[11,23,52,58,62,65,79,87,88,94,97,105,120,123,124,127]	[40,66,71,72,86,99,118,119,129]	[11,16,17,22,45,63,66,72,90,104,108]	[66]
C22		[11,14,67,71,118,125]	[11,90]	[14]	[11,14,78,82,89,95,120,123]	[22,23,95,103,115]	[11]	[78]
C23		[121]	[11,125]		[64,81,95]	[40]	[54,58]	[54]
C24		[126]			[58]	[108,129]	[63,97,102,109]	
C25		[11,14,94]	[11,14,48]	[11,14]	[11,14,94]	[14,63]	[14]	[11,14]
C26			[90]		[57,81,107]	[40]		[41]
C27		[48]			[95]	[50]	[5,38]	
C28		[11,49,70,71]	[90]		[45,50,57]	[57]	[45,57,72,104]	[72]
C29					[66]		[93]	
C30		[41,69,91]		[79]	[11,69,82,95]	[41,56,99]	[11,17,126]	[5,54,91]
C31		[42,69,71]	[90]	[79]	[57,65,81]	[62,84,111]	[57,60,84,104,110,128]	[5,54,72,104]
C32		[11,14,40,47,67,70,72,93,103,114,115]	[11,14]	[47]	[11,14,40,46,60,69,70,82,88–90,99,104,109,114,124]	[11,14,40,72,87,91,108]	[11,14,16,40,72,90,104]	[11,14,46,104,109]
C33		[123]			[95]		[95]	[101]
C34					[110]	[40]	[110]	
C35		[11]	[11]		[11,101,105,111]			[93]
C36								[81]
C37			[90]		[61,95]		[88,90]	

Legend: DevOps Capabilities C01-C37 mentioned in Table 3.
Life Cycle Processes LCP24 - Integration; LCP25 - Verification; LCP26 - Transition; LCP27 - Validation; LCP28 - Operation; LCP29 - Maintenance; LCP30 - Disposal.

developer feedback. This is consistent with incremental development processes making effective use of different tools. Automated processes that are implemented early ensure a well-defined, trackable and completely automated deployment into production [17,56,99,121,125]. Common processes include version control, CM, containerization and infrastructure as code, together with testing and design guidelines [24,40,47,54,59,107,110,114,120].

LCP24. *Integration:* Synthesizes system elements into a realized product or service, assembling them and activating interfaces to facilitate interoperation and meet system/software requirements [11].

Cultural capabilities: Integration in DevOps enables collaborative teamwork, process automation, and reduced software delivery cycles [100,114]. Continuous integration, CM, and security capabilities improve application security [14,92]. Open-source tools like Jenkins CI facilitate automation and accessibility [11,40,41,49,68,116]

Measurement capabilities: Automated testing, monitoring, and quality control are vital within the integration process. DevOps tools like Jenkins and OpenTelemetry aid automation and observability [17,23,57,61,126]. Integration methods enable rapid modifications and knowledgeable choices via bidirectional traceability and automated systems [14,94].

Process capabilities: DevOps integrates Development and Operations functions for continuous improvement [11,69]. Integration processes contain continuous integration, deployment, evaluation, and automated testing to standardize the data system [14,42,96]. Security is integrated to make sure information protection and privacy [23,47,83].

Technical capabilities: CI automates building and testing of software components [71,93,114]. CD includes CM, deployment, and verification practices [48,49,68,69,96,127]. Test Automation frameworks and tools speed up the testing process for faster integration and delivery [17,23,47,70,91,121,126].

LCP25. *Verification:* Confirms that the system fulfills specified requirements [11].

Cultural capabilities: Effective collaboration and communication are crucial for successful verification in DevOps [11]. Knowledge sharing, automation, and proactive monitoring play essential roles in verification [47].

Measurement capabilities: QA involves monitoring project outcomes using automated tools to measure goal achievement. Testing is performed throughout the system delivery life cycle, while QA ensures processes are followed competently [11].

Process capabilities: DevOps verification activities comprise peer reviews, bugs diagnosis and fixing, logging of tests and automated testing [11]. Thus, process improvement and proper application of changes is driven by collaboration and communication among stakeholders [14,89].

Technical capabilities: Verification requires establishing performance baselines, maintaining a repository for artifacts, and developing test plans and automation from the design process [11,90]. Test scripts provide a repeatable procedure for verifying requirements [125]. Transferable test scripts and a library of verification procedures expedite automatic execution. CM facilitates the verification process [14,48].

LCP26. *Transition:* Moves the system into operational status, ensuring functionality and compatibility.

Cultural capabilities: Blameless postmortems and minimizing fear of failing are essential for adequate transition [47]. At this stage, it is necessary to prioritize and document process improvements based on system thinking, feedback loops and continuous improvement [11].

Measurement capabilities: Proactive monitoring, observability, as well as autoscaling are essential for a successful transition [71]. Monitoring gives metrics that help to improve runtime controllers and also help diagnose operational problems. It also helps in predicting container group performance [57].

Process capabilities: In DevOps, the transitioning process integrates activities for managing and designing changes in business processes to be moved, and validating the system that is transitioning [14]. Therefore, it is important to incorporate customer feedback throughout the entire process and establish corrective actions plus operational guidance, while improving testing procedures [47].

Technical capabilities: The issue of transition mainly involves running new software to different places possible [14]. It is driven by Configuration Management, Verification and QA processes [11]. Essential tools for project planning include human and technical resources as well as security tools [47]. The incorporation of automation, containerization

and Infrastructure as Code eases this transition while ensuring security throughout the application life cycle [79].

LCP27. *Validation:* Provides evidence that the system achieves its intended use in operational environments [11].

Cultural Capabilities are all about ensuring that data, communication, processes and systems are as they should be. Testing, verification and review discover errors or mistakes [40,46,77,81,87]. In particular, in the more sensitive industry sectors, it is critical because of quality standards, assurance of reliability and compliance requirements adherence [11,14,23,129].

Measurement capabilities: Assessment aims at confirming and checking systems, goods or operations so that the desired criteria and specifications are fulfilled [114,119,122,123,128]. It encompasses techniques such as testing, examination and accreditation for compliance attainment and risk mitigation [89,94,119,127].

Process capabilities: Validation verifies accuracy and reliability through testing, simulation, or comparison with standards [41,66,79]. It identifies errors, increases confidence, and supports decision-making [14,58,62,98].

Technical capabilities: Validation ensures systems, processes, or products meet standards and requirements. It verifies functionality, performance, security, and user experience [11,40,61,81,82,90]. Effective strategies prevent errors and ensure reliable outcomes [23,57,64,70,89,104,109,129].

LCP28. *Operation:* Utilizes the system to deliver services while monitoring performance [11].

Cultural capabilities: Embedding quality management in operations requires continuous monitoring, communication, and collaboration [11,52,66,88,99,115]. DevOps enables CI/CD and automation using FLOSS [40,41,87]. Maturity assessment and procedure implementation restore normal operations [14,75,89].

Measurement capabilities: Operations involve coordinated efforts, ranging from systems to business operations [47,75,106]. Planning, communication, and execution are crucial for success [14,40,108,118,119,128].

Process capabilities: DevOps prioritizes continuous improvement through lightweight change approval, customer feedback, and automation [40,86,120]. Collaboration between operations and development is important for QA and problem resolution [11,43].

Technical capabilities: DevOps emphasizes fully automated processes such as CI, delivery, deployment automation, and CM. Security controls are maintained throughout the application lifecycle [14,40,63,108,129]. AIOps, streamlined pipelines, real-world testing, and infrastructure as code enhance operational efficiency [11,14,41,50,56,57,62,84,99].

LCP29. *Maintenance:* Maintains the system's service delivery capability [11].

Cultural capabilities: Collaboration between operations and development teams support a culture of experimentation and learning. FLOSS adoption and performance-oriented culture improve maintenance [24,43,50,52,63,81,84,89,115,126].

Measurement capabilities: Monitoring and observability are crucial for maintenance. Automation and continuous improvement enhance efficiency [46,57,61]. A continuous maintenance strategy should consider business value and change control procedures [11,45,119].

Process capabilities: DevOps improves maintenance through software reuse, user feedback, and working in small batches [38,82]. Supporting developers with fast change-requests from user feedback reduces rework [11,14,38,83,90].

Technical capabilities: Maintenance ensures optimal performance and longevity. Regular audits, repairs, and cleaning prevent problems. Proper planning, documentation, and coordinated ownership are essential [14,63,97,102,109]. Technical capabilities include artifacts management, database change management, Infrastructure as Code, containerization, and trunk-based development [5,17,38,40,57,60,72,84,90,93,104,110,128].

LCP30. *Disposal:* Manages the end of a system's intended use, including disposal of elements [11].

Cultural capabilities: Effective disposal in DevOps requires cross-team communication, collaboration, and supportive learning culture [11,89,101]. FLOSS adoption and establishing trust among stakeholders are important [101].

Measurement capabilities: DevOps uses features like container rotation, rolling updates, and autoscaling to make applications resilient. Proactive monitoring and system observation support the disposal process [54].

Process capabilities: Traceability, revising operating procedures, and CM improve the disposal process. Lightweight change approval and continuous improvement ensure consistency between products and their configurations [11].

Technical capabilities: Disposal in DevOps involves CI/CD, Test Automation, VCSs, and Cloud Computing. Loosely coupled architecture, Infrastructure as Code, Containerization, and Trunk-based development requires systematic and secure disposal of obsolete resources [5,54,78,104]. Test Data Management and Chaos Engineering also demand proper disposal for security and reliability [41,72,81,93].

