INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Department of Information Science and Technology

Achieving Successful DevOps Adoption in IT Organizations

Ricardo Manuel Duarte Amaro

PhD in Information Science and Technology

Supervisors:
Doctor Rúben Filipe de Sousa Pereira, Assistant Professor,
ISCTE - Instituto Universitário de Lisboa

Doctor Miguel Leitão Bignolas Mira da Silva, Full Professor,
IST - Instituto Superior Técnico

September, 2024

iscte

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

Achieving Successful DevOps Adoption in IT Organizations

Ricardo Manuel Duarte Amaro

PhD in Information Science and Technology

Jury:
Doctor Fernando Brito e Abreu, Associate Professor (President),
ISCTE - Instituto Universitário de Lisboa
Doctor Jessica Díaz, Associate Professor,
Universidad Politécnica de Madrid
Doctor António Rito Silva, Associate Professor,
Instituto Superior Técnico
Doctor Vitor Basto-Fernandes, Associate Professor w/ Aggreg.,
ISCTE - Instituto Universitário de Lisboa
Doctor Rúben Filipe de Sousa Pereira, Assistant Professor,
ISCTE - Instituto Universitário de Lisboa

September, 2024

To my loving family, Dália, Vicente, and Maria Rita.
Without your unconditional support, I would never have completed this thesis successfully.

Acknowledgment

Where to begin? Every journey starts with a small step. And, as some predicted, this journey has been like walking through the desert. However, as with any desert, you can cross it if you have strong motivation and enough resources. Both things I owe to my family, colleagues, friends, teachers, and anonymous individuals who helped me succeed. Looking back, I believe it eventually allowed me to bring something special to share with you all through this acknowledgment.

To commence, I am grateful to my professors Rúben Pereira and Miguel Mira da Silva for challenging me and providing guidance throughout my doctoral studies. Their knowledge and invaluable support enabled me to successfully complete my doctorate.

I would want to offer my heartfelt gratitude to my wife Dália for all of her love, commitment, patience, and assistance in making this work a success, as well as for being by my side during the most difficult times in my life. Also, thank you to my son Vicente and daughter Maria Rita for all of the joy, support, and kindness you bring into my life, inspiring me every day to make the world a better place.

Thank you to my family, including father João, mother Maria, brother Bruno, father and mother-in-law, brother and sister-in-law, nephew, uncles, and cousins, for your unwavering support, strength, and affection.

Additionally, I would like to extend my appreciation to all the fiends and colleagues I have known in Acquia, Drupal, DevOps and in the Free and Open source community. This is another contribution back for all that you have given me along so many years. Together, we continue to shape the world.

Finally, I want to gratefully acknowledge the organizations and people who helped with this thesis.

This is dedicated to each of you. Thank you for being an important part of this incredible journey.

Resumo

No contexto dinâmico das organizações de TI, DevOps — desenvolvimento (Dev) e operações de TI (Ops) — tornou-se uma estratégia crítica para facilitar as necessidades crescentes de eficiência contínua e ótima no desenvolvimento e entrega de software nas organizações. Esta necessidade é impulsionada por exigências dos clientes, a concorrência intensa, as ameaças externas sofisticadas e a legislação rigorosa. Apesar dessa importância reconhecida, ainda existem desafios de adoção e trabalho empírico limitado sobre a forma como as capacidades e métricas inerentes impulsionam a adoção bem sucedida de DevOps, levando a resultados inconsistentes nas organizações.

Esta tese tem como objetivo explorar a forma de melhorar e conseguir uma adoção bem sucedida do DevOps nas organizações de TI, centrando-se nas principais capacidades, métricas e processos do ciclo de vida do DevOps. As questões de investigação apontam para encontrar os vetores relacionais mais impactantes e como as organizações podem aplicá-los para superar os desafios de adoção. São utilizados vários métodos de investigação, como revisões de literatura, entrevistas, análise de documentos, estudo de caso, grupos de foco e Design Science Research (DSR) ao longo de seis artigos submetidos a revistas científicas.

O trabalho resultante oferece uma descrição detalhada de como entender e implementar DevOps, fornecendo valor a profissionais da indústria, peritos, e investigadores. Uma estrutura é proposta com o objetivo de abordar os principais desafios através do desenvolvimento de estratégias de melhoria contribuindo para o desenvolvimento e a entrega de software fiável e de elevado desempenho. Os artigos desenvolvem e comunicam o conjunto organizado de conhecimentos necessários para a adoção bem sucedida de DevOps.

Palavras-chave: Capacidades e práticas de DevOps, Métricas de DevOps, Processo de engenharia de software, Gestão e entrega de software, Desenvolvimento de software, Desempenho de entrega de software.

Abstract

In the dynamic Information Technology (IT) landscape, DevOps — development (Dev) and IT operations (Ops) — has become a critical strategy to facilitate the increasing needs for continuous and optimal efficiency in software development and delivery across organizations. This need is driven by constraints such as customer demands, intense competition, sophisticated external threats, and strict government legislation. Despite its acknowledged importance, there are still adoption challenges, and limited empirical work on how inherent capabilities and metrics drive successful DevOps adoption, leading to inconsistent results across organizations.

This thesis aims to explore how to improve and achieve successful DevOps adoption in IT organizations by focusing on the main DevOps capabilities, metrics, and life cycle processes. The research questions point to finding the most impacting relational vectors and how organizations can apply those to overcome adoption challenges. Multiple research methods are used, like literature reviews, interviews, document analysis, case study, focus group and Design Science Research (DSR) over six articles submitted to scientific journals.

The resulting work offers a detailed description of how to understand and implement DevOps, providing value to industry professionals, experts, and researchers. A framework is proposed focusing on addressing the main challenges by developing improvement strategies contributing to reliable and high-performing software development and delivery. The articles develop and communicate the organized body of knowledge required for successful DevOps adoption.

Keywords: DevOps Capabilities and Practices, DevOps Metrics, DevOps Adoption, Software Engineering Process, Software release management and delivery, Software Delivery Performance.

Contents

List of Figures

List of Tables

List of Code

List of Acronyms

CALMS	Culture, Automation, Lean principles, Measuring and Sharing
CAMS	Culture, Automation, Measuring and Sharing
CD	*Continuous Delivery or Deployment*
CFR	Change Failure Rate
CI/CD	Continuous Integration & Delivery or Deployment
DevOps	Developer (Dev) and Operations (Ops)
DevSecOps	Developer (Dev), Security (Sec) and Operations (Ops)
DF	Deployment Frequency
DORA	DevOps Research and Assessment
DSR	Design Science Research
FLOSS	Free/Libre and Open Source Software
IaC	Infrastructure as Code
IT	Information Technology
KPI	Key Performance Indicator
LCP	Life Cycle Process
MLR	Multivocal Literature Review
MLT	Mean Lead-time for Changes
MSA	Microservices Architecture
MTTD	Mean Time To Detection
MTTR	Mean Time To Recover/Restore
QA	Quality Assurance
SDLC	Software Development Life Cycle
SD	Software Development
SLI	Service level Indicator
SLO	Service level Objective
SLR	Systematic Literature Review
SRE	Site Reliability Engineering
WIP	Work in Progress

CHAPTER 1

Introduction

This thesis builds upon the research initiated during the completion of the Master's Degree in *Information and Enterprise Systems*. The intricate issues identified and highlighted earlier, along with the fundamental goals of this study, are also reflected in this introduction, indicating a work of critical and contextual analysis. Therefore, the initial exploration of motivation demonstrates the significant importance and intellectual curiosity surrounding the subject being studied, with the aim of offering specific responses to the problem and research questions.

1.1 Motivation

In an increasingly competitive world of software development, delivery and support, the DevOps movement has emerged as a fundamental strategy, driven by the development (Dev) and operations (Ops) teams themselves[1], with a vision of achieving continuous and optimal efficiency[2]. More specifically, DevOps is a growing trend resulting from the alignment between companies and the constantly evolving IT, which face challenges arising from the needs of their customers, under intense competition, sophisticated external threats and strict government legislation [3]. At the same time, they seek to establish a competitive advantage by delivering and supporting software faster and more efficiently than their competitors. In this context, it has been observed that the acceleration of continuous software delivery and support without an integration of DevOps capabilities [4] can compromise the reliability and performance of the activity, unless it is accompanied by consistent builds, adequate testing and release automation[5]. All of this in turn underlines the crucial importance of adopting DevOps for the development and delivery of software in a consistent and predictable manner, as well as for the adaptability, monitoring, reliability and security of systems[6].

Contextually, agile methods, through their manifesto, began to influence software development in organizations at the beginning of the 21st century, having had a positive impact by improving collaboration with the customer and accelerating development iteratively in development teams [7]. Therefore, this acceleration ended up highlighting the misalignment between the development and operations teams, leading to more friction and silos between these teams [8, 9]. With this sudden acceleration in software delivery, the operations teams, accustomed to slower and more spaced releases and updates, experienced a substantial detriment to their quality of life and service, due to the increase in recurring errors in production, which escaped due to a lack of consistency in testing and compilation. These facts are well portrayed in iconic books such as "The Phoenix Project" [10] or "Continuous Delivery" [5]. In this context, DevOps emerges as a dialectical response, seeking to extend the benefits of the Agile movement and find a healthy balance between the worlds of development and operations, keeping the focus on both of their antagonistic goals: development teams want faster deliveries, while operations teams want more stability. DevOps has thus become a widely recognized concept in the last decade, promoting an organizational approach that values empathy and encourages closer collaboration between the teams in charge of delivering software [11]. This effort seeks not only to provide a better end-user experience but also to decrease development time, increase delivery rates, improve stability and optimize processes while reducing Software Development Life Cycle (SDLC) costs and decreasing the average time to recover from problems [12].

1.2 Research Problem

However, despite the growing consensus in the industry and in academia on how DevOps is a crucial factor in improving software development and efficiency [13, 14], there is still little empirical evidence of how inherent capabilities and metrics influence successful DevOps adoption. This is reflected in the inconsistency of results obtained when trying to adopt DevOps in different organizations. Several recent studies point out challenges and complications [15–18], namely because it is necessary to provide organizational leaders with the appropriate strategies, together with information and tools, so that they can overcome adoption challenges within the software life cycle [19, 20].

On the other hand, to support the effective use of DevOps in organizations, a clear understanding and guidance is needed [16]. For leadership to be able to apply improvements to the process, it is crucial to have a thorough understanding of the processes to follow, based on concrete data, to increase efficiency and facilitate the successful implementation of DevOps capabilities. This challenge necessitates a rigorous self-assessment and continuous monitoring through efficacy metrics to enhance maturity and performance levels.

Therefore, the research problem and questions of this thesis, synthesized in Figure 1.1, are:

"RQ1: What are the key DevOps capabilities, metrics, and processes that have the most positive impact on DevOps adoption?"

"RQ2: How can organizations effectively apply key DevOps capabilities, metrics, and processes to overcome adoption challenges?"