4.2. RQ2 - Which categories of DevOps capabilities are most relevant to the software life cycle processes?

To facilitate answering the second research question,

Table 7 shows the total number of relations between categories of DevOps Capabilities and the Life Cycle Processes. It is considered a relation where at least one of the publications in this literature review is found to be relating a LCP with a DevOps Capability. Cultural capabilities have 100 relations in total, with an average of 14.29 relations per Life Cycle Process. Measurement capabilities contain a total of 79 relations, with an average of 15.80 relations per Life Cycle Process. Process capabilities have 101 total relations, with an average of 14.43 relations per Life Cycle Process. Technical capabilities have 309 total relations, with an average of 16.30 relations per Life Cycle Process.

The *Total # of Relations* column in Tables 7, 8, 10 and 11 represents the cumulative number of interactions or connections each category has with LCPs or capabilities from a quantitative standpoint. On the other hand, the *Average # of Relations* column reflects the mean value of these relations per each capability or process that composes that category, as expressed in Eq. (1).

$$Average\ \#\ of\ Relations = \frac{Sum\ of\ all\ relations\ in\ category}{Number\ of\ processes\ or\ capabilities} \tag{1}$$

For example, there are five "Measurement Capabilities", therefore the average will be:

$$\frac{30+8+20+6+15}{5} = \frac{79}{5} = 15.80$$

Per the observed data in Table 7, it is seen that the DevOps capability category with more relations in the technical one, followed by process, measurement, and cultural capabilities. The reason could be attributed to the fact that **Technical capabilities** are more directly related to software development and delivery [16,83]. For instance, CI/CD comprehends Technical capabilities that are essential for the software development process [21,72]. While **Cultural capabilities** are very important to foster a culture of collaboration and communication within the organization [89]. For example, cross-team collaboration is a cultural capability that can help to improve the flow of information between different teams within an organization [103].

On the other hand, the **Process capabilities** are seen as fundamental for software development, while less directly related to coding itself [4]. Continuous improvement is a process capability essential to improve the process of developing software over time [23]. Finally, **Measurement capabilities** are relevant to understanding the

Table 7
Relation sums and averages for each DevOps Capability category.

DevOps capability category	Total # of Relations	Average # of Relations
Technical Capabilities	309	16.30
Cultural Capabilities	100	14.29
Process Capabilities	101	14.43
Measurement Capabilities	79	15.80

Table 8
Relation sums and averages for each Life Cycle Process category.

Life Cycle Process category	Total # of Relations	Average # of Relations
Technical processes	290	20.71
Technical Management processes	150	18.75
Organizational Project-Enabling Processes	102	17.00
Agreement Processes	47	23.50

software development processes performance factors [22,90] as well as measuring the full life cycle. For example, proactive monitoring is a measurement capability that can be used to identify and resolve problems early on.

Overall, the analysis indicates that all four DevOps categories of capabilities are important for software development. However, technical capabilities are likely to have the most direct impact on the quality, speed, and reliability of software delivery.

On the other hand, Table 8 shows the total number of relations and the average number of relations for each Life Cycle Process category. The categories analyzed are Technical processes, Technical Management processes, Organizational Project-Enabling Processes, and Agreement Processes.

Prevalence of DevOps **Technical Processes** having the most practices and principles (290) because it is closest to DevOps Technical capabilities. The 20.71 relations per technical process indicate a consistent DevOps adherence pattern. A Significant Role of **Technical Management processes**, with 150 relations and an average of 18.75, show that DevOps is important in project technical management. This implies DevOps aids in technical resources, risk, and quality management. The influence on **Organizational Project-Enabling Processes** with DevOps integration having 102 relations and an average of 17.00 for Organizational Project-Enabling. By managing resources, making decisions, and engaging stakeholders, DevOps helps projects succeed. **Agreement Processes** have the fewest relations (47), but their high average number of relations per process (23.50) shows a good DevOps Capabilities on supply and acquisition processes. DevOps is likely to play a critical role in ensuring clear internal and external agreements and good collaboration. This might reflect the critical role of DevOps in ensuring clear, effective agreements and collaborations both within the organization and with external parties.

Overall, this table also suggests that there is a significant relationship between DevOps Capabilities and the Life Cycle Process categories [11], particularly Technical processes and Technical Management processes [22], but less organizational project-enabling and agreement processes. DevOps accelerates software development, delivery, collaboration, and flexibility throughout the life cycle.

5. Discussion

This research is built upon the knowledge gathered from our previous studies, with the same aim of enhancing successful DevOps adoption in organizations. Several authors still point out in recent studies that there are challenges and unknowns when it comes to an efficient transformation and implementation of DevOps [86,91,114,130]. As seen in Table 9, this study differs from the previous capabilities study which did not include the new IEEE DevOps standard [11], thus enabling us to relate DevOps Capabilities related to LCPs. This brings refreshed discussion to map and find the impact of the previously found DevOps Capabilities [4]. In the current SLR, it is seen the actual opportunity to cross investigate capabilities and DevOps processes, which, to the best of our knowledge, no one has pursued yet, thus contributing new knowledge beyond what was previously published. Furthermore, this study has uncovered important relationships that were not previously discussed but are standard across some software development models, including DevOps.

In this section, a discussion is performed, considering two interesting findings as a starting point. First, in Section 4.2, categories with the lowest total value always seem to have a large average value. Second, the data gathered from a number of publications contributes to different concepts under DevOps Capabilities and Life Cycle Processes.

5.1. Categories with fewer relations but high average values

Interesting finding to note in Tables 7 and 8, the last two categories stand out with the lowest total of relations, but still a large average value. This implies that even though the concept has a low frequency of occurrence, it tends to have high importance for the overall score. The observation happens in the two categories with the smallest number of capabilities and processes: agreement processes and measurement capabilities.

To try to explain why they have higher average values in LCPs despite their lower number of relations, it is first necessary to dive into what these values mean from a statistical perspective before highlighting their intrinsic qualities and strategic importance in achieving broad impact across LCPs. Here, a significant individual influence across the LCPs is denoted when sorting by average and examining each capability or process within that category. Although the agreement processes and measurement capabilities have fewer relations, their influence with LCPs seems particularly effective and meaningful based on the results. This suggests that while they associate with fewer processes (lower total relations), their interactions are high-quality (higher average), indicating a greater impact, raising new suspicions: Could this be a lead indicator of how to improve LCPs with DevOps Capabilities? Could there be high impact or even exceptional relationships? A topic for more debate over the course of this discussion in Section 5.2.

Furthermore, agreement processes, with the highest average of 23.50, hold supply and acquisition. This high number, when compared to Table 11 denotes the strategic importance of the supply process (LCP02) to align stakeholder expectations, project scopes, and quality standards [114]. The reason is that in DevOps, cross-functional collaboration is a bit more focused on supplying the customer [4,49] with 29 relations. Which is only 1.6 times more than acquisition process (LCP01), with 18 relations. Thus, this average is high when comparing both, but somewhat balanced.

However, when cross-checking the measurement capabilities, which have a lower average of 15.80, with Table 11, it is seen that the case is much different here: Proactive Monitoring, Observability, and

Table 9
Comparison of Objectives, Methodology, and Findings between our current and previous studies.

Study	Objectives	Methodology	Findings
The current Study	Identify how DevOps Capabilities map to the software life cycle per the new IEEE DevOps Standard [11].	Systematic Literature Review (SLR) of peer-reviewed articles.	Identified key DevOps Capabilities across the software life cycle, highlighting areas needing further research.
Capabilities and Metrics in DevOps: A Design Science Study [3]	Define and classify key DevOps metrics and capabilities for promoting effective adoption.	Design Science Research (DSR) with qualitative methods, including interviews.	Developed an outcome-based capability evaluation matrix, emphasizing team empowerment and organizational culture.
DevOps Metrics and KPIs: A Multivocal Literature Review [131]	Provide relevant DevOps metrics for assessing and enhancing DevOps implementation efficiency.	Multivocal Literature Review (MLR) of a wide range of sources.	Defined and categorized 22 main DevOps metrics, offering insights into their improvement and practical application.
DevOps benefits: A systematic literature review [130]	Consolidate the benefits of DevOps as reported in literature and empirically validate them through case studies.	Two systematic literature reviews to gather benefits and then map them to case studies.	Identified and validated benefits such as improved collaboration and faster delivery, and increased automation.
Capabilities and Practices in DevOps: A Multivocal Literature Review [4]	Explore the relationship between DevOps Capabilities and practices to aid in better implementation.	Multivocal Literature Review (MLR) including books, articles, white papers, and conferences.	Presented an organized list of 37 capabilities and their relation to practices, emphasizing the dynamic nature of DevOps Capabilities.

Table 10
Categories with fewer relations but high average.

Category	Type	Total	Relations	Average
Agreement	Processes	2	47	23.50
Measurement	Capabilities	5	79	15.80
Organizational Project-Enabling Processes	Processes	6	102	17.00
Cultural	Capabilities	7	100	14.29
Process	Capabilities	7	101	14.43
Technical Management processes	Processes	8	150	18.75
Technical processes	Processes	14	290	20.71
Technical	Capabilities	18	309	16.30

Table 11
Agreement process and Measurement capabilities relation overview.

ID	Name	Category	Relations
LCP02	Supply process	Agreement	29
LCP01	Acquisition process	Agreement	18
C08	Proactive Monitoring, Observability and autoscaling	Measurement	30
C10	Monitor systems to inform business decisions	Measurement	20
C12	Visual management Capabilities	Measurement	15
C09	Emergency response/proactive failure notification	Measurement	8
C11	Working in progress limits	Measurement	6

autoscaling capabilities (C08) have 30 mentioned relations, which is five times more than the lowest working in progress limits (C11) which have only 6 relations. Further, monitoring systems to inform business decisions (C08), comes next with 20 relations, which is still 3.3 times of C11. The notable variation in the spread of relationships among Measurement Capabilities highlights the strategic focus on those that directly impact software delivery quality and reliability, thereby also keeping customer satisfaction in mind.

LCP02, focuses on supply agreements with clients [11]. These agreements define the scope of work, deliverables, schedule and quality standards of the software development or system implementation project [14]. In the context of DevOps, both supply and acquisition agreements are crucial to ensuring that the teams involved in the software delivery process are aligned and have clear expectations about their responsibilities, the scope of the project and the quality requirements. Teams that adopt DevOps collaborate with different stakeholders, such as business owners, production teams and external suppliers, to ensure that everyone is working towards the same goals and objectives [6,89,102].

Measuring software delivery performance, identifying bottlenecks, and improving performance are key points in C08 and C10 for organizations [22,68]. DevOps measurement enables businesses to track key performance indicators (KPIs) such as deployment frequency, MLT, MTTR, and CFR [131]. DevOps Measurement Capabilities help organizations collect and analyze software development and delivery data to identify patterns, trends, and improvement opportunities. A data-driven approach helps organizations make informed decisions, reduce risks, and optimize software delivery processes [22,23,120].

In summary, the reason behind the high average values of both Agreement Processes and Measurement Capabilities, despite their fewer relations, can indeed be considered their shared ultimate goal of servicing the customer effectively. Through their functions, both categories reflect the customer-centric nature of DevOps. They express the DevOps

aim of providing efficient customer service by ensuring project completion and meeting customer expectations (Agreement Processes) and continuously measuring and monitoring for improving product quality and delivery (Measurement capabilities). DevOps promotes strategies and processes that improve the software delivery lifecycle to meet client expectations. This particularly interesting finding leads to the next discussion, which explores other capabilities that improve LCPs.

5.2. Improving life cycle processes with DevOps capabilities

For asserting the most influential DevOps Capabilities on LCPs Activities and Tasks, it is proposed to discuss the results, regarding the top 3% and 1% most referenced relations, while diving into what authors are mentioning in Sections 4.1 and 4.2 and demonstrating the proposed reasoning. To facilitate a structured discussion, a percentile-based impact scale, seen in, Eq. (2) is used to quantify and compare the relations found. Previous authors [132,133] also apply a percentile-based approach to help discussion in qualitative research, while Verner et al. (2014) [134] use percentiles to understand the distribution of responses regarding project outcomes and other factors. This statistical measure represents the value below which a particular percentage of observations in a dataset falls. Percentiles help in comparing datasets by indicating the percentage of observations that a given value surpasses, thus aiding in drawing precise conclusions from the research data [19, 134]. Let P be the percentile rank of mentions and N the number of mentions. The classification function, $C(P)$, can be defined as:

$$C(P) = \begin{cases} \text{Exceptional (Top 1\%)} & \text{if } P \geq 99 \quad (N \geq 14) \\ \text{Very High (Top 3\% but below 1\%)} & \text{if } 97 \leq P < 99 \quad (N \geq 11) \\ \text{High} & \text{if } 90 \leq P < 97 \quad (N \geq 6) \\ \text{Increased} & \text{if } 75 \leq P < 90 \quad (N \geq 3) \\ \text{Medium} & \text{if } 50 \leq P < 75 \quad (N \geq 2) \\ \text{Low} & \text{if } P < 50 \quad (N < 2) \end{cases} \tag{2}$$

From Table 12 it can be seen that eight capabilities are classified as "Exceptional" (Top 1%), and 11 capabilities are classified as "Very High" (Top 3% to 1%), which is a strong indication from literature of their relation influencing activities and tasks of the respective LCPs.

5.2.1. Exceptional impact relations

We start by looking at the **exceptional** relations. There are four LCPs in this classification.

LCP02 Supply Process is improved by Shifting Left on Security (C32) and Monitoring, Observability, and Autoscaling (C08), by assuring continuous, secure, and efficient product/service delivery. Alonso et al. (2022) [60] stress the need for strong monitoring in cloud services to help identify and resolve performance issues quickly. Gokarna et al. (2021) [69] integrate security early in development using open-source software over the cloud, preventing supply chain disruptions and maintaining service delivery, while network monitoring and diagnostic tools [61] improve operational efficiency and dependability through constant observability and security. On the other hand, vulnerability scans and monitoring catch flaws and destructive activities early [40]. As an example, Netflix supplies resilient cloud services for its front-end, monitoring, and payment operations [53].

The addition of C08 and C32 has a direct impact on the **LCP19 System/Software Requirement Definition**. The AIDOaRt project [119] utilizes AI and ML to enhance DevOps toolchains and convert stakeholder views into measurable system needs using runtime data. Levy et al. (2022) mention that using this information ensures that needs are based on operational realities rather than theoretical assumptions [86]. According to Gokarna et al. (2021), their continuous security model incorporates security practices early in development, changing criteria to include security from the outset [69]. Monitoring can identify issues and spur modifications in requirements, making them dynamic and sensitive to real-world conditions [11,60].

Continuous Integration (C20) accelerates the **LCP24 Integration Process** by with rapid feedback, failure detection [11] and automated merges and tests in an early phase. Some authors examine the impact of CI in DevOps and project management, pointing out software components' improved consistency and quality with less effort [48,93,127]. Fitzgerald et al. (2017) [68] highlight its heterogeneity in definition but key importance in XP methodologies. Other authors discuss CI in terms of automating build and testing tasks and improving software process health through frequent application component merging [40, 126], in pair with Xuan et al. (2021) emphasizing CI responsibility for guaranteeing security throughout the integration process.

C20, C21, and C32 collectively facilitate the **LCP27 Validation process**. According to Ayerdi et al. (2020) [123], CD plays an important role in automating the deployment of new software releases, leading to faster validation. CI involves compiling code, running tests, and validating code coverage, which is critical for early anomaly identification [11,68] . As an example, Kumar et al. (2020) [40] discuss technologies like Maven, Gradle, and Jenkins to automate code in CI, address bugs quickly, and minimize time to adopt new features, while the importance of incorporating security tests early in the development cycle, guarantees a fast security verification process [11,114].

5.2.2. Very high impact relations

Regarding **very high impact relations**, there are eight LCPs in this classification, which are not in the top 1% but fall into the top 3%.

Authors mention C28 to improve the modularity, scalability, and resilience of **LCP02 Supply process** which expedites development and deployment [5,51], improve the supply chain by containerizing and deploying across environments [45] and isolate service failures [51].

C30 simplifies **LCP04 Infrastructure Management Process** consistency, performance, and security [11] with tools like Puppet, Chef, and Ansible [23,82] enabling organizations to quickly react to changing markets and consumer expectations [63,66].

C21 enables **LCP19 System/Software Requirements Definition** to fulfill stakeholder needs and customer input by adding new features and distributing new software quickly [49,123], revealing expectations conflicts and enabling rapid design and software modifications [23,97].

C32 adds early security in the **LCP23 Implementation process**, providing safety and quality [99,114] including traceability and audit controls [11] mitigating risks, ensuring high security standards [47,54].

C08, C21, and C32 provide a safer, faster, and improved **LCP24 Integration**. C08 detects production, deployment, and integration problems during CI/CD to optimize software reliability [11,71]. C21 enhances software deployment, feature integration, and customer satisfaction [48,118] via on-demand deployments [23,70,72]. C32 enforces continuous security in each phase, resulting in safer integrations [70, 72].

C02 and C08 enhance **LCP27 Validation** by encouraging continuous learning and adaptive monitoring to assess operational scenarios. C02 focuses on controlled trials and empirical validation to promote DevOps innovation [38,58]. With C08, data tracking and observation allow for real-time validation [11,123].

C08 improves the **LCP28 Operation process** in real time, allowing smart infrastructure changes [60,123]. Logging, monitoring, and alerting help operational management notice, respond, and adapt issues and speed incident response [40,57].

C21 streamlines the **LCP29 Maintenance process** with consistent, efficient, and reliable deployments across all environments [11], increased team cooperation [90,108], and rapid response to changes and customer needs improve the maintenance process with Continuous Delivery automation [63].

5.2.3. Applying the life cycle concepts

Table 13 shows publications contributing to the concepts seen in the conceptual map in Fig. 1.

Table 12
Capabilities with exceptional and very high impact on Life Cycle Processes.

			LCP02	LCP04	LCP19	LCP23	LCP24	LCP27	LCP28	LCP29
		LCP:	Supply Processes	Infrastructure Management process	System/Software Requirements Definition	Implementation	Integration	Validation	Operation	Maintenance
	Devops Capability	Category	Agreement	Project	Technical					
C02	Learning and experiment	Cultural						11		
C08	Monitoring, Observability and autoscaling	Measurement	16		14		11	13	13	
C20	Continuous Integration	Technical					14	16		
C21	Continuous Delivery	Technical			13		12	16		11
C28	Loosely coupled architecture	Technical	13							
C30	Infrastructure as Code	Technical		11						
C32	Shift left on security	Technical	15		14	11	11	16		

Legend: indicates **Exceptional** impact DevOps Capabilities impact related to Life Cycle Processes (top 1%), and indicates **Very high** impact DevOps Capabilities impact related to Life Cycle Processes (top 3%).

Table 13
Publications contributing to this study concepts from Fig. 1.