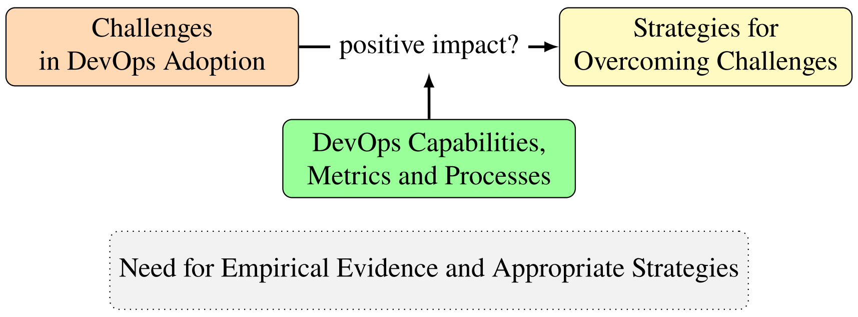

Challenges in DevOps Adoption
positive impact?
Strategies for Overcoming Challenges
DevOps Capabilities, Metrics and Processes
Need for Empirical Evidence and Appropriate Strategies

Figure 1.1: Research Problem: Inconsistent DevOps Adoption Success

1.3 Objective

Hence, the general objective of this thesis is to investigate how to improve and achieve a successful adoption of DevOps in IT organizations. The specific objectives mentioned in each of the articles outline the stages of this task to be accomplished through exploratory, descriptive and explanatory processes.

This thesis is grounded on the idea that it is possible to provide strategies for successful DevOps adoption if we find a strong and impactful relation between the concepts involved in the adoption of DevOps [4, 17, 21–24]. Figure 1.2 shows the concepts of DevOps Capabilities, DevOps Metrics, Life cycle Processes, Strategies, DevOps Benefits, DevOps Challenges, and Successful DevOps Adoption since these are the core elements identified in the research as critical to understanding and implementing DevOps successfully.

- **DevOps Capabilities** are the required skills and practices for executing DevOps[4].
- **DevOps Metrics** measure the effectiveness of DevOps capabilities[22].
- **Life Cycle Processes** are the activities and tasks around software and systems improved by DevOps capabilities[23].
- **DevOps Strategies** are hereafter defined as the initiatives and drivers which facilitate DevOps transformation [25].
- **DevOps Benefits** are the result or the gain from the use of DevOps [4].
- **DevOps Challenges** are the conditions restricting the implementation of DevOps[4].
- **Successful DevOps Adoption** is the intended research outcome showing the effective execution of DevOps along with the benefits to be received and challenges to be minimized.

The conceptual structure in Figure 1.2 proposes that DevOps capabilities influence the selection and use of metrics observed in measuring their effectiveness [25, 26]. Metrics, on the other hand, are the indicators of information that can help to enhance life cycle processes [23]. Capabilities define the strategies that must be executed, and these strategies deliver on relevant benefits [27]. But attaining gains also implies facing challenges [28] and when difficulties are overcome, it can, in fact, bring advances to the life cycle processes [23]. This is the reason behind the feedback loops [29] from Metrics to Capabilities and from Life cycle processes to metrics, in order to demonstrate that the feedback cycle has improved as reported on Chapter 6. Finally, the arrows to successful DevOps adoption indicate that in the end all efforts will lead into a successful adoption of DevOps being the central target [28].

This is a compass or navigation tool for navigating the interrelationships of capabilities, metrics, life cycle processes, strategies, benefits and challenges. When seen together, these

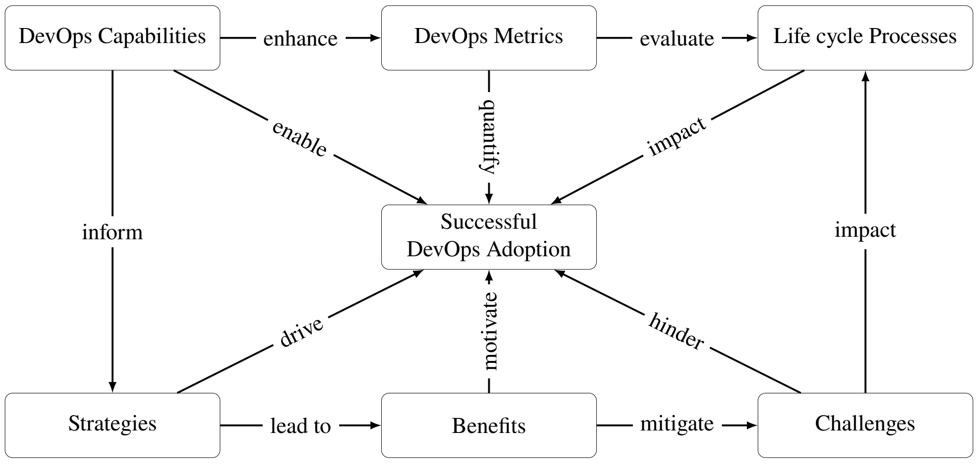


Figure 1.2: DevOps Adoption Concepts.

relationships show organizations and researchers what factors influence successful DevOps adoption and how this view can help them plan future strategies.

This thesis generated several research questions that were answered throughout six articles. Intended for publication in periodic scientific journals of this area. As a result, new and organized knowledge is generated that is important for professionals, industry, and researchers.

1.4 Research Background

This thesis, as previously mentioned, is focused on the successful adoption of DevOps while considering all of its associated concepts. Consequently, this section provides a comprehensive overview of these concepts, with a particular emphasis on the interconnected theories that have influenced and furthered their development. It is the objective of this section to establish a theoretical background that enables the subsequent chapters to be more comprehensive and detailed in their discussion and analysis.

1.4.1 DevOps Adoption

Prior to exploring the concept of DevOps adoption, it is crucial to have a comprehensive understanding of DevOps itself. The term "DevOps" refers to coordination and cooperation between development (Dev) and operations (Ops) teams. Together, these engineering teams work to eliminate the so-called "information silos" [30], based on a cultural shift initially advocated by Patrick Debois [1], in order to achieve collaboration, automation, and integration between software developers and IT professionals, fostering a culture that promotes agility, reliability, and security in technology organizations. Thus improving the efficacy of the life cycle process and delivering high-quality software continuously.

Jabbari et al. (2016) [31] mentioned that DevOps is a development approach that bridges the gap between development and operations by leveraging communication, collaboration, continuous integration, automated delivery, and consideration for quality assurance through automated delivery. Humble et al. (2010) [5] suggested prioritizing Culture, Automation, Measuring and Sharing (CAMS), and later, they have added the Lean (L) to these four building blocks, resulting in Culture, Automation, Lean principles, Measuring and Sharing (CALMS) [32].

More recently, the IEEE Std 2675-2021 [23] introduced a standardized definition of DevOps: a set of principles and practices that make it easier for teams involved to communicate and collaborate, with the goal of creating, developing, and running software systems, products, or services while continuously improving software and system life cycle processes.

According to Díaz et al. (2021) [19], DevOps is a cultural and professional movement that aims at improving the delivery of business value. Since the inception of DevOps, authors have looked at different ways to define the term, and they include the various practices that enable the software delivery process to be automated. DevOps is generally regarded as a methodology that focuses on continuous integration, continuous delivery, and rapid deployment to adapt to the dynamism of the market [4, 33]. For a delivery to be considered high-quality, Luz et al. (2019) argue that DevOps needs to be implemented within a system, something that results from the collaboration between development and operation teams in any organization [25]. A cultural shift is needed to break down silos, enhance communication, and embrace a true collaborative environment within organizations, as it promotes the close working of the development and operations teams. DevOps, in the software industry, prioritizes the integration of development and client support, enabling teams to collaborate effectively. The advantages of DevOps include reduced development time, increased deployment frequency, enhanced stability, shorter delivery cycles, and cost savings in both delivery and execution [29].

The DevOps adoption process is used in organizations to integrate DevOps capabilities and practices into their software development and IT operations. This integration is to facilitate better collaboration between teams, automate the software delivery process overall by improving the speed and quality of releases [34–36]. Several authors in the literature make reference to the term DevOps adoption. It stands for the process when organizations incorporate DevOps capabilities and practices into their software development and IT operations [34, 37]. DevOps adoption introduces cultural changes, adding new tools, and changing processes to support continuous integration, delivery, and deployment [2, 38, 39].

Per Maroukian and Gulliver (2020) [39] successful DevOps adoption requires an organizational shift in organizational mindset, skillset, and toolset, where more concern about collaborating rather than just focusing on things like containerization, automation, or tooling [25]. Transformational and servant leadership are related features of leaders who can influence the adoption or resistance to DevOps [39]. In regulated domains like medical device development, DevOps adoption could be a big challenge, but it can outweigh conventional practices, especially in compliance and security aspects [26, 40].

In further reviewing related work previous to this research, authors write about various aspects of DevOps adoption and implementation, including the challenges and processes involved as explained in Section 1.4.5. Although implementing DevOps can offer many benefits such as faster releases with fewer deployment errors and better incident handling [18], it can be difficult based on cultural, organizational and technological reasons [19, 41].

Bucena et al. (2017) [13] conducted a study that focused on both the challenges of DevOps adoption and methods for simplifying DevOps adoption. Maroukian et al. [39] concentrated on the latter issue, discussing aspects of leading the adoption of DevOps practices and principles. Rafi et al. (2020) [38] presented RMDevOps, which could be a specific model or framework for DevOps implementation. Another empirical study by Zarour et al. [42] focused on the case of adopting a DevOps process model in Saudi Arabia, implying a possible assisted replication that could directly or indirectly support implementation. While not presenting a model, Smeds et al. (2015) [12] provide a clear definition of DevOps and identify impediments to its adoption, introducing failures that should help organizations anticipate and overcome such barriers.

Perez et al. (2022) argue that academic explanations may be far from reality and not generalizable to real productive environments, DevOps adoption is an ideal candidate for nontraditional ways of teaching that "bring" industry experience to DevOps courses in some way [43]. Leite et al. (2019) [17] finds that there is poor discussion in academia on the technical implications and complexity of adopting DevOps practices such as automation, microservices architectures, containerization, and toolset management. Practical DevOps concepts and implications empower managers to make assertive and strategic decisions and provide engineers with best practices for adoption. Adopting new tools takes time and effort, and doing so without considering the results can waste it. A DevOps adoption process must prioritize team structure over tool selection.

Despite the mentioned research, the adoption of DevOps is still a challenging task and none of the previous authors as approached solving the challenges of this task from a three vector perspective, effectively researching empirical evidence of the impactful relations between DevOps capabilities, metrics and life cycle processes.

Several aspects of DevOps adoption still need to be addressed. The first step is to integrate continuous security practices. It is supported by a conceptual model for automated Developer (Dev), Security (Sec) and Operations (Ops) (DevSecOps) on the cloud that employs Free/Libre and Open Source Software (FLOSS) mentioned by Kumar et al. (2020) [44]. The next aspect is that there are many challenges, like code and data quality in DevOps, as observed by Rafi et al. (2020) in applying their fuzzy analysis in the context of the DevOps challenges [38]. Meaning, the adoption of cutting-edge DevOps practices and trends should also be prioritized [39]. The final aspect is the transition to enterprise-scale agile software development, which is frequently achieved through DevOps, and the lessons learned from this process can help with DevOps adoption [45]. To achieve a successful DevOps transformation, the approach should be aligned with the organization's dynamics, internal and external competitive forces, and constraints [23].

1.4.2 DevOps Capabilities and Practices

The literature indicates a progression and refinement of capabilities, practices and tools to improve software delivery and operational performance. Overall, the studies on DevOps capabilities over the years presented in Figure 2.8 suggest there is an ever-growing interest in exploring what DevOps uncovers from its evolving nature. On the other hand, the morphing lists of capabilities seen in Tables 2.4 and 2.5 and 2.6 suggest that DevOps capabilities have evolved to become more sophisticated, with a greater emphasis on automation, collaboration, and continuous improvement to meet the demands of modern software development and delivery [4, 12, 46].

The 2016 DevOps Handbook recommends automation, continuous integration, delivery, and collaboration, to assist IT organizations in adopting DevOps for increased agility, dependability, and security [29]. According to the 2019 State of DevOps Report [47], the sector has matured, and there is an increasing interest in quantifying performance [22] leading to the importance of also exploring DevOps metrics in Article 2 to measure capabilities. Puppet Labs DevOps report (2019) [48] discovered that top performers improve automation, integration, and delivery while also using key metrics. This thesis seeks to standardize DevOps capabilities and practices that are currently ambiguous and interchangeable in the Multivocal Literature Review (MLR) of Article 1.

The existing studies on DevOps capabilities and practices, make use of both terms, and are vague and ambiguous regarding their definitions. Thus, the empirical evidence required to attain DevOps practices that enable capabilities, while improving software development and deployment speed, quality, and reliability while encouraging continuous improvement [39, 44, 49, 50]. In the absence of research on the main capabilities and practices of DevOps, as well as their distinctions, this study sheds light on how and why practitioners should adopt them, as well as what aspects they should change or keep in mind. Teixeira et al. (2020) [14] mentioned that future research should be carried out into the most referenced capabilities, while other authors indicate that several DevOps capabilities and practices aspects still pose challenges and require consensus and solutions like aligning DevOps principles and practices with organizational culture and goals to ensure commitment from stakeholders, developing lightweight model-based testing methods, prioritizing team collaboration, tool chain integration, and quality at speed [39, 44, 51–53].

Lastly, a few other commonly mentioned capabilities include: Software Delivery Process Automation, which is said to enhance software release speed, quality, and reliability [14, 18]. Collaboration between Development and Operations Teams, promoting Continuous improvements Culture [33, 54]. Monitoring and Logging supports software to meet necessary standards and expectations [19, 33] specially on the customer side [55]. Continuous Security Integration of DevSecOps represents a few challenges while adding security capabilities into DevOps pipeline [18, 44]. Efficient Test Automation needs lightweight model-based testing for quick and efficient automation [18, 33]. Learning New Skills: Both developers and system administrators need to acquire new skills for continuous software delivery [12, 18].

1.4.3 DevOps Metrics

There are a few important points discussed in previous literature regarding DevOps metrics or measurements. The main points where there appears to be an agreement is that DevOps metrics are essential to improve the software quality and the delivery process [2, 38, 56, 57]. Also, the community has developed tools and methods for quality analysis and to support DevOps [2, 56, 58]. However, there is a lack of cohesiveness in research themes, as well as a scarcity of detailed empirical evidence to support the key DevOps metrics already being used by engineering teams while improving their adoption of DevOps [17, 54]. Furthermore, despite the increase in interest from the community, there is still lack of systematic discussion and consensus on the metrics and how to implement them in a DevOps cultural way, as seen in Figure 3.6.

As such, this thesis also aims to explore and discuss the concept of metrics in DevOps and how to apply them in an organizational context. DevOps metrics are quite diverse and have various applications throughout the DevOps life cycle as investigated in the MLR of Article 2. Despite the DevOps metrics being diverse, most authors agree on the four DevOps core metrics proposed by the DevOps Research and Assessment (DORA)[1]:

- **Mean Time To Recover/Restore (MTTR)**: The amount of time it takes to get back to normal after a service interruption, whether the disruption was caused by a recent deployment or a single system failure.
- **Mean Lead-time for Changes (MLT)**: This is the time taken to commit and release (deploy) it from the default (trunk) branch up until production.
- **Deployment Frequency (DF)**: The cadence at which new code is deployed to production over some period.
- **Change Failure Rate (CFR)**: The percentage of code changes that require hot fixes or post-production fixes.

In terms of previous work related to DevOps metrics, it is stated by Riungu-Kalliosaari et al. (2016) [24], that DevOps reduces the time it takes to implement system changes in production, affecting MLT and DF. While also doing it with trust and empathy, improving the MTTR and setting a standard for high-quality work while minimizing conflicts, blame and CFR. The book "Accelerate" [59] explores how modern organizations apply DevOps principles and practices, to achieving delivery performance. It uses statistical methods to assess organizational performance of IT organizations, specifically for producing software and delivering it at a high rate. Forsgren et al. (2018) [22] also defined two ways for measuring the performance of DevOps, survey data and system data. Each has its advantages and disadvantages, while multiple authors suggest starting by determining the metrics to measure when evaluating a DevOps implementation in

[1]https://dora.dev/

organizations. There is little literature on the quantification of DevOps capabilities and practices [17, 60].

Authors discuss DevOps measurements in specific contexts. Particularly, Forsgren et al. (2018) [22] investigate the need to measure DevOps implementations at scale in organizations, with a focus on establishing a measurement baseline through system-based measurements, which includes survey-based software development metrics. Farshchi et al. (2018) [61] examined the utilization of metrics in cloud operations, specifically for detecting anomalies during DevOps operations, and propose a metric selection approach based on regression analysis.

In their book, Snyder and Curtis (2017) [58] present a case study on Fannie Mae IT's transition to Agile/DevOps from traditional waterfall methods; software analytics were used to guide improvements and assess progress. It is specifically stated that "project-level analytics allowed agile teams to monitor structural quality and assess their practices". Detailed insights into detecting anomalies in public clouds are provided by Sun et al. (2016) [62], emphasizing the difficulty of distinguishing between these and DevOps activities. On the other hand, Roche (2013) [57] discusses how DevOps capabilities are implemented in the areas of quality assurance and mentions metrics while not specifically advocating for them.

Mishra et al. (2020) [2] mention the need for further research in many areas of DevOps. Such as measurement, development of metrics of different stages to assess their performance, culture, practices toward ensuring quality assurance, and quality factors such as usability, efficiency, software maintainability and portability. While Erich et al. (2017) [33] emphasize that measuring the effect of DevOps on actual benefits accurately is essential. Nevertheless, it can be difficult to track proper quantitative measures of DevOps success and clarify the different definitions of value, which represent another perspective on what constitutes a definition for DevOps. The effectiveness of applying DevOps capabilities and practices should not only be measured subjectively. However, the absence of clear metrics to quantify DevOps implementation effectiveness has been cited as a challenge in multiple research papers.

1.4.4 DevOps Life Cycle Processes

Based on the IEEE Std 2675-2021 [23], Life Cycle Processes (LCPs) can be described as processes that comprise the life cycle of software and systems. These processes are applicable to "software, systems, products, and services" and include "conception, development, production, utilization, support, and retirement". The standard also specifies that these processes are meant to be applied concurrently, iteratively, recursively and incrementally to its elements with the involvement of stakeholders and with the aim of achieving customer satisfaction. The life cycle processes also encompass "the activities necessary to establish an agreement between two organizations" through Acquisition and Supply processes. Additionally, the document clarifies that the order of the processes does not imply the sequence of implementation, as they "can be performed iteratively and concurrently with other processes". The compiled list of LCPs is shown in Figure 6.4.

Other authors cover various aspects closely related to the DevOps LCPs. Akbar et al. (2022) [63] proposed developing security in depth for DevSecOps as part of the engineering process aimed at building intrinsically secure applications. Alnafesaah et al. (2021) [56] observe a lack of thematic consistency in DevOps research, as well as low coverage in surveys of quality engineering within DevOps processes. Waseem et al. (2020) [64], systematically classify research on Microservices Architecture (MSA) related to DevOps, including microservices development and operations in DevOps, process approaches, and tool support for the implementation of MSA-based systems, as well as migrating to MSA in DevOps. Rafi et al. (2020) [38] discuss quality assessment process challenges, particularly from heterogeneous sources, highlighting a few important trends that may be harmful to the data quality assessment process when combined with DevOps technology.

Mishra et al. (2020) [2] perform a Systematic Mapping of the impact of DevOps on the software quality process. Importantly, it is emphasized the need of research in "measurement, development of metrics of different stages to assess performance, culture, practices toward ensuring quality assurance, and quality factors such as usability, efficiency, software maintainability and portability". John et al. (2017) [65] introduce the Service Provider DevOps (SP-DevOps) framework, which addresses issues with service verification, observability, and troubleshooting. The study also examines the entire development process, from design goals to tool implementation, demonstrating how SP-DevOps can enable carrier-grade operations and management in the network virtualization era.

Furthermore, the IEEE DevOps standard mentions that the purpose of DevOps LCPs is to specify required capabilities and practices for operations, development, and other key stakeholders to collaborate and communicate to deploy systems and software in a secure and reliable way [23]. Aiming to provide "a defined set of processes and methods to facilitate DevOps principles and practices, including improved communication between stakeholders throughout the systems life cycle", but this is described while not relating or explaining directly the larger set of capabilities or metrics found in literature and gathered in Article 1 and in Article 2. Despite that, the standard does mention that DevOps LCPs are designed to address the increased rate of change in modern development methodologies and to achieve "end user goals for increased productivity and quality", which reinforces the need to address the research problem of this thesis.

Finally, the LCP vector of this research builds upon the mentioned IEEE standard, drawing themes like the associated Quality Assurance (QA) process in DevOps provides objective evidence that prescribed processes have been followed competently and according to approved plans to meet expected outcomes [23]. This thesis addresses this and other related issues in Article 6.

1.4.5 DevOps Challenges and Benefits

The DevOps adoption together with its capabilities, metrics and process, has multiple benefits for business, such as increased speed to market, product and service quality, customer relevance and satisfaction, productivity, and innovation [44], however, this does not come without challenges. In a study by Jabbari et al. (2018) [26] many challenges for the DevOps adoption were identified which shows the need to prioritize what are the high-impact ones that need to be prioritized. While Leite et al. (2019) [17] critically explore some of the most relevant DevOps challenges reported by the literature.