Concepts	Publications	Total
Capabilities and Process	[1,5,11,14,16,17,21–24,38–129]	102
Teams	[1,5,11,16,17,21–24,38,40,43–45,47–52,57,58,60,63–69,72,74–80,82,85–87,89,90,92,95,96,98–100,102–104,106,108,109,112–115,119–122,124,127,129]	69
Activities and Tasks	[11,14,23,24,40,44–48,54,60,62,63,68,69,76–78,82,90,95,99,102,106,109,110,121,122,127]	31
Outcomes	[1,11,14,21,22,24,38,45,49,56,61,63,67,73,76,78,81,88,97,99,101,103,104,106,109,110,114,117,120,122]	30
Assessment	[14,22,24,38,44,46,49,52,63,67,69,82,83,88,90,94,97,99,101,103,109,111,114,122,124]	25
Skills and Knowledge	[5,11,14,17,24,46,47,58,63,75–77,80,87,89,95,102,106,123]	19
Purpose	[11,14,23,24,46,53,59,66,88,91,102,110,112,113]	14

The *Capabilities and Process* concepts, referenced in 102 articles, focuses on DevOps Capabilities (Fig. 2) connected to software Life Cycle Processes (Fig. 4). This shows the importance given by authors of integrating them together when adopting DevOps. 69 publications have discussed the *Teams* concept, which incorporates collaboration and communication among DevOps teams, such as development, operations, and QA [17,77,89]. The number of publications on the cultural capacity concept indicates how important teamwork and communication are for DevOps implementations. Teams do process activities, develop skills, and drive execution [102,106]. 31 articles identify *Activities and Tasks* as a vital concept for DevOps activities such as testing, developing, deploying, and monitoring [11] and a growing number of publications highlight the importance of understanding the activities and tasks involved in successful DevOps implementations [90]. *Outcomes* is a concept found in 30 publications examining the benefits of DevOps adoption in the software/systems life cycle [11,89]. Publications highlight DevOps positive effects on efficiency, quality, and customer satisfaction [23,52]. Amaro et al. [4] define DevOps Capabilities as providing process outcomes, facilitating task execution, and requiring specific skills and knowledge. 25 publications on *DevOps Assessment* explore various methods for integrating DevOps into Life Cycle Processes, while providing clarity on how to quantify DevOps effectiveness. DevOps assessment is critical for connecting expected objectives with actual achievements [22,97]. Furthermore, it examines DevOps skills to improve process outcomes [14,24]. The successful implementation of DevOps principles in the software/systems life cycle is dependent on *Skills and Knowledge*, as highlighted in 19 articles. Research underlines the necessity of continuous learning and experimentation in DevOps, highlighting the need for individuals and teams to increase skills and knowledge to traverse complexities [5,63,77]. Finally, *Purpose* emphasizes each LCP intention to employ DevOps to achieve their objectives. 14 papers discuss the rationale and goals of DevOps adoption. Integrating DevOps into Life Cycle Processes necessitates understanding its goal of connecting actions and strategies to intended outcomes [11,88,102]. The reviewed articles explain how DevOps Capabilities improve software/systems life cycle collaboration, efficiency, and outcomes.

In order to show this relevance, Fig. 9 highlights and graphically demonstrates these approaches to improve DevOps adoption through capabilities, integrating the Conceptual Map overview (Fig. 1) and Exceptional Impact Relations (Table 12), and focusing on LCP outcomes derived from capabilities.

The conceptual map in Section 1 highlights capabilities in DevOps connected to desired improvements, while Table 13 quantifies key concepts for successful implementations. Fig. 9 shows how the exceptional impact of relations in conceptual map flows might achieve desired outcomes. We do this for exceptional ones in order to be concise and to focus on the top 1% which will return best results, but the same could be done for very high impact relations or even high. Apply DevOps Capabilities to Life Cycle Processes they affect, describe output generation pathways, and highlight benefits. This diagram illustrates how implementing the best DevOps skills can improve the SDLC and achieve desired results. It shows how DevOps adoption directly improves software development and delivery processes, helping people comprehend its potential benefits.

5.3. Impact and practical applications on the field of DevOps

This work advances the understanding of DevOps Capabilities and how they work with LCPs, which makes it possible to design methods for DevOps adoption that are more successful. This increase in knowledge helps businesses identify capabilities that are vital and require attention, especially when they significantly affect their processes. The methodology targets the most pressing areas for organizations by identifying capabilities with Exceptional, Very High, and High impacts. However, the literature review indicates that the expected benefits of focusing on smaller impact capabilities are still unclear, thus further research like a case study is needed to validate the findings.

The integration of Monitoring, Observability, and Autoscaling (C08), alongside Shifting Left on Security (C32), Continuous Integration (C20) and Continuous Delivery (C21), significantly enhances various LCPs, including Supply Process (LCP02), System/Software Requirements Definition (LCP19), Integration Process (LCP24), and Validation Process (LCP27). According to these findings, integrating C08, C32, C20 and C21 improves operational security and efficiency while ensuring that

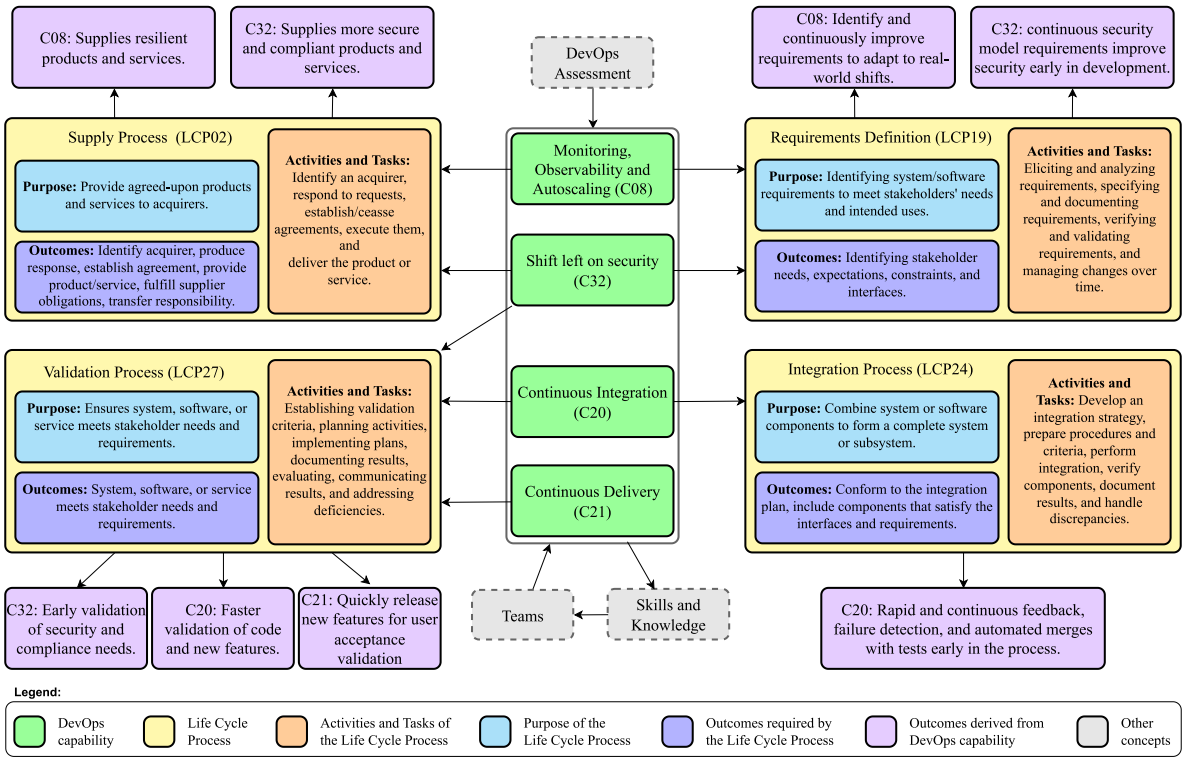


Fig. 9. Improving LCP outcomes with exceptional DevOps Capabilities.

products and services are appropriately engineered to perform in operational settings.

6. Conclusion

6.1. Contributions

This study is part of a broader set that aims to improve the successful adoption of DevOps by examining the problems and uncertainties in adopting DevOps. It distinguishes itself from our previous work by conducting an extensive systematic literature review, focusing on the cross-research of DevOps Capabilities and processes. The study also explored the connections between Life Cycle Processes (LCPs) and the capabilities that drive process improvements. The inclusion of the IEEE DevOps standard provides a refreshed discussion on the impact of the most important DevOps Capabilities.

The 102 publications analyzed discuss capabilities and processes, teams, activities and tasks, results, tools and techniques, according to the analysis. The main contributions are made, emphasizing software development capabilities and processes:

- This paper maps 37 DevOps Capabilities to 30 Life Cycle Process (LCP), demonstrating their application throughout various stages of software development and maintenance. By mapping the pre-existing categorization of capabilities and LCPs, the paper provides valuable insights on improving efficiency and achieving better results through DevOps in each Life Cycle Process.
- Technical DevOps Capabilities and Technical processes show the most relations and impact, highlighting a significant connection between DevOps Capabilities and LCPs in these Technical groups.
- The paper identifies and discusses DevOps Measurement Capabilities and Agreement Processes as having fewer direct relations but influencing multiple SDLC aspects or capabilities. This insight emphasizes the importance of these categories in achieving DevOps success.
- For improving LCP with DevOps Capabilities, an impact scale classification is found that identifies and explains exceptional and very high impact relations. Based on the publications and the conceptual map, it is shown how exceptional DevOps Capabilities can improve LCP outcomes in a diagram of concepts (Fig. 9).
- The paper explores applying the life cycle concept map, demonstrating the flow of improving LCP outcomes with exceptional DevOps Capabilities, while also pointing out agreement, organizational project-enabling, and technical LCPs. The analysis reveals that both Life Cycle Processes and DevOps Capabilities achieve desired outcomes through the activities and tasks mentioned in the literature.

This paper improves understanding of DevOps and LCPs. It discusses life cycle concepts, DevOps implementations, and a framework for integrating DevOps Capabilities into software development and maintenance.

6.2. Limitations

This SLR acknowledges potential threats to its validity. External validity concerns related to the limited scope of the selected scientific databases may have led to the exclusion of pertinent articles. However, efforts were done to address this by reviewing the references and to make sure that important research was added through snowballing. Internal validity, meaning how well the study design and execution prevents systematic error, was safeguarded through a predefined procedure and established quality evaluation guidelines by Kitchenham & Charters (2007) [34]. A possible selection bias was minimized by a broad inclusion and exclusion criteria and more than two reviewers participated, ensuring a rigorous study selection process. Limiting the search to studies in English may exclude relevant studies in other languages. Also, setting time limits for inclusion may exclude relevant research published before or after the time-frame. Finally, address construct validity and content validity by reviewing a large number of documents. Construct validity is ensuring accurate capture and

synthesizing of key concepts. Content validity is improving coverage of relevant research. A comprehensive approach provides nuanced understanding and a comprehensive overview of the domain, making these findings robust and reliable. The study is using secondary sources to understand relationships, thus conclusions could be strengthened by exploratory case studies in the industry. Finally, monitoring and security technologies change quickly, so the findings may need a periodic refresh.

6.3. Future work

The results and findings revealed in this study will assist to feed new research so that future studies can be put into practice. Focusing on an assessment model composed of the interconnected dimensions of DevOps Capabilities, DevOps metrics [3], and LCPs.

Future work can also provide applicable guidelines and practices for DevOps integrating into LCPs. Such as empirical and exploratory research on how DevOps Capabilities affect LCPs would be useful. These guidelines can improve software development and maintenance by providing step-by-step instructions, frameworks, and recommendations. These studies could assess how specific capabilities influence software quality, development speed, reliability, and cost-effectiveness. Cultural, measurement, process, and technical capabilities can be explored further. Investigating specific practices, techniques, and tools within each category can improve LCPs and DevOps implementations. DevOps adoption and implementation depend on organizational and cultural factors. Researching the obstacles organizations face in adopting DevOps Capabilities and ways to promote collaboration, continuous learning, and innovation. Long-term DevOps evaluations can reveal their sustainability and scalability by tracking DevOps Capabilities and their extended effects on software development. Case studies of DevOps Capabilities adoption in various industries are needed. Lastly, future research could integrate DevOps with AI, machine learning, and blockchain. Researching how these technologies improve LCPs and DevOps implementations would be interesting.

CRediT authorship contribution statement

Ricardo Amaro: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis,Data curation, Conceptualization. **Rúben Pereira:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Miguel Mira da Silva:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing 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 paper.

Data availability

Data will be made available on request.

References

[1] P. Tell, J. Klunder, S. Kupper, D. Raffo, S.G. Macdonell, J. Munch, D. Pfahl, O. Linssen, M. Kuhrmann, What are hybrid development methods made of? An evidence-based characterization, in: Proceedings - 2019 IEEE/ACM International Conference on Software and System Processes, ICSSP 2019, IEEE, 2019, pp. 105–114, http://dx.doi.org/10.1109/ICSSP.2019.00022.

[2] D. Solajić, A. Petrović, Devops and modern software delivery, in: Proceedings of the International Scientific Conference - Sinteza 2019, Singidunum University, Novi Sad, Serbia, 2019, pp. 360–368, http://dx.doi.org/10.15308/Sinteza-2019-360-368.

[3] R. Amaro, R. Pereira, M.M. da Silva, Capabilities and metrics in DevOps: a design science study, Inf. Manag. (2023) 32, http://dx.doi.org/10.1016/j.im.2023.103809.

[4] R. Amaro, R. Pereira, M. Mira da Silva, Capabilities and practices in DevOps: A multivocal literature review, IEEE Trans. Softw. Eng. 1 (2022) 20, http://dx.doi.org/10.1109/TSE.2022.3166626.

[5] L. Leite, C. Rocha, F. Kon, D. Milojicic, P. Meirelles, A survey of DevOps concepts and challenges, ACM Comput. Surv. 52 (6) (2019) 35, http://dx.doi.org/10.1145/3359981.

[6] N. Azad, S. Hyrynsalmi, DevOps critical success factors — A systematic literature review, Inf. Softw. Technol. 157 (2023) 107150, http://dx.doi.org/10.1016/j.infsof.2023.107150.

[7] K. Maroukian, S. R. Gulliver, Synthesis of a leadership model for DevOps adoption, in: 2021 2nd European Symposium on Software Engineering, in: ESSE 2021, Association for Computing Machinery, New York, NY, USA, 2021, pp. 58–66, http://dx.doi.org/10.1145/3501774.3501783.

[8] R.N. Rajapakse, M. Zahedi, M.A. Babar, H. Shen, Challenges and solutions when adopting DevSecOps: A systematic review, Inf. Softw. Technol. 141 (2022) 106700, http://dx.doi.org/10.1016/j.infsof.2021.106700.

[9] S. Sharma, The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise, IBM Press, John Wiley & Sons, Inc., Indianapolis, Indiana, 2017, http://dx.doi.org/10.1002/9781119310778.

[10] G. Kim, J. Humble, P. Debois, J. Willis, The DevOps Handbook : How to Create World-Class Agility, Reliability, and Security in Technology Organizations, IT Revolution Press, USA, 2016, https://www.amazon.com/DevOps-Handbook-World-Class-Reliability-Organizations/dp/1942788002.

[11] IEEE, IEEE Standard for DevOps: Building reliable and secure systems including application build, package, and deployment: IEEE Standard 2675-2021, IEEE Std 2675-2021, 1 (16 Apr 2021) (2021) 91, http://dx.doi.org/10.1109/IEEESTD.2021.9415476.

[12] IEEE, ISO/IEC/IEEE International Standard - Systems and software engineering – System life cycle processes, ISO/IEC/IEEE 15288 First edition 2015-05-15, 2015, p. 118, http://dx.doi.org/10.1109/IEEESTD.2015.7106435.

[13] IEEE Standards Association, IEEE Standard for configuration management in systems and software engineering: IEEE Std 828™-2012 (Revision of IEEE Std 828-2005), IEEE Std 828-2012 (Revision of IEEE Std 828-2005), 2012, (March) 2012, http://dx.doi.org/10.1109/IEEESTD.2012.6170935.

[14] IEEE Standard, ISO/IEC/IEEE international standard - systems and software engineering – Software life cycle processes, ISO/IEC/IEEE 12207:2017(E) First edition 2017-11, 2017, p. 157, http://dx.doi.org/10.1109/IEEESTD.2017.8100771.

[15] J. Díaz, D. López-Fernández, J. Pérez, Á. González-Prieto, Why are many businesses installing a DevOps culture into their organization? Empir. Softw. Eng. 26 (2) (2021) 50, http://dx.doi.org/10.1007/s10664-020-09919-3.

[16] C. Jones, A proposal for integrating DevOps into software engineering curricula, in: B. Meyer, M. Mazzara, J.-M. Bruel (Eds.), Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, DEVOPS 2018, vol. 11350 LNCS, Springer Verlag, 2019, pp. 33–47, http://dx.doi.org/10.1007/978-3-030-06019-0_3.

[17] M. Senapathi, J. Buchan, H. Osman, DevOps capabilities, practices, and challenges: insights from a case study, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 - EASE'18, in: EASE'18, (June) ACM, Association for Computing Machinery, New York, USA, 2018, pp. 57–67, http://dx.doi.org/10.1145/3210459.3210465.

[18] N. Forsgren, J. Humble, G. Kim, Accelerate: The Science of Lean Software and Devops: Building and Scaling High Performing Technology Organizations, IT Revolution, USA, 2018, URL https://itrevolution.com/accelerate-book/.

[19] L. Bass, I. Weber, L. Zhu, DevOps: A software architect's perspective, in: SEI Series in Software Engineering, Addison-Wesley, New York, 2015, URL http://my.safaribooksonline.com/9780134049847.

[20] P. Debois, Agile infrastructure and operations: How infra-gile are you? in: Proceedings - Agile 2008 Conference, 2008, pp. 202–207, http://dx.doi.org/10.1109/Agile.2008.42.

[21] M. Waseem, P. Liang, M. Shahin, A systematic mapping study on microservices architecture in DevOps, J. Syst. Softw. 170 (2020) http://dx.doi.org/10.1016/j.jss.2020.110798.

[22] A. Mishra, Z. Otaiwi, Devops and software quality: a systematic mapping, Comp. Sci. Rev. 38 (1) (2020) 14, http://dx.doi.org/10.1016/j.cosrev.2020.100308.

[23] P. Rodríguez, M. Mäntylä, M. Oivo, L.E. Lwakatare, P. Seppänen, P. Kuvaja, Advances in using agile and lean processes for software development, in: A. Memon (Ed.), Advances in Computers, vol. 113, Academic Press Inc., Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland, 2019, pp. 135–224, http://dx.doi.org/10.1016/bs.adcom.2018.03.014.

[24] R. Kneuper, Software Processes and Life Cycle Models: An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes, Springer International Publishing, Cham, 2018, http://dx.doi.org/10.1007/978-3-319-98845-0.

[25] H.D. Benington, Production of large computer programs, Ann. Hist. Comput. 5 (4) (1983) 350–361, http://dx.doi.org/10.1109/MAHC.1983.10102.

[26] W.W. Royce, Managing the development of large software systems: Concepts and techniques, in: Proceedings of the 9th International Conference on Software Engineering, in: ICSE '87, IEEE Computer Society Press, Washington, DC, USA, 1987, pp. 328–338.

[27] T.E. Bell, T.A. Thayer, Software requirements: Are they really a problem? in: Proceedings of the 2nd International Conference on Software Engineering, in: ICSE '76, IEEE Computer Society Press, Washington, DC, USA, 1976, pp. 61–68.

[28] B.W. Boehm, Software Engineering Economics, first ed., Prentice Hall, Englewood Cliffs, N.J, 1981.

[29] J. Münch, O. Armbrust, M. Kowalczyk, M. Soto, Software process definition and management, The Fraunhofer IESE Series on Software and Systems Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, http://dx.doi.org/10.1007/978-3-642-24291-5.

[30] B.W. Boehm, Guidelines for verifying and validating software requirements and design specifications, in: P.A. Samet (Ed.), Euro IFIP 79, North Holland, 1979, pp. 711–719.