Previous reviewed authors have largely acknowledged the obstacles faced by organizations in adopting Developer (Dev) and Operations (Ops) (DevOps) capabilities to improve software delivery. These challenges include communication structures that may hinder cross-department collaboration, cultural change required to embrace DevOps practices, and technical barriers, such as different development or production environments [23]. For instance, there are clear problems when operation teams don't align in monitoring metrics or often the developers disagree internally over priorities like server uptime versus release frequency, according to Leite et al. (2019) [17]. The refined aspects of cultural behavior are complex, specifically in big organizations where management is not prepared for a cultural shift, or does not see it as a requirement for a competitive advantage in a digital transformation. On the other hand, the increase in Deployment Frequency (DF) is considered having a benefit, as are test automation practices and greater cross-department collaboration for good communication [12]. In addition to the speed advantages of small releases, DevOps adoption provides a highly powerful and effective means of enhancing the final product by virtue of its more experimental nature.

In the following subsections, a background is provided based on the existing literature within a synthesized list of challenges and benefits. A more detailed Systematic Literature Review (SLR) research is done in Article 4.

Challenges

Technical and cultural: While adopting DevOps, several barriers prevent close collaboration between the development and operations teams due to technical and cultural challenges [33, 66]. Adoption of DevOps requires overcoming technical challenges, such as different development and production environments, as well as cultural ones, which are limiting cooperation between engineering teams [33, 66]. Anandya et al. (2021) [67] mention highly bureaucratic deployment processes, insufficient communication and collaboration issues, and a lack of a clear definition of the strategic direction that is part of DevOps implementation. Miller et al. (2022) [68] investigate the obstacles associated with DevOps in an environment of combat systems at US Navy, where most challenges serialized were non-technical challenging circumstances, but business regulations related restrictions still present significant bureaucratic friction.

Security integration: DevSecOps is the practice of integrating security processes and controls into an organization's DevOps pipeline. This comes with some challenges — protecting software delivery while preserving agility in DevOps [69, 70]. The challenges of DevSecOps include the absence of secure coding patterns, the nonexistence of automated security test tools, and the lack of knowledge of static security tests due to unfamiliarity [63]. Rafi et al. (2020) [38] talk about the problem in quality assessment of heterogeneous environment integrated data as well DevOps security challenges and a prioritization based taxonomy to address them. Al-marsy et al. (2021) [71] proposed a model to assess the issues in applying cloud computing for health information systems, namely financial performance/cost, IT operational efficiency, and security/governance/compliance.

Complexity in automation: The complexity of performance engineering approaches and tools is a major hurdle to the common use of performance analysis in DevOps-driven projects as well [72]. Hemon et al. (2020) [73] identify risk amplification with DevOps, and orchestration of interactions between automation and knowledge sharing, respectively as problems areas that need to be addressed, stating the need for research of any emergent sharing practices for DevOps in large organizations. Waseem et al. (2020) [64] discuss challenges on implementing Microservices Architecture (MSA) in DevOps and provide a list of common pitfalls.

Lack of a clear definition: It is difficult to determine the specific practices that organizations ought to implement because there is no clear definition of what DevOps is [17, 31]. In Luz et al. (2019) [25] the lack of knowledge for successful paths to DevOps adoption has been described as a challenge. Kumara et al. (2021) [74] find challenges in implementation, design and the violation of/adherence to the essential principles of Infrastructure as Code (IaC). Teixeira et al. (2020) [14] highlight the challenge around common understanding of what "Dev-Ops" is, as well as a lack of adoption models or fine-grained maturity models to assist DevOps maturation and implementation.

Benefits

Improved collaboration and trust: DevOps strengthens the relationships between development and operations teams, which in turn makes the workflow much more fluent [4, 12]. Alnafessah et al. (2021) [56] described DevOps as a method to break the silos between developers and operations teams for continuous and fast delivery, and quick responses to changing requirements within the software life cycle. Hemon-Hildgen et al. (2020) [73] argue that DevOps leads to higher job satisfaction compared to Agile only, and mention automation, orchestration and sharing as a means of increasing the levels of job satisfaction and trust.

Enhanced IT performance: DevOps capabilities contribute to enhanced IT performance by reducing waste and optimizing the entire delivery pipeline [5, 75]. Leite et al. (2019) [17] review the literature and correlate the DevOps automation tools with performance, which benefits engineers, managers, and stakeholders. Díaz et al. (2019) [76] describe the benefits when adopting DevOps in terms of software delivery performance and highlights the metrics provided by DORA as indicators for defining a set of software delivery performance industry profiles (elite, high, medium and low performance).

Increased deployment frequency: DevOps enables organizations to increase the frequency of software releases, allowing for rapid delivery of new features and products [21, 59]. Romero et al. (2022) [77] document the implementation of DevOps practices, stating that DevOps "helps organizations to speed up delivery time, improve software quality and collaboration between teams" with practices such as IaC and Continuous Integration & Delivery or Deployment (CI/CD), which also lead to increased product quality.

Better Quality Assurance and security: While DevOps practices improve services development and quality assurance throughput, it also results in better software [78] Rafi et al. (2020) [38] also discuss creating a taxonomy for DevOps security challenges to help practitioners "with securing their DevOps implementations to provide secure better and continuous software. According to Mishra et al. (2020) [2], DevOps focuses on the concept of deployment speed, frequency, and quality as a key strategy to this quality pressure.

Regarding the IEEE Std 2675-2021 [23], it does not explicitly list challenges, but it implies that there is a need for "establishing effective compliance and information technology (IT) controls" as a response to challenges in "build, package, and deploy systems and applications in a reliable and secure way", which are stated has benefits.

In summary, there are several types of benefits resulting from DevOps' adoption, which can be translated into technical, cultural, and business benefits [29]. The adoption of DevOps capabilities automates the development process, resulting in improved service performance, enhanced scalability, and continuous software release and deployment, which may greatly reduce the production cycle time, which was typically considered slow. Microservice architectures are cited as an advantage because they allow faster release cadences to be achieved through component decomposition. To sum-up, the authors recommend that DevOps various challenges concerning communication & cultural change should not discourage organizations from its adoption.

1.4.6 DevOps Outcomes

According to the previous authors, DevOps outcomes can be categorized into tangible and intangible accomplishments that can be achieved through the implementation of DevOps capabilities and cultural values [26, 69]. They aim to improve collaboration between the development and operations teams, improve the speed and quality of Software Development (SD), and guarantee the systems and applications safety and reliability. More precisely, the outcomes can be considered an accelerated frequency of deployment, improved recovery times, and decreased change failure rates, which will lead to better organizational results. Furthermore, the outcomes of DevOps pertain to the effective utilization of CI/CD, and automated testing within the IT infrastructure, with the aim of enhancing efficiency and accountability [38]. Overall, the ultimate goal of these outcomes is the delivery of a product of the necessary quality to users and other stakeholders. As a result, business purposes and technical demands may be addressed over the Software Life Cycle Processes (LCPs) [23]. The DevOps standard defines DevOps results as tangible outcomes of software development and IT operations teams integrating and collaborating [26]. These outcomes are enabled by the set of practices and procedures that simplify the entire software development cycle, from the design to production and implementation. The implementation of DevOps practices in accordance with IEEE Std 2675-2021 [23] is expected to ensure that the outcomes of faster delivery of value to end-users, higher system reliability, and enhanced compliance with legal and other requirements will be achieved. In summary, DevOps outcomes are focused on improving the performance, quality of service, and security of software delivery by bringing development and operations processes together and using automated workflows and communication strategies among the parties.

Other authors offer insights into less systematized outcomes, like **performance oriented culture** where DevOps facilitates earlier and more frequent communication between development and operations, which reduces the time needed to set up development environments [33]. **Low burnout**, since organizations expect that the goal of reducing time-to-market will not be achieved at the expense of product quality or increased staff burnout [19]. This leads to the outcome of **job satisfaction** since DevOps adoption is expected to increase worker's well-being, team effectiveness and customer's satisfaction [2]. Consequently, an increase in speed and effectiveness of problem-solving permits **faster recovery**, due to improved communication between development and operations personnel, which leads to more efficient and effective resolution of problems [33] and better **informed decisions**. Therefore, with fewer problems, **customer satisfaction** itself is stated as another important outcome [17]. **Predictable and faster releases** are another outcome resulting from optimization and automation of processes leading to faster time-to-market and improve software quality and **faster delivery of new products and features**, reducing the lead time from development to deployment [19, 79] and **release reliability**. DevOps strives to continuously improve software testing results by producing **high-quality code**, which means fewer production failures [26] and **better security and compliance** [44]. This thesis proposes organizing capabilities with outcomes in Figure 2.10.

Finally, better communication and collaboration among stakeholders throughout the software life cycle positively affects DevOps outcomes for acquirers, suppliers, integrators, maintainers, and users. DevOps practices allow teams to create a continuous delivery pipeline that automates the application build, package, and deployment. This, in turn, leads to faster and iterative development.

1.4.7 DevOps Culture

The word "Culture" is defined as the way of life, especially the general customs and beliefs, of a particular group of people at a particular time [80]. The literature suggests that DevOps culture is associated with a movement intended to improve business value and at the same time enhancing collaboration between development and operations teams. The DevOps culture aims to break down the conventional silos, which Agile has evidenced [81], and take a next step to foster collaboration among software-producing departments, addressing the issue of slow feature releases and the compromise between speed and quality [2]. DevOps shifts the mindset to say that organizations can achieve both speed and quality without having to sacrifice one of them. This culture, if well adopted, can change the current environment to one that focuses on standardization, automation, and a customer-oriented approach for quick software delivery.

Furthermore, DevOps culture is not limited to tools or associated practices, but instead serves as an indicator of a broader organizational shift toward constant learning and adaptation to market needs. It is characterized by a set of principles that promote a more flexible software development life cycle centered around quality and efficiency [82]. In other words, DevOps culture will encourage the removal of technical and cultural blockers while additionally promoting the following principles:

Collaboration: reduces barriers between development and operations, enhancing software deliveries with a shorter development lifecycle, and providing *Continuous Delivery or Deployment* (CD) at higher software quality [25, 83].

Automation: is an essential DevOps culture pillar for building, deploying, and infrastructure provisioning/management. [82, 83].

Continuous Improvement: promotes Feedback loops and iterative development are important in the culture of DevOps [2, 25, 84].

Knowledge Sharing: enables a collaborative DevOps culture via the exchange of learned know-how and transparency across teams [82, 84].

Quality Assurance: benefits from integrating continuous testing and monitoring, QA happens through the life cycle [2, 31, 83].

Resilience: prompt recovery from failures and sustaining high availability. An autonomous system is highly resilient in a DevOps culture because software systems are designed to not fail easily or at all, while high available [23, 25, 85].

Agility: DevOps culture also promotes agile organizations that are responsive to or even encourage changes in market condition and consumer demands [25, 86, 87].

Customer-Centric Approach: DevOps culture is associated with delivering valuable outcomes directly to customers (end-users) and advocates for solid customer support [23, 62, 83, 88].