[31] C. Larman, V. Basili, Iterative and incremental developments. a brief history, Computer 36 (6) (2003) 47–56, http://dx.doi.org/10.1109/MC.2003.1204375.

[32] C. Floyd, A systematic look at prototyping, in: R. Budde, K. Kuhlenkamp, L. Mathiassen, H. Züllighoven (Eds.), Approaches To Prototyping, Springer, Berlin, Heidelberg, 1984, p. 18, http://dx.doi.org/10.1007/978-3-642-69796-8_1.

[33] B.W. Boehm, A spiral model of software development and enhancement, Computer 21 (5) (1988) 61–72, http://dx.doi.org/10.1109/2.59.

[34] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. rep., Technical report, ver. 2.3 ebse technical report. ebse, 2007.

[35] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele University 33 (2004) 26.

[36] D. Moher, A. Liberati, J. Tetzlaff, D.G. Altman, T.P. Group, Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement, PLOS Med. 6 (7) (2009) e1000097, http://dx.doi.org/10.1371/journal.pmed.1000097.

[37] C. Wohlin, 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, in: EASE '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 10, http://dx.doi.org/10.1145/2601248.2601268.

[38] N. Ali, H. Daneth, J.-E. Hong, A hybrid DevOps process supporting software reuse: A pilot project, J. Softw.: Evol. Process 32 (7) (2020) e2248, http://dx.doi.org/10.1002/smr.2248.

[39] M. Sánchez-Gordón, R. Colomo-Palacios, Characterizing DevOps culture: A systematic literature review, in: I. Stamelos, T. Rout, R. O'Connor, A. Dorling (Eds.), 18th International Conference on Software Process Improvement and Capability Determination, SPICE 2018, vol. 918, Springer Verlag, Østfold University College, Halden, 1757, Norway, 2018, pp. 3–15, http://dx.doi.org/10.1007/978-3-030-00623-5_1.

[40] R. Kumar, R. Goyal, Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC), Comput. Secur. 97 (2020) 101967, http://dx.doi.org/10.1016/j.cose.2020.101967.

[41] I. Al-Surmi, B. Raddwan, I. Al-Baltah, Next generation mobile core resource orchestration: comprehensive survey, challenges and perspectives, Wirel. Pers. Commun. 120 (2) (2021) 1341–1415, http://dx.doi.org/10.1007/s11277-021-08517-w.

[42] D. Yang, D. Wang, D. Yang, Q. Dong, Y. Wang, H. Zhou, H. Daocheng, DevOps in practice for education management information system at ECNU, in: M. Cristani, C. Toro, C. Zanni-Merk, R.J. Howlett, L.C. Jain (Eds.), Procedia Computer Science, Vol. 176, 2020, pp. 1382–1391, http://dx.doi.org/10.1016/j.procs.2020.09.148.

[43] A.W. Miller, R.E. Giachetti, D.L. Van Bossuyt, Challenges of adopting devops for the combat systems development environment, Def. Acquis. Res. J.: Publ. Def. Acquisit. Univ. 29 (1) (2022) 22–49, http://dx.doi.org/10.22594/dau.21-870.29.01.

[44] N.M. Noorani, A.T. Zamani, M. Alenezi, M. Shameem, P. Singh, Factor prioritization for effectively implementing DevOps in software development organizations: A SWOT-AHP approach, Axioms (2075-1680) 11 (10) (2022) N.PAG–N.PAG.

[45] C.E. da Silva, Y.d. Justino, E. Adachi, SPReaD: Service-oriented process for reengineering and DevOps, Serv. Orient. Comput. Appl. 16 (1) (2022) 16, http://dx.doi.org/10.1007/s11761-021-00329-x.

[46] ISO/IEC/IEEE15288, 21840–2019 - ISO/IEC/IEEE International Standard - Systems and software engineering – Guidelines for the utilization of ISO/IEC/IEEE 15288 in the context of system of systems (S0S), ISO/IEC/IEEE 15288, 2019, http://dx.doi.org/10.1109/IEEESTD.2019.8929110.

[47] M. Muñoz, M.N. Rodríguez, A guidance to implement or reinforce a DevOps approach in organizations: A case study, J. Softw.: Evol. Process 1 (2021) 21, http://dx.doi.org/10.1002/smr.2342.

[48] C. Baron, V. Louis, Towards a continuous certification of safety-critical avionics software, Comput. Ind. 125 (2021) http://dx.doi.org/10.1016/j.compind.2020.103382.

[49] F. Helwani, J. Jahić, ACIA: A methodology for identification of architectural design patterns that support continuous integration based on continuous assessment, in: 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), 2022, pp. 198–205, http://dx.doi.org/10.1109/ICSA-C54293.2022.00046.

[50] D. Pianini, A. Neri, Breaking down monoliths with Microservices and DevOps: An industrial experience report, in: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2021, pp. 505–514, http://dx.doi.org/10.1109/ICSME52107.2021.00051.

[51] L.J. Pérez, J. Salvachúa, L.J. Perez, J. Salvachua, An approach to build E-health IoT Reactive Multi-Services based on technologies around cloud computing for elderly care in smart city homes, Appl. Sci.-Basel 11 (11) (2021) http://dx.doi.org/10.3390/app11115172.

[52] S. Rafi, M.A. Akbar, A.A. AlSanad, L. AlSuwaidan, H. Abdulaziz AL-ALShaikh, H.S. AlSagri, Decision-making taxonomy of DevOps success factors using preference ranking organization method of enrichment evaluation, Math. Probl. Eng. (2022) 15, http://dx.doi.org/10.1155/2022/2600160.

[53] N. Herbst, A. Bauer, S. Kounev, G. Oikonomou, E. Van Eyk, G. Kousiouris, A. Evangelinou, R. Krebs, T. Brecht, C.L. Abad, A. Iosup, Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges, ACM Trans. Model. Perform. Eval. Comput. Syst. 3 (4) (2018) 36, http://dx.doi.org/10.1145/3236332.

[54] A. Poniszewska-Marańda, E. Czechowska, Y.-S. Chen, Kubernetes cluster for automating software production environment, Sensors (14248220) 21 (5) (2021) 1910, http://dx.doi.org/10.3390/s21051910.

[55] S. Leech, J. Dunne, D. Malone, A framework to model bursty electronic data interchange messages for queueing systems†, Fut. Int. 14 (5) (2022) 149, http://dx.doi.org/10.3390/fi14050149.

[56] H. Zhou, Y. Hu, X. Ouyang, J. Su, S. Koulouzis, C. de Laat, Z. Zhao, CloudsStorm: A framework for seamlessly programming and controlling virtual infrastructure functions during the DevOps lifecycle of cloud applications, Softw. - Pract. Exp. 49 (10) (2019) 1421–1447, http://dx.doi.org/10.1002/spe.2741.

[57] M. Usman, S. Ferlin, A. Brunstrom, J. Taheri, A survey on observability of distributed edge & container-based microservices, IEEE Access 10 (2022) 86904–86919, http://dx.doi.org/10.1109/ACCESS.2022.3193102.

[58] P. Haindl, R. Plosch, Focus areas, themes, and objectives of non-functional requirements in DevOps: A systematic mapping study, in: A. Martini, M. Wimmer, A. Skavhaug (Eds.), Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Institute of Electrical and Electronics Engineers Inc., Johannes Kepler University Linz, Institute of Business Informatics - Software Engineering, Linz, Austria, 2020, pp. 394–403, http://dx.doi.org/10.1109/SEAA51224.2020.00071.

[59] E. Grunewald, P. Wille, F. Pallas, M. Borges, M.-R. Ulbricht, TIRA: An OpenAPI extension and toolbox for GDPR transparency in RESTful architectures, in: Proceedings - 2021 IEEE European Symposium on Security and Privacy Workshops, Euro S and PW 2021, 2021, pp. 312–319, http://dx.doi.org/10.1109/EuroSPW54576.2021.00039.

[60] J. Alonso, L. Orue-Echevarria, M. Huarte, CloudOps: Towards the operationalization of the cloud continuum: Concepts, challenges and a reference framework, Appl. Sci. (Switzerland) 12 (9) (2022) http://dx.doi.org/10.3390/app12094347.

[61] W. John, G. Marchetto, F. Nemeth, P. Skoldstrom, R. Steinert, C. Meirosu, I. Papafili, K. Pentikousis, Service provider DevOps, IEEE Commun. Mag. 55 (1) (2017) 204–211, http://dx.doi.org/10.1109/MCOM.2017.1500803CM.

[62] J. Dobaj, A. Riel, T. Krug, M. Seidl, G. Macher, M. Egretzberger, Towards digital twin-enabled DevOps for CPS providing architecture-based service adaptation & verification at runtime, in: Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems, in: SEAMS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 132–143, http://dx.doi.org/10.1145/3524844.3528057.

[63] A.A. Khan, M. Shameem, Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process, J. Softw.: Evol. Process 32 (10) (2020) http://dx.doi.org/10.1002/smr.2263.

[64] V. Singh, A. Singh, A. Aggarwal, S. Aggarwal, DevOps based migration aspects from legacy version control system to advanced distributed VCS for deploying micro-services, in: CSITSS 2021 - 2021 5th International Conference on Computational Systems and Information Technology for Sustainable Solutions, Proceedings, 2021, p. 5, http://dx.doi.org/10.1109/CSITSS54238.2021.9683718.

[65] Z. Sampedro, A. Holt, T. Hauser, Continuous integration and delivery for HPC: Using singularity and Jenkins, in: ACM International Conference Proceeding Series, Association for Computing Machinery, New York, NY, USA, 2018, p. 6, http://dx.doi.org/10.1145/3219104.3219147.

[66] M. Airaj, Enable cloud DevOps approach for industry and higher education, Concurr. Comput.-Pract. Exp. 29 (5) (2017) http://dx.doi.org/10.1002/cpe.3937.

[67] Y. Wang, M. Pyhäjärvi, M.V. Mäntylä, Test automation process improvement in a DevOps team: Experience report, in: 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2020, pp. 314–321, http://dx.doi.org/10.1109/ICSTW50294.2020.00057.

[68] B. Fitzgerald, K.-J. Stol, Continuous software engineering: A roadmap and agenda, J. Syst. Softw. 123 (2017) 176–189, http://dx.doi.org/10.1016/j.jss.2015.06.063.

[69] M. Gokarna, R. Singh, DevOps: A historical review and future works, in: P. Astya, M. Singh, N. Roy, G. Raj (Eds.), 2021 IEEE International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2021, Institute of Electrical and Electronics Engineers Inc., IEEE, IBM India Pvt Ltd, Manyata Tech Park, Bangalore, India, 2021, pp. 366–371, http://dx.doi.org/10.1109/ICCCIS51004.2021.9397235.

[70] A. Saboor, M. Hassan, R. Akbar, E. Susanto, S. Shah, M. Siddiqui, S. Magsi, Root-of-trust for continuous integration and continuous deployment pipeline in cloud computing, Comput. Mater. Contin. 73 (2) (2022) 2223–2239, http://dx.doi.org/10.32604/cmc.2022.028382.

[71] A. Alnafessah, A.U. Gias, R. Wang, L. Zhu, G. Casale, A. Filieri, Quality-aware DevOps research: Where do we stand? IEEE Access: Pract. Innov. Open Solutions 9 (2021) 44476–44489, http://dx.doi.org/10.1109/ACCESS.2021.3064867.

[72] J. Xuan, T. Duan, Q. Guo, F. Gao, J. Li, X. Qiu, S. Wu, Microservice publishing technology based on DevOps architecture, in: 2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC), Vol. 5, 2021, pp. 1310–1314, http://dx.doi.org/10.1109/ITNEC52019.2021.9586904.

[73] I. Kohyarnejadfard, D. Aloise, S.V. Azhari, M.R. Dagenais, Anomaly detection in microservice environments using distributed tracing data analysis and NLP, J. Cloud Comput. 11 (1) (2022) http://dx.doi.org/10.1186/s13677-022-00296-4.

[74] B. Snyder, B. Curtis, Using analytics to guide improvement during an Agile-DevOps transformation, IEEE Softw. 35 (1) (2017) 78–83, http://dx.doi.org/10.1109/MS.2017.4541032.

[75] M. Munoz, M. Negrete, M. Arcilla-Cobian, Using a platform based on the Basic profile of ISO/IEC 29110 to reinforce DevOps environments, J. Univ. Comput. Sci. 27 (2) (2020) 91–110, http://dx.doi.org/10.3897/jucs.65080.

[76] X. Chen, D. Badampudi, M. Usman, Reuse in contemporary software engineering practices-an exploratory case study in A medium-sized company, E-Inf. Softw. Eng. J. 16 (1) (2022) 220110, http://dx.doi.org/10.37190/e-Inf220110.

[77] A. Hemon, B. Lyonnet, F. Rowe, B. Fitzgerald, From Agile to DevOps: Smart skills and collaborations, Inf. Syst. Front. 22 (4) (2020) 927–945, http://dx.doi.org/10.1007/s10796-019-09905-1.

[78] S. Rafi, M.A. Akbar, W. Yu, A. Alsanad, A. Gumaei, M.U. Sarwar, Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis, Appl. Soft Comput. 116 (2022) 108377, http://dx.doi.org/10.1016/j.asoc.2021.108377.

[79] L. Banica, M. Radulescu, D. Rosca, A. Hagiu, Is DevOps another project management methodology? Inf. Econ. 21 (3) (2017) 39–51, http://dx.doi.org/10.12948/issn14531305/21.3.2017.04.

[80] A. Hemon, B. Fitzgerald, B. Lyonnet, F. Rowe, Innovative practices for knowledge sharing in large-scale DevOps, IEEE Softw. 37 (3) (2020) 30–37, http://dx.doi.org/10.1109/MS.2019.2958900.

[81] I. Jimenez, M. Sevilla, N. Watkins, C. Maltzahn, J. Lofstead, K. Mohror, A. Arpaci-Dusseau, R. Arpaci-Dusseau, The Popper convention: Making reproducible systems evaluation practical, in: Proceedings - 2017 IEEE 31st International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2017, 2017, pp. 1561–1570, http://dx.doi.org/10.1109/IPDPSW.2017.157.

[82] I. Kumara, M. Garriga, A.U. Romeu, D. Di Nucci, F. Palomba, D.A. Tamburri, W.-J. van den Heuvel, The do's and don'ts of infrastructure code: A systematic gray literature review, Inf. Softw. Technol. 137 (2021) 106593, http://dx.doi.org/10.1016/j.infsof.2021.106593.

[83] Y. Zhou, Y. Su, T. Chen, Z. Huang, H.C. Gall, S. Panichella, User review-based change file localization for mobile applications, IEEE Trans. Softw. Eng. (2020) 1, http://dx.doi.org/10.1109/TSE.2020.2967383.

[84] M. Chen, W. Yao, J. Chen, H. Liang, Y. Chen, H. Qiao, C. Yang, M. Li, J. Tong, Critical challenges and solutions for an ultra-large-scale enterprise DevOps platform, in: 2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), 2022, pp. 167–171, http://dx.doi.org/10.1109/ICCCBDA55098.2022.9778937.

[85] L. Firdaouss, B. Ayoub, B. Manal, Y. Ikrame, Automated VPN configuration using DevOps, in: Procedia Computer Science, in: 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare, 198, 2022, pp. 632–637, http://dx.doi.org/10.1016/j.procs.2021.12.298.

[86] L.-N. Lévy, J. Bosom, G. Guerard, S. Amor, M. Bui, H. Tran, DevOps model appproach for monitoring smart energy systems, Energies 15 (15) (2022) 27, http://dx.doi.org/10.3390/en15155516.

[87] R. Jabbari, N. bin Ali, K. Petersen, B. Tanveer, Towards a benefits dependency network for DevOps based on a systematic literature review, J. Softw.: Evol. Process 30 (11) (2018) 26, http://dx.doi.org/10.1002/smr.1957.

[88] J. Sandobalin, E. Insfran, S. Abrahao, J. Sandobalín, E. Insfran, S. Abrahão, On the effectiveness of tools to support infrastructure as code: model-driven versus code-centric, IEEE Access 8 (2020) 17734–17761, http://dx.doi.org/10.1109/ACCESS.2020.2966597.

[89] A. Trigo, J. Varajão, L. Sousa, DevOps adoption: Insights from a large European Telco, Cogent Eng. 9 (1) (2022) http://dx.doi.org/10.1080/23311916.2022.2083474.

[90] M.F. Lie, M. Sanchez-Gordon, R. Colomo-Palacios, DevOps in an ISO 13485 regulated environment: A multivocal literature review, in: International Symposium on Empirical Software Engineering and Measurement, ACM, New York, NY, USA, 2020, p. 11, http://dx.doi.org/10.1145/3382494.3410679.

[91] E.E. Romero, C.D. Camacho, C.E. Montenegro, Ó.E. Acosta, R.G. Crespo, E.E. Gaona, M.H. Martínez, Integration of DevOps practices on a noise monitor system with CircleCI and Terraform, ACM Trans. Manag. Inf. Syst. 13 (4) (2022) 36:1–36:24, http://dx.doi.org/10.1145/3505228.

[92] F. Almeida, J. Simões, S. Lopes, Exploring the benefits of combining DevOps and Agile, Fut. Int. 14 (2) (2022) 63, http://dx.doi.org/10.3390/fi14020063.

[93] M.A.A. Alamin, G. Uddin, S. Malakar, S. Afroz, T. Haider, A. Iqbal, Developer discussion topics on the adoption and barriers of low code software development platforms, Empir. Softw. Eng. 28 (1) (2022) http://dx.doi.org/10.1007/s10664-022-10244-0.

[94] S. Badshah, A.A. Khan, B. Khan, Towards process improvement in DevOps: A systematic literature review, in: 24th Evaluation and Assessment in Software Engineering Conference, EASE 2020, ACM, Association for Computing Machinery, Comsats University Islamabad, Islamabad, Pakistan, 2020, pp. 427–433, http://dx.doi.org/10.1145/3383219.3383280.

[95] L.E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo, C. Lassenius, DevOps in practice: A multiple case study of five companies, Inf. Softw. Technol. 114 (March 2017) (2019) 217–230, http://dx.doi.org/10.1016/j.infsof.2019.06.010.

[96] I.-C. Donca, O.P. Stan, M. Misaros, D. Gota, L. Miclea, Method for continuous integration and deployment using a pipeline generator for agile software projects, Sensors (Basel, Switzerland) 22 (12) (2022) http://dx.doi.org/10.3390/s22124637.

[97] S. Rafi, W. Yu, M.A. Akbar, A. Alsanad, A. Gumaei, Multicriteria based decision making of DevOps data quality assessment challenges using fuzzy TOPSIS, IEEE Access 8 (1) (2020) 46958–46980, http://dx.doi.org/10.1109/ACCESS.2020.2976803.

[98] P. Perera, R. Silva, I. Perera, Improve software quality through practicing DevOps, in: 17th International Conference on Advances in ICT for Emerging Regions, ICTer 2017 - Proceedings, 2018-Janua, IEEE, Institute of Electrical and Electronics Engineers Inc., Department of Computer Science aSoftware Development model nd Engineering, University of Moratuwa, Moratuwa, Sri Lanka, 2017, pp. 13–18, http://dx.doi.org/10.1109/ICTER.2017.8257807.

[99] S. Rafi, W. Yu, M. Akbar, A. Alsanad, A. Gumaei, Prioritization based taxonomy of DevOps Security Challenges Using PROMETHEE, IEEE Access 8 (2020) 105426–105446, http://dx.doi.org/10.1109/ACCESS.2020.2998819.