Cultural shift: to adopt DevOps culture, the organization needs to move away from the traditional function-based, siloed structures [24, 83].

Measurement and Metrics: DevOps culture supports the use of metrics and continuous measurement to gain insights into progress, performance, and reliability [2, 22, 25, 44].

However, some problems may also arise because emerging capabilities and practices might call for a loosely defined concept, as do possible emerging metrics, which is likely to slow adoption rates and lead to disagreements between team members [19, 33]. A deeper examination of the possible interactions between continuous integration, culture, sharing, automation, and others is done in this thesis to discover and systematize capabilities and metrics.

Finally, the transition to DevOps may also be more complex because it is a collaborative process that is different from the methods that software companies have been using for many years, thus the need for a cultural mindset shift [14]. DevOps is more than simply implementing certain tools or practices. It implies a change in the organizational mindset, promoting a culture of continuous improvement, knowledge sharing, and embracing changes triggered by market demands [4, 89]. The culture is supported by values that facilitate a more responsive and agile software development life cycle, along with standards of quality in performance efficiency [4, 25]. In essence, DevOps culture is about breaking down the silos between software development and IT operations staff, all in order to establish a more closely integrated and automated approach to creating, delivering high-quality software quickly and with flexible reliability [4, 25, 33, 89].

1.5 Research Communication

This thesis's main contributions are related to the research problem that is stated in Section 1.2. Given the aim of this research, the expected contributions are going from a theoretical approach towards gathering empirical evidence to a more practical application of the findings in the last article and in Chapter 8, the conclusion of this thesis. The enablement starts by considering the lack of consensus in the literature regarding DevOps practices, capabilities, and metrics. Subsequently, it encompasses the exploration and systematization of understudied capabilities and metrics by extending the research to the existing gray literature. Which leads to a third relational vector of DevOps life cycle processes from the meanwhile released IEEE Std 2675-2021 [23] as seen in Figure 1.3.

By including original research that could be published, the goal was to ensure that the findings had a steady and significant impact on the field, were peer-reviewed, and published in reputable journals. As it can be observed in Figure 1.3 this was achieved by addressing the research problem during thesis development. The responsibility of publishing and revising the papers is done within the group of researchers, while the PhD candidate was the first author in five of the six papers, the other authors provided essential feedback, mentoring or supervision.

A1) Capabilities and Practices in DevOps: A Multivocal Literature Review.

This first article conducts a MLR on the correlation between DevOps capabilities and practices, aiming to comprehend diverse community perspectives and gain valuable insights towards a more effective mapping of processes. An expanded literature review of 93 documents in the form of books, academic articles, white papers and conference proceedings was used

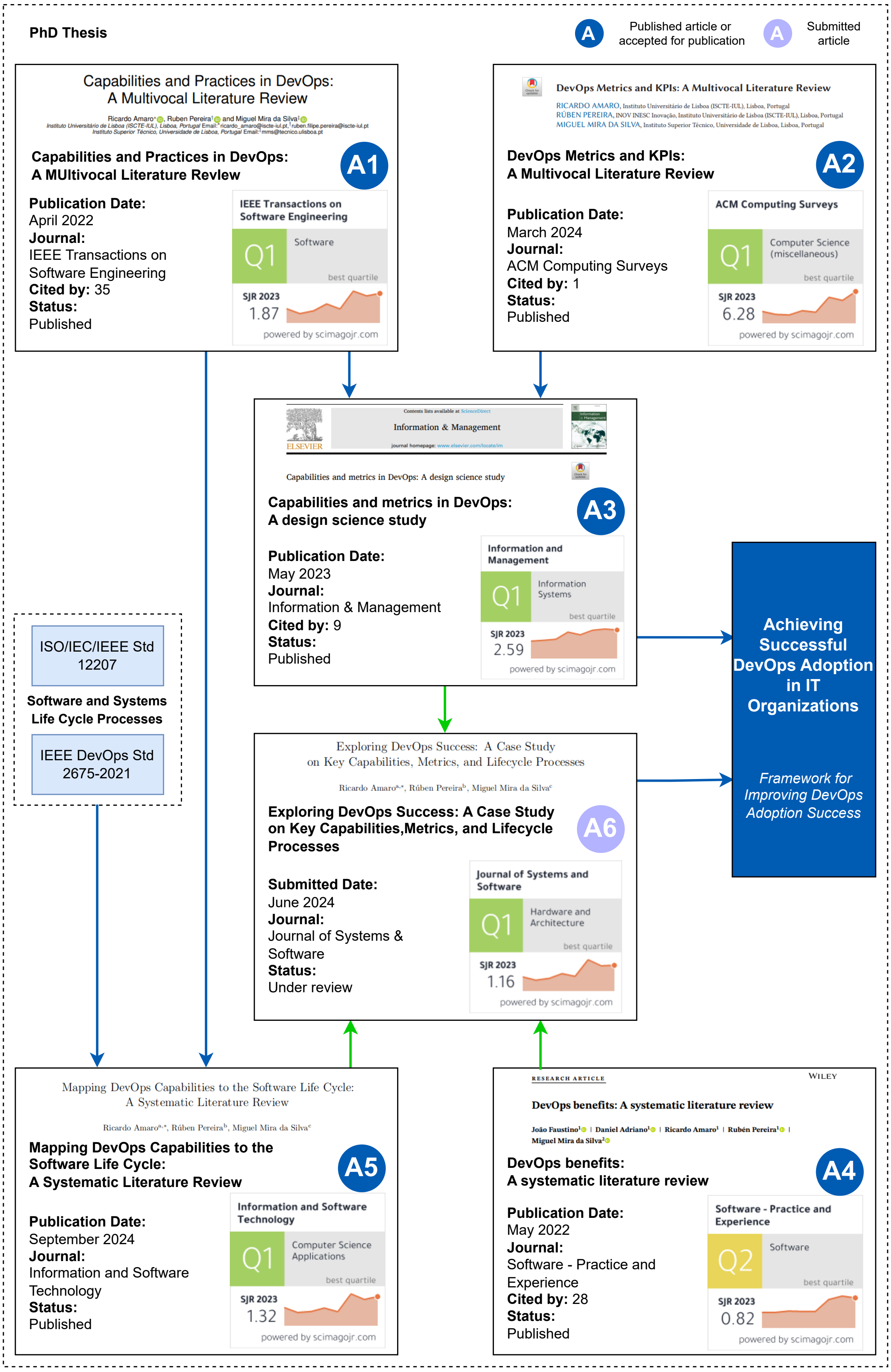

PhD Thesis
A Published article or accepted for publication
A Submitted article
Capabilities and Practices in DevOps: A Multivocal Literature Review
A1
Publication Date: April 2022
Journal: IEEE Transactions on Software Engineering
Cited by: 35
Status: Published
IEEE Transactions on Software Engineering
Q1 Software best quartile
SJR 2023 1.87
powered by scimagojr.com
DevOps Metrics and KPIs: A Multivocal Literature Review
A2
Publication Date: March 2024
Journal: ACM Computing Surveys
Cited by: 1
Status: Published
ACM Computing Surveys
Q1 Computer Science (miscellaneous) best quartile
SJR 2023 6.28
Capabilities and metrics in DevOps: A design science study
A3
Publication Date: May 2023
Journal: Information & Management
Cited by: 9
Status: Published
Information and Management
Q1 Information Systems best quartile
SJR 2023 2.59
Exploring DevOps Success: A Case Study on Key Capabilities, Metrics, and Lifecycle Processes
A6
Submitted Date: June 2024
Journal: Journal of Systems & Software
Status: Under review
Journal of Systems and Software
Q1 Hardware and Architecture best quartile
SJR 2023 1.16
Mapping DevOps Capabilities to the Software Life Cycle: A Systematic Literature Review
A5
Publication Date: September 2024
Journal: Information and Software Technology
Status: Published
Information and Software Technology
Q1 Computer Science Applications best quartile
SJR 2023 1.32
DevOps benefits: A systematic literature review
A4
Publication Date: May 2022
Journal: Software - Practice and Experience
Cited by: 28
Status: Published
Software - Practice and Experience
Q2 Software best quartile
SJR 2023 0.82
ISO/IEC/IEEE Std 12207
Software and Systems Life Cycle Processes
IEEE DevOps Std 2675-2021
Achieving Successful DevOps Adoption in IT Organizations
Framework for Improving DevOps Adoption Success

Figure 1.3: Publications included in this research

for this investigation. The report looked at 37 different DevOps capabilities and grouped them into four categories: cultural, measurement, process, and technical. This article investigates how capabilities emerge and evolve over time, demonstrating the adaptation process in the context of dynamic collaboration among teams. The article explains the relationship between capabilities and practices at various levels, as well as how they should be applied. Therefore, the study indicates that industrial research vastly exceeded scientific research on this topic, which supports a clear difference between capabilities and practices in achieving better DevOps implementation.

This publication aims to assist in the adoption of DevOps. The explicit mapping of capabilities to practices aids organizations in identifying the skills needed, and the processes contained, making the transition from one process to a more refined and effective one easier. Collaboration between teams, visual management to determine maturity levels, clear information about key practices and capabilities, community agreement on important practices and capabilities, and focusing on specific capabilities like cross-team collaboration, continuous integration, continuous delivery, proactive monitoring, and visual management capabilities.

A2) DevOps Metrics and KPIs: A Multivocal Literature Review.

The article presents the task of defining and categorizing the DevOps metrics to simplify the process of DevOps adaptation and improve the performance analysis of DevOps practices in organizations. The MLR included 139 documents, and the output was the identification of 22 main metrics and their definition, importance, in sets of Key Performance Indicators (KPIs). The investigation reveals the need to use precise metrics to determine whether the software delivery is being done well or needs improvement. The study's outcomes assist researchers and practitioners in understanding DevOps metrics and how to implement them effectively. The findings highlight the importance of metrics like Mean Time To Recover/Restore (MTTR), Mean Lead-time for Changes (MLT), Deployment Frequency (DF), and Change Failure Rate (CFR) in driving outcomes and improving DevOps adoption success.

As DevOps adoption helpers, these metrics provide organizations with the tools to measure and improve their DevOps processes, ensuring that they can track progress and make data-driven decisions to enhance their software delivery performance. The publication points out that the success of DevOps adoption in organizations can be attained by using suitable DevOps metrics that are quantifiable, business-relevant, trustworthy, actionable and traceable indicators; categorizing these metrics into Key Performance Indicators (KPIs); ensuring metrics are measurable, relevant, actionable, reliable, and traceable; collecting broader sources, including practitioners' perspectives; and using precise metrics to assess the success of DevOps capabilities adoption.

A3) Capabilities and metrics in DevOps: A design science study.

This is a research article that aims at finding the correlation between DevOps capabilities and metrics, developing an artifact in the form of a matrix for evaluation, which supports the implementation and the adoption of DevOps. The study included the two MLRs and 31 semi-structured interviews with practitioners and experts, resulting in 37 DevOps capabilities and 24 metrics identified. Being that 22 metrics are gathered from the literature and two additional were strongly mentioned in interviews. The publication also shows that for DevOps to succeed, teams will need to be empowered, and the organizational culture will need to change. The key contribution is a newly created outcome-based capability evaluation matrix. This matrix helps understand which capabilities could use long-term gains with the right measurements and converge, as well as which skills should be the main focus of future IT investments. As a result, this finding shows that DevOps adoption requires a comprehensive approach to IT and organizational improvement, which is vital for the delivery of a software performance.

As a DevOps adoption helper, the evaluation matrix provides a structured approach for organizations to assess and improve their DevOps practices, ensuring that they focus on the most impactful capabilities and metrics to drive continuous improvement: Such as empowering teams to make decisions and changes, cross-team collaboration and communication, support learning culture and experimentation, enabling transformational leadership, and shifting to blameless postmortems to reduce fear of failure Thus increasing job satisfaction, team happiness, and talent retention.

A4) DevOps benefits: A systematic literature review.

This work performs two SLRs. The first uses a total of 98 publications to elicit benefits, while the second synthesizes the 19 DevOps benefits found, mapping them into 36 DevOps case studies. The study identifies several key benefits, including improved collaboration and communication between developers and operators, faster time to market, increased code quality, and enhanced automation. Additionally, the study highlighted the challenges that companies face while adopting DevOps, including resistance to change and cultural transformation. Consolidating the benefits and challenges associated with DevOps is the most significant contribution this study makes, providing actual data to support the various conclusions. The research provides organizations who are interested in adopting DevOps with significant insights, highlighting the need of tackling cultural and technical challenges in order to achieve effective adoption.

As DevOps adoption facilitators, the detailed mapping of benefits to specific case studies offers practical examples of how organizations have successfully implemented DevOps and the outcomes they achieved, providing a roadmap for other organizations to follow: Concept-centric approach, willingness to share knowledge, use of technological tools for automation and measurement, process optimization through lean methodologies, improving people/processes/technologies capabilities, providing training to employees, and inclusion of operation team members early in development.

A5) Mapping DevOps Capabilities to the Software Life Cycle: A Systematic Literature Review.

This SLR maps DevOps capabilities to the Life Cycle Processes (LCPs) established in the IEEE 2675-2021 standard [23]. The study investigated the relationships between DevOps capabilities and LCPs, categorizing them and emphasizing the close interaction between technical DevOps capabilities and LCP technical procedures. The main finding of this study is that eight capabilities seem to have a very high impact on the supply process, requirements definition, integration, and validation process. On the other hand, we can see that agreement processes and measurement capabilities, despite having fewer relations, their influence with LCPs seems particularly effective and meaningful based on the results. These findings can be useful for organizations willing to improve the DevOps adoption by aligning capabilities with LCPs in order to enhance the speed, quality, and reliability of the software delivery.

It is seen as DevOps adoption helpers, tools, and education aimed at applying the structured approaches to processes optimization and DevOps capabilities assessment will enable organizations to get better performance and maturity levels in LCPs. This will also ensure that DevOps practitioners will be able to seamlessly adopt the principles into their workflows: Cultural mindset change, collaboration among stakeholders, CI/CD, automation using FLOSS tools — like Chef, Puppet, or Ansible — continuous monitoring and feedback loops, alignment with IEEE standards, a structured approach for optimizing processes, providing appropriate information to managers and teams, fostering a culture of learning and experimentation.

A6) Exploring DevOps Success: A Case Study on Key Capabilities, Metrics, and Life Cycle Processes.

In this case study, the journey of a private software company is examined towards adopting DevOps as a software development approach and discuss the implications of this adoption. The study identifies, in total, 38 symbiotic relations between capabilities and metrics and between metrics of life cycle processes; along with key initiatives, benefits, and challenges observed. It also sheds light on the need for self-service repository management along with other self-services for empowering teams, like an internal catalog to validate maturity and cloud-native infrastructure that enables ease of management. These findings do reinforce the strong need for product prioritization of non-functional requirements, information transparency, automation, and standardization for successful DevOps adoption. The results include important takeaways for companies seeking to adopt DevOps with attention paid largely to business, technical and life cycle process alignment in driving towards capabilities working together as one engine instead of disjointed units.

As DevOps adoption helpers, the detailed analysis of the company's DevOps journey provides insights into the practical challenges and strategies for overcoming them, offering a comprehensive guide for other organizations to follow: Self-service repository management, internal catalog for evaluating maturity, cloud-native and open-source solutions, product prioritization, information visibility, automation, standardization, improved collaboration and communication, CI/CD, automated testing, monitoring, continuous improvement through data gathering and analysis. Furthermore, the case study was presented internally in the organization, where feedback was given and considered. Plans have been discussed to publish a practical version of the case study together with Google DORA in the future.

In conclusion, the six articles emphasize the importance of establishing and classifying DevOps capabilities and metrics with lifecycle processes, as well as dealing with cultural and technical challenges for better DevOps adoption success. The full scope of the research underscores the importance of a holistic strategy for increasing team autonomy, sustaining an environment where it is safe to take risks and learn from failure on a consistent basis. Furthermore, using concrete operational performance improvement metrics to achieve better software delivery performance. These observations should give good direction for organizations who are building new DevOps teams and also improving their software and systems life cycle processes.

1.6 Thesis Organization

The presented document follows the format of an article-based thesis. This allowed the authors to have more time to write high-quality scientific papers.

The Introduction is done in **Chapter 1**, where the motivation, research problem, the objectives and deliverables of this research are explained and at the same time In **Section 1.4 a detailed theoretical background** of the key concepts is given about DevOps, DevOps Adoption, Capabilities, Metrics, Life Cycle Processes, Challenges and Benefits, Outcomes and Culture. The remainder of the document is structured as follows.

From **Chapter 2** to **Chapter 7** the six articles are introduced and included, retaining the same format that was submitted, in order to comply with the required formatting of the journals. Despite them appearing as separate work, their respective figures, tables, and major sections appear and are linked via the lists and table of contents. This was done using the LaTeX package pdfpages[2]. The articles appear in the same order as numbered in Figure 1.3.

Lastly, **Chapter 8** contains the conclusion of the research with summary and discussion where the key concepts and the framework for improving DevOps adoption success is presented and discussed as another important output of this thesis. After this, limitations are listed and some possible upcoming future work is suggested.

[2]https://ctan.org/pkg/pdfpages

CHAPTER 2

Article #1

This article (A1) starts the research that motivated and led to this thesis [90]. Since its inception, it has been produced, strongly improved, and published during the development of this thesis. Therefore, it is included as the basis for gathering community consensus on the relationship between DevOps capabilities and practices. The study conducts a Multivocal Literature Review (MLR) to identify, categorize, and analyze the main DevOps capabilities and practices, their definitions, relationships, and how they contribute to better DevOps adoption.

As a result, it contributes and categorizes 37 capabilities, their mentions in literature, and their definitions. The concepts of Practices and Capabilities were mapped in ordered taxonomy. The outcomes are designed to help researchers and practitioners understand how capabilities and practices are connected at different levels. The research framework of analysis was used to analyze data in a structured way and connect capabilities and practices findings. The relation of several capabilities and practices was analyzed and discussed for generating outcomes and results.

Article details:

- Title: Capabilities and Practices in DevOps: A Multivocal Literature Review
- Date: April 2022
- Journal: IEEE Transactions on Software Engineering
- Scimago Journal Rank: Q1
- Publisher: Institute of Electrical and Electronics Engineers Inc.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

Capabilities and Practices in DevOps: A Multivocal Literature Review

Ricardo Amaro*, Ruben Pereira† and Miguel Mira da Silva‡
Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal Email:*ricardo_amaro@iscte-iul.pt,†ruben.filipe.pereira@iscte-iul.pt
Instituto Superior Técnico, Universidade de Lisboa, Portugal Email:‡mms@tecnico.ulisboa.pt

Abstract—Context: To meet the demands of customers and market, IT organizations are seeking to implement DevOps. While many succeed in DevOps adoption, others lack the knowledge on how to incorporate DevOps culture, process, measurements, and techniques in their business. Thus, successful adoption is still inconsistent, highlighting the need to provide management with relevant information to support the development of DevOps Capabilities effectively. But what are these Capabilities? Unfortunately, there is still a lack of clarity about DevOps Capabilities and their relationships to DevOps Practices and Outcomes among researchers and practitioners.
Objective: This research aims to gather community consensus on the relationship between Capabilities and Practices, so a better DevOps implementation can be mapped. Seeking to define DevOps Capabilities and Practices concepts and to identify, organize and summarize Capabilities as they relate to Practices.
Method: A MLR is conducted, with 93 documents gathered and thoroughly examined from throughout the community, including books, scientific articles, white papers, and conferences, among others.
Results: This survey contributes a list of 37 organized Capabilities, their mentions in literature, and their definitions. The concepts of Practices and Capabilities were mapped and categorized in an ordered taxonomy.
It is concluded that industry research has much outweighed scientific research on this topic, with Capabilities evolving dynamically over time, reinforcing *team collaboration and communication* as the most crucial one. The study's Outcomes will assist researchers and practitioners understand how Capabilities and Practices are related at different levels and how to better implement them.

Index Terms—DevOps Capabilities, DevOps Practices, Software Engineering Process, Software release management and delivery, Software Development, Multivocal Literature Review

1 Introduction

Today, IT organizations are increasingly challenged with ever-changing customer requirements, competition, regulatory environments and sophisticated outside threats [1]. Therefore, the need to establish a competitive advantage by doing things faster and better than competitors [2], like delivering and supporting software, with reliability and in a predictable form has become increasingly important [3]. However, this need for frequent software delivery, without sustained builds, proper testing and release automation, generates burnout and pain in the engineers doing operations, decaying software delivery performance [4] and reliability [5].

In consequence, during the last decade, DevOps has become a rising mindset [6] in the Software Engineering (SE) industry. DevOps organizational approach stresses empathy and encourages greater collaboration between engineering teams involved in the software delivery [7], in order to provide better end user experience [8], reduce development time, enhance deployment rates, increase stability [9], optimize Mean Time to Recover and reduce cost of deployment and implementation [10].

On the other hand, in the cases that leadership supports this transformation, it still needs to have a clear vision [11] of the steps to take ahead based on information in order to increase efficiency [2] and ensure the success of applying DevOps Capabilities. This implementation is a complex process [12] demanding control via a rigorous systematization of self-assessment [13], which in turn should result in growth of the maturity levels that will lead to improved performance [14]. A good way to control and assess these levels of maturity and adoption would be by implementing visual management [15] communicating, in a way that takes little or no prior training to interpret, for instance, if a given software delivery process within the pipeline [16], is at optimal levels or could be improved.

As a result, it would be beneficial to visually comprehend the Practices that comprise DevOps as a process, as well as the teams skills included in Capabilities required to carry out these Practices as shown in Figure 1.