[100] M.A. Akbar, S. Rafi, A.A. Alsanad, S.F. Qadri, A. Alsanad, A. Alothaim, Toward successful DevOps: A decision-making framework, IEEE Access: Pract. Innov. Open Solutions 10 (2022) 51343–51362, http://dx.doi.org/10.1109/ACCESS.2022.3174094.

[101] S. Dallapalma, D. Di Nucci, F. Palomba, D.A. Tamburri, Within-project defect prediction of infrastructure-as-code using product and process metrics, IEEE Trans. Softw. Eng. (2021) 1, http://dx.doi.org/10.1109/TSE.2021.3051492.

[102] A. Wiedemann, M. Wiesche, H. Gewald, H. Krcmar, Understanding how DevOps aligns development and operations: A tripartite model of intra-IT alignment, Eur. J. Inf. Syst. 29 (5) (2020) 458–473, http://dx.doi.org/10.1080/0960085X.2020.1782277.

[103] S. Rafi, M.A. Akbar, S. Mahmood, A. Alsanad, A. Alothaim, Selection of DevOps best test practices: A hybrid approach using ISM and fuzzy TOPSIS analysis, J. Softw.: Evol. Process 34 (5) (2022) e2448, http://dx.doi.org/10.1002/smr.2448.

[104] S. Throner, H. Hutter, N. Sanger, M. Schneider, S. Hanselmann, P. Petrovic, S. Abeck, An advanced DevOps environment for microservice-based applications, in: 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2021, pp. 134–143, http://dx.doi.org/10.1109/SOSE52839.2021.00020.

[105] O. Zimmermann, Microservices tenets, Comput. Sci. - Res. Dev. 32 (3–4) (2017) 301–310, http://dx.doi.org/10.1007/s00450-016-0337-0.

[106] W.P. Luz, G. Pinto, B. Bonifacio, 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 (ESEM 2018), Oulu, Finland, 2018, p. 11, http://dx.doi.org/10.1145/3239235.3240299.

[107] A. Al-marsy, P. Chaudhary, J. Rodger, A model for examining challenges and opportunities in use of cloud computing for health information systems, Appl. Syst. Innov. 4 (1) (2021) 20, http://dx.doi.org/10.3390/asi4010015.

[108] A. Premchand, M. Sandhya, S. Sankar, Simplification of application operations using cloud and DevOps, Indonesian J. Electr. Eng. Comput. Sci. 13 (1) (2019) 85–93, http://dx.doi.org/10.11591/ijeecs.v13.i1.pp85-93.

[109] K. Tuma, C. Sandberg, U. Thorsson, M. Widman, T. Herpel, R. Scandariato, Finding security threats that matter: Two industrial case studies, J. Syst. Softw. 179 (2021) 111003, http://dx.doi.org/10.1016/j.jss.2021.111003.

[110] B.Y. Chen, Z.M. Jiang, A survey of software log instrumentation, ACM Comput. Surv. 54 (4) (2021) http://dx.doi.org/10.1145/3448976.

[111] Y. Liu, Z. Ling, B. Huo, B. Wang, T. Chen, E. Mouine, Building A platform for machine learning operations from open source frameworks, in: IFAC-PapersOnLine, in: 3rd IFAC Workshop on Cyber-Physical & Human Systems CPHS 2020, Vol. 53, 2020, pp. 704–709, http://dx.doi.org/10.1016/j.ifacol.2021.04.161.

[112] H. Topi, G. Spurrier, Invited paper: a generalized, enterprise-level systems development process framework for systems analysis and design education, J. Inf. Syst. Educ. 30 (4) (2019) 253–265.

[113] H. Topi, G. Spurrier, A generalized, enterprise-level systems development process framework for systems analysis and design education, J. Inf. Syst. Educ. 30 (4) (2019) 253–265.

[114] M.A. Akbar, K. Smolander, S. Mahmood, A. Alsanad, Toward successful DevSecOps in software development organizations: A decision-making framework, Inf. Softw. Technol. 147 (2022) 106894, http://dx.doi.org/10.1016/j.infsof.2022.106894.

[115] R. Ranawana, A.S. Karunananda, An agile software development life cycle model for machine learning application development, in: 2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI), 2021, p. 6, http://dx.doi.org/10.1109/SLAAI-ICAI54477.2021.9664736.

[116] H. Liu, Q. Han, Y. Wang, F. He, Z. Mao, C. Li, An analysis of DevOps architecture for EMIS based on jBPM, in: 2020 International Conference on Service Science (ICSS), 2020-Augus, IEEE, 2020, pp. 96–101, http://dx.doi.org/10.1109/ICSS50103.2020.00023.

[117] M. Camilli, A. Guerriero, A. Janes, B. Russo, S. Russo, Microservices integrated performance and reliability testing, in: Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test, in: AST '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 29–39, http://dx.doi.org/10.1145/3524481.3527233.

[118] S.T. Lai, F.Y. Leu, A micro services quality measurement model for improving the efficiency and quality of DevOps, in: L. Barolli, F. Xhafa, N. Javaid, T. Enokido (Eds.), Advances in Intelligent Systems and Computing, Vol. 773, Springer International Publishing AG, Gewerbestrasse 11, CHAM, CH-6330, SWITZERLAND, 2019, pp. 565–575, http://dx.doi.org/10.1007/978-3-319-93554-6_55.

[119] R. Eramo, V. Muttillo, L. Berardinelli, H. Bruneliere, A. Gomez, A. Bagnato, A. Sadovykh, A. Cicchetti, AIDOaRt: AI-augmented automation for DevOps, a model-based framework for continuous development in cyber-physical systems, in: 2021 24th Euromicro Conference on Digital System Design (DSD), 2021, pp. 303–310, http://dx.doi.org/10.1109/DSD53832.2021.00053.

[120] M. Pardo, H. Erazo, C. Lozada, Documenting and implementing DevOps good practices with test automation and continuous deployment tools through software refinement, Period. Eng. Nat. Sci. 9 (4) (2021) 854–863, http://dx.doi.org/10.21533/pen.v9i4.2239.

[121] C. Vassallo, F. Zampetti, D. Romano, M. Beller, A. Panichella, M. Di Penta, A. Zaidman, Continuous delivery practices in a large financial organization, in: Proceedings - 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, 2017, pp. 519–528, http://dx.doi.org/10.1109/ICSME.2016.72.

[122] J. Chen, Performance regression detection in DevOps, in: Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: Companion, ICSE-Companion 2020, 2020, pp. 206–209, http://dx.doi.org/10.1145/3377812.3381386.

[123] J. Ayerdi, A. Garciandia, A. Arrieta, W. Afzal, E. Enoiu, A. Agirre, G. Sagardui, M. Arratibel, O. Sellin, Towards a taxonomy for eliciting design-operation continuum requirements of cyber-physical systems, in: Proceedings of the IEEE International Conference on Requirements Engineering, 2020-August, 2020, pp. 280–290, http://dx.doi.org/10.1109/RE48521.2020.00038.

[124] C. Paule, T.F. Dullmann, A. Van Hoorn, Vulnerabilities in continuous delivery pipelines? a case study, in: Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019, 2019, pp. 102–108, http://dx.doi.org/10.1109/ICSA-C.2019.00026.

[125] M. Wöhrer, U. Zdun, DevOps for ethereum blockchain smart contracts, in: 2021 IEEE International Conference on Blockchain (Blockchain), 2021, pp. 244–251, http://dx.doi.org/10.1109/Blockchain53845.2021.00040.

[126] I.D. Rubasinghe, D.A. Meedeniya, I. Perera, Towards traceability management in continuous integration with sat-analyzer, in: ACM International Conference Proceeding Series, Association for Computing Machinery, New York, NY, USA, 2017, pp. 77–81, http://dx.doi.org/10.1145/3162957.3162985.

[127] J. Bergelin, A. Cicchetti, Towards continuous modelling to enable DevOps: A preliminary study with practitioners, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, in: MODELS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 774–783, http://dx.doi.org/10.1145/3550356.3561582.

[128] C. Castellanos, C.A. Varela, D. Correal, ACCORDANT: A domain specific-model and DevOps approach for big data analytics architectures, J. Syst. Softw. 172 (2021) http://dx.doi.org/10.1016/j.jss.2020.110869.

[129] R. Subramanya, S. Sierla, V. Vyatkin, From DevOps to MLOps: Overview and application to electricity market forecasting, Appl. Sci. (Switzerland) 12 (19) (2022) http://dx.doi.org/10.3390/app12199851.

[130] J. Faustino, R. Amaro, D. Adriano, R. Pereira, M.M. da Silva, DevOps benefits: A systematic literature review, Softw. - Pract. Exp. 52 (9) (2022) 1905–1926, http://dx.doi.org/10.1002/spe.3096.

[131] R. Amaro, R. Pereira, M.M. da Silva, DevOps Metrics and KPIs: A multivocal literature review, ACM Comput. Surv. (2024) http://dx.doi.org/10.1145/3652508.

[132] G. Guest, E. Namey, M. Chen, A simple method to assess and report thematic saturation in qualitative research, PLOS ONE 15 (5) (2020) e0232076, http://dx.doi.org/10.1371/journal.pone.0232076.

[133] V. Garousi Yusifoğlu, Y. Amannejad, A. Betin Can, Software test-code engineering: A systematic mapping, Inf. Softw. Technol. 58 (2015) 123–147, http://dx.doi.org/10.1016/j.infsof.2014.06.009.

[134] J.M. Verner, M.A. Babar, N. Cerpa, T. Hall, S. Beecham, Factors that motivate software engineering teams: A four country empirical study, J. Syst. Softw. 92 (1) (2014) 115–127, http://dx.doi.org/10.1016/j.jss.2014.01.008.