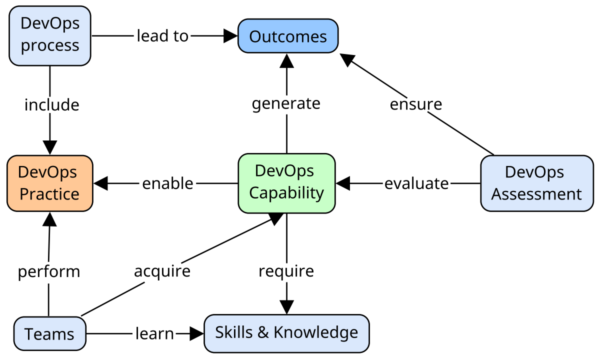


Figure 1. DevOps Capabilities and Practices Conceptual Map.

In this conceptual map, to be treated as the **research framework of analysis**, based on literature findings, the

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

team learns skills [17] that are required by DevOps Capabilities and related Practices, generating DevOps Outcomes and results [18]. Following an *impact mapping* [19] where we should work backwards from the Outcome [9]. This might be beneficial for practitioners, since it will later improve a DevOps assessment [1] that is done to ensure Outcomes are met and evaluate [20] each DevOps Capability. The framework is instantiated and analyzed in Section 6.5.

In this model, if a team wanted to enhance a specific Capability's Outcome, such as doing *Continuous Integration* (CI) well, they might look at which related team skills have a positive influence on the Outcome and seek to improve the team's knowledge needed for these Practices [21] as part of the DevOps Process. If the team does CI properly, they may be able to deploy more regularly. This *Continuous Delivery or Deployment* (CD) is likewise a desirable Outcome, and it is done through other Practices. Thus, the meaning of Capability varies according to context. Outcomes are the expected results researched by Diaz et al. [18] like faster time-to-market, better software of research process productivity, and team effectiveness & satisfaction. In essence, a framework supported by a good distinction between Capabilities and Practices focused on Outcome results to pursue a better DevOps implementation.

Providing teams and managers a clear picture of the key Practices, the Capabilities required of individuals and teams to do these Practices well, and the desired Outcomes would contribute to better knowledge of how to apply DevOps Capabilities iteratively to various organizational situations. But what are these Capabilities and their relation to Practices? Unfortunately, relationships between DevOps Capabilities and Practices, leading to Outcomes, are still unclear within the DevOps community, with dispersed publications having different or incomplete approaches to clarify the problem.

Therefore, the *research problem* is identified as the following: There is a lack of clarity about DevOps Capabilities and their relationships to DevOps Practices among the DevOps communities of researchers and practitioners, as there seems to be scattered knowledge about their definitions and relations in dispersed publications.

This article proposes to understand and synthesize the main DevOps Capabilities that are mentioned in publications and how they relate to DevOps Practices. Based in Figure 1 do a broad literature review in order to instantiate this theory, which is found in the formal literature, and try to deepen some concepts of this theory by expanding the research to texts from practitioners in the industry. It is in this connection that an investigation will be conducted using a literature review to determine what Capabilities exist and how they are related to Practices. The objective of the research is to survey a community consensus on which DevOps Practices and Capabilities are most significant, as well as their relationships, enablement, and Outcomes.

What Practices and Capabilities are important in DevOps, and what are their relationships, enablement and Outcomes? Since the DevOps topic has been much more explored and analyzed from the industry side than from the scientific side, as observed in several previous related MLR works [22], [23], [24], [25], [26], with major technology companies releasing regular reports [10], [27], [28], [29], [30], [31], [32] and regular conferences [33], [34], notwithstanding Systematic Literature Reviews (SLRs) and Surveys [20], [35], [36], [37], in this paper a Multivocal Literature Review [38] is conducted to identify the main DevOps Capabilities and Practices, their definitions and relationships in order to map out the Capabilities mentioned by DevOps practitioners, researchers and how they relate Capabilities to Practices. Therefore, including the viewpoint of practitioners and contrasting academic and practitioner disparities in place, with the goal of aligning them, hypothesizing, and drawing attention to these discrepancies.

The primary objective of this study, which is to investigate Capabilities and Practices, can be translated into the following research questions:

- **RQ1**. What are the main Capabilities needed to implement DevOps?
- **RQ2**. How frequently are Practices or Capabilities mentioned in gray literature compared to academic literature?
- **RQ3**. Are authors aligned on the relation between DevOps Capabilities and Practices?

2 DevOps

DevOps is an acronym for the Developer (Dev) and Operations teams (Ops). These engineering teams work collaboratively to eliminate so-called "information silos" [39], based on a cultural mindset change early proposed by Debois et al. [40], in order to achieve higher delivery, quality and cooperation [41] via human collaboration across departments and automation [36]. A standard definition for DevOps is proposed by Olszewska et al. in the IEEE Standard 2675-2021 [42] where it is mentioned a set of principles and Practices that promote greater communication and cooperation among key stakeholders for the goals of defining, creating, and operating systems and software products or services, as well as continual improvement in all elements of that entity's life cycle.

From the industry side, blog posts on the topic are common, but they mostly distinguish on a concrete concept of the term. Jabbari et al. [43] proposed that DevOps is defined as a development approach aiming at bridging the gap between development and operations by stressing communication and cooperation, continuous integration, quality assurance, and delivery with automated deployment through the use of a set of development processes [44]. It can also be seen as a conceptual framework that is based on the Capabilities, mentioned in Section 6.2. Humble et al. proposed it to be focused on the acronym CAMS (Culture, Automation, Measurement and Sharing) [45] and later added to these four pillars, the Lean (L) pillar, becoming the acronym CALMS [46].

Ebert et al. [48] in his paper about DevOps, emphasizes the culture shift toward collaboration between development, quality assurance, and operations. This new approach to software delivery that occurs through the three ways, which are the principles underpinning DevOps, illustrated in Figure 2, as opposed to the traditional approach that is separated in organizational silos. This characteristic of good cooperation between IT Development and IT Operation teams is crucial in order to ensure successful deployment and operations of IT systems [49].

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

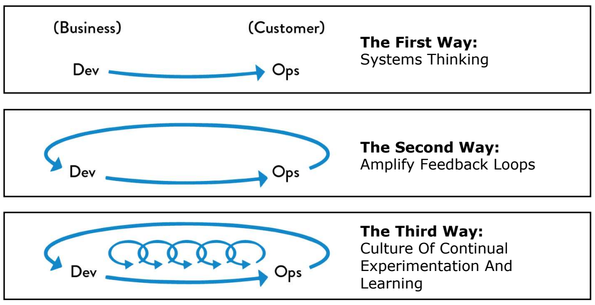


Figure 2. The Three Ways: The principles underpinning DevOps (adapted) [47].

DevOps is then a culture, movement or Practice emphasized on collaboration and communication, focused on improving the software release cycle speed to production and build automation of new software components, while keeping high quality, mentioned in Lwakatare et al. [17], where a literature review on the term DevOps concludes that DevOps is a change of mindset substantiated with a set of automated Practices to encourage cross-functional team collaboration.

Lastly and according to Riungu-Kalliosaari et al. [50], DevOps is a set of Practices aimed to reduce the time that a change made to a system takes to go into normal production, while ensuring high quality and the least friction and blame between teams as opposed to trust and empathy.

3 Multivocal Literature Review

A Multivocal Literature Review (MLR) [38] is a type of Systematic Literature Review (SLR) [51], which aims to incorporate gray literature like blogs, videos, web-pages and white papers, which are constantly produced by SE practitioners outside academic forums, notwithstanding the published (peer-reviewed) writing like journal articles and conference papers. Therefore, MLRs are important to the expansion of the research by including literature that normally would not be taken due to its “gray” nature [52], as show in Figure 3.

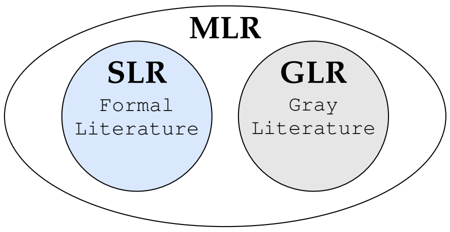


Figure 3. The relationship of SLR, GLR and MLR Studies [38].

While considering conducting a Literature Review from formal literature in the specific topic of DevOps, a few researchers already realized that “broadening” the scope and including Gray Literature (GL) would add value and benefits to the review study. Some examples of successful DevOps research, in the same area, using MLR already exist [22], [23], [24], thus corroborating the practical usefulness of this method for the proposed research, expanding the diversity of sources that are available in a variety of forms, reflecting different purposes and perspectives [53]. The objective to be pursued with this MLR research is to map out the DevOps Capabilities and Practices, mentioned by both DevOps practitioners and researchers, and how they relate Practices to Capabilities. Therefore, it is fundamental to survey what practitioners outside the academia are also saying on this matter.

Given the need to expand this study outside the boundaries of scientific knowledge and therefore MLR gives us that opportunity, while still maintaining a rigorous qualitative analysis procedure [53] for reviewing that literature. The separation of several types of literature is seen in Table 1, where is listed ’White’ and ‘Gray’ literature sources into 1st tier, with high credibility, and 2nd tier with moderate credibility. For DevOps, it is preferable to include 2nd tier, given that there is valuable expertise and knowledge on those sources. However, it is also necessary to exclude literature that corresponds to ideas, concepts and thoughts, like tweets, social networks or emails from the 3rd tier.

Table 1
Spectrum of the ’white’, ‘gray’ and excluded literature (adapted) [38].

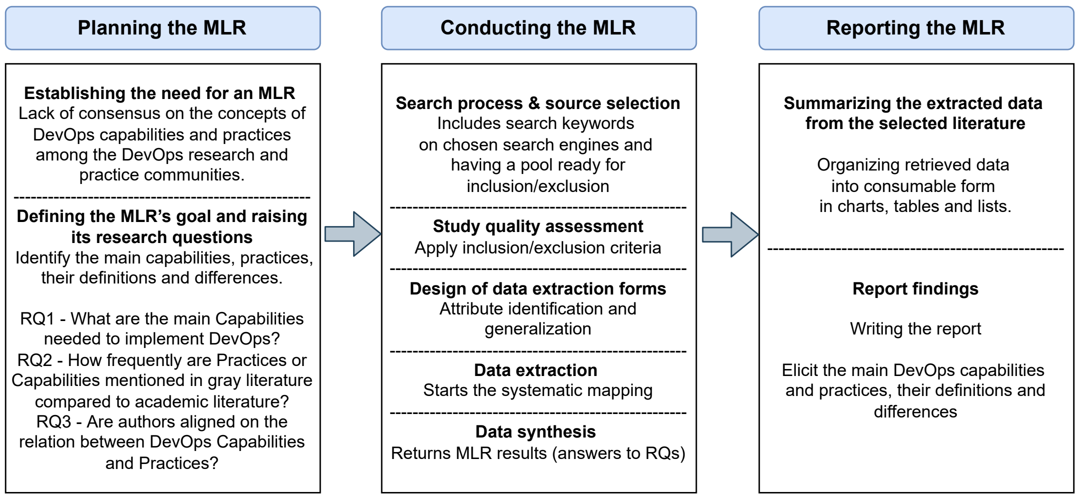
’White’ literature	’Gray’ literature	Excluded literature
Published journal papers	Preprints	Ideas
Conference proceedings	e-Prints	Concepts
Books	Technical reports	Thoughts
	Lectures	
	Data sets	
	Audio-Video (AV) media	
	Blogs	

Multiple guidelines exist in the literature to conduct SLR studies in SE. However, several phases of MLRs differ from those of traditional SLRs. In particular, the process of researching and assessing the quality of the source. Therefore, SLR guidelines are only partially useful for conducting MLR studies as seen in Figure 4. This process shows the planning, conducting and reporting as proposed by Garousi et al. [38].

In following this process, it is expected that the gray literature will return substantial knowledge in certain areas of this DevOps research, but of course, the inclusion of such literature will bring certain challenges as the evidence provided is often based on experience and opinion. For that reason, for this research process systematic guidelines will be used for performing MLR in software engineering (SE) [52], to approach a structured search, similarly to SLR, collecting the materials by applying the inclusion and exclusion criteria in the search results obtained from well known search engines like Google search, Google Scholar and others.

1) The MLR **planning phase** consists of the following two phases.
   - Establishing the need for an MLR in a given topic.
   - Defining the MLR’s goal and raising its research questions.
2) Once the MLR is planned, we proceed to **conducting the review** in five phases.
   - **Search process** for formal or GL is typically done via means of using defined search strings.
   - **Source selection** normally includes determining the selection criteria and performing the selection process.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022



Planning the MLR
Establishing the need for an MLR
Lack of consensus on the concepts of DevOps capabilities and practices among the DevOps research and practice communities.
Defining the MLR's goal and raising its research questions
Identify the main capabilities, practices, their definitions and differences.
RQ1 - What are the main Capabilities needed to implement DevOps?
RQ2 - How frequently are Practices or Capabilities mentioned in gray literature compared to academic literature?
RQ3 - Are authors aligned on the relation between DevOps Capabilities and Practices?
Conducting the MLR
Search process & source selection
Includes search keywords on chosen search engines and having a pool ready for inclusion/exclusion
Study quality assessment
Apply inclusion/exclusion criteria
Design of data extraction forms
Attribute identification and generalization
Data extraction
Starts the systematic mapping
Data synthesis
Returns MLR results (answers to RQs)
Reporting the MLR
Summarizing the extracted data from the selected literature
Organizing retrieved data into consumable form in charts, tables and lists.
Report findings
Writing the report
Elicit the main DevOps capabilities and practices, their definitions and differences

Figure 4. Multivocal Literature Review (MLR) steps adopted in this research [38].

- **Study quality assessment of sources** in order to determine the extent to which a source is valid and free of bias.
- **Data extraction** design forms, procedures and logistics, with possibility of automated data extraction and synthesis.
- **Data synthesis** with chosen qualitative and quantitative techniques.

3) Finally, **reporting the review** is the last phase.
   - The reporting phase of an MLR is similar to the SLR guidelines of Kitchenham and Charters [54], summarizing the extracted data from the selected literature and report findings.

4 Research Design and Implementation

This section corresponds to the first phase of the mentioned MLR process. It begins by explaining the motivation for this work, followed by the objectives and the corresponding research question that is intended to be answered throughout the research. After that, a review protocol is presented.

4.1 Motivation

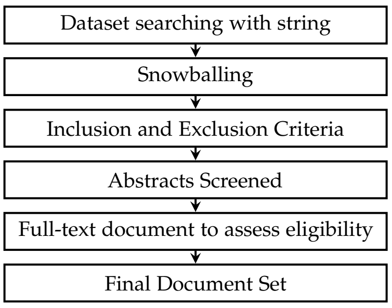
In software development organizations, that want to implement DevOps internally, management needs to have relevant supporting information about this technological transformation, in order to assess the success and increase efficiency [2] of applying Capabilities [55], [56]. However, the concept and relationship of Capabilities and Practices is still not well-defined within DevOps practitioners community.

As this topic has been further explored and analyzed from the industry side than from the scientific side, with leading technology companies regularly publishing reports [57], a Multivocal Literature Review expands the diversity of sources to identify the main Capabilities and Practices.

To provide order and clarity to the meanings and relationships of DevOps Practices and required Capabilities, an expanded variety of sources must be gathered. With the addition of gray literature to review, a comprehensive survey can be conducted on not only what the scientific literature specifies about Capabilities and Practices, but also what the industry generates dynamically and uses internally. Combining both points of view will enrich the research questions that are proposed in Section 1.

4.2 Review Protocol

The complete review protocol is illustrated in Figure 5.



Dataset searching with string
Snowballing
Inclusion and Exclusion Criteria
Abstracts Screened
Full-text document to assess eligibility
Final Document Set

Figure 5. Review protocol performed in this research.

In order to find other studies related to this work, that may achieve answers to the proposed research questions, a search was conducted in April 2021 using various keywords. The search string used to perform the search in order to retrieve the maximum number of studies and the chosen datasets are listed in this section.

- **Search String**: `(devops AND (Practices OR Capabilities))`.
- **Datasets**: The search engines used were, Google search[1], Scopus[2], Web of Science[3], IEEE[4], ACM[5] and EBSCO[6].

The first set of papers is obtained. In a first phase, after the search is complete and snowballing is done, inclusion

1. https://www.google.com
2. https://www.scopus.com
3. https://apps.webofknowledge.com
4. https://ieeexplore.ieee.org
5. https://dl.acm.org
6. https://search.ebscohost.com

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

and exclusion criteria will be applied for refining the search results.

To facilitate searching and collecting high volumes of gray literature, some code was developed as seen in the source code in Appendix **??** (**??**) in order to parse the data into CSV files [58]. This way we can ensure obtaining clean results that are not specific for the user, but general, this solving the problem of consistency in the returned results because Google returns customized results that are tailored differently for different users based on their previous search history and preferences. Lastly, it facilitates the work of fetching the results into spreadsheet files that are easily consumable in the MLR process.

The inclusion and exclusion criteria for this MLR is shown in Table 2. After that step, the abstracts must be screened in order to evaluate the relevance they have to the research. Finally, the relevant papers are read in order to obtain the final selection of studies to perform the review.

Table 2
Inclusion and exclusion criteria applied in this research.

Inclusion Criteria	Exclusion Criteria
Written in English	Unidentified author
Published in and after 2013	No publication date
Full-text accessible	Advertisement or Job Post
Mentions DevOps Capabilities or Practices	

5 Conducting the MLR

In this section, it is described how the review is conducted, which is the second phase of the SLR. At this moment, the search is performed using the search query over the selected databases and an analysis is carried on top of the extracted data.

5.1 Selection of Studies

For reference, the complete summary of the review process is shown in the diagram in Figure 6 with a visual representation of the applied MLR selection process. This reflects all the selection work done through the methodical process of MLR.

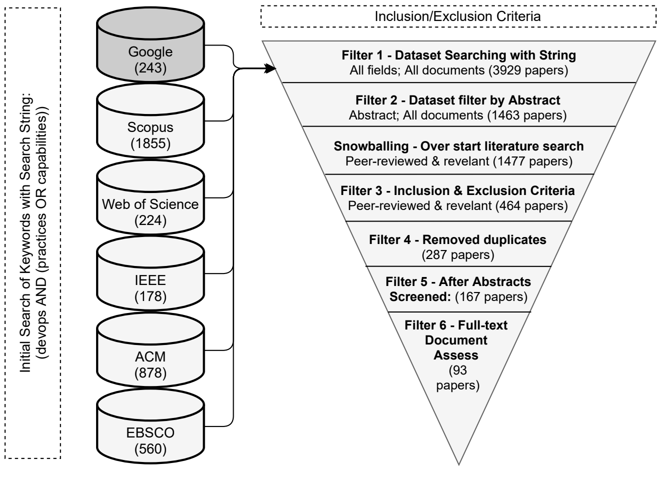

Initial Search of Keywords with Search String: (devops AND (practices OR capabilities))
Google (243)
Scopus (1855)
Web of Science (224)
IEEE (178)
ACM (878)
EBSCO (560)
Inclusion/Exclusion Criteria
Filter 1 - Dataset Searching with String
All fields; All documents (3929 papers)
Filter 2 - Dataset filter by Abstract
Abstract; All documents (1463 papers)
Snowballing - Over start literature search
Peer-reviewed & revelant (1477 papers)
Filter 3 - Inclusion & Exclusion Criteria
Peer-reviewed & revelant (464 papers)
Filter 4 - Removed duplicates
(287 papers)
Filter 5 - After Abstracts Screened: (167 papers)
Filter 6 - Full-text Document Assess (93 papers)

Figure 6. Followed Multivocal Literature Review process (adapted) [38].

In the initial search step *filter 1* (All fields; All documents) was used together with the search string, both present in Table 3. This is shown in Table 3, as part of the MLR protocol to find the final set of article, which gives us a relation of the articles found in conjunction with the filters used.

The discrepancy from *filter 1* to *filter 2* is justified by the fact that initially the keywords could be found anywhere within the returned item and some search engines return more literature than just academic papers, like newspapers or reports. While in the case of Google search engine, this does not apply. Thus, the results remaining the same.

On a second pass, *filter 2* (Abstracts; All documents) was used over the existing search results, therefore reducing the number of documents that have an abstract mentioning the keywords, narrowing down to a total of 1463 publications.

Table 3
Filters used in the MLR protocol.

Database	Filter 1	Filter 2	Snowballing	Filter 3	Filter 4	Filter 5	Filter 6
Google	243	243	+14	89	89	77	75
Scopus	1855	342		157	42	40	2
Web Of Science	224	174		91	29	24	1
IEEE	178	146		67	67	14	8
ACM	878	92		22	22	6	4
EBSCO	560	475		38	38	6	3
Total	3929	1463	1477	464	287	167	93

Legend: Filter 1 = Query All fields, All documents; Filter 2 = Query Abstracts, All documents; Snowballing = Applied over starting literature search [38] Filter 3 = Relevant (inclusion/exclusion criteria); Filter 4 = Remove duplicates; Filter 5 = After Abstracts Screened; Filter 6 = Full-text Document Assess;

In the next phase a *snowballing* [59] is conducted leading to extra 14 relevant publications found, which increased the total number of papers.

Applying inclusion & exclusion criteria *filter 3*, present in Table 2, 464 publications remain. This leads to *filter 4*, which is defined to remove the duplicates from the list of results in order to obtain the set of documents to have abstracts screened. For the cases belonging to gray literature, there is no abstract. Therefore, all text was skimmed, making it possible to better assert an inclusion or exclusion of that publication. In the end, after all abstracts are screened, 93 publications remain for full-text document assess.

5.2 Data Extraction Analysis

After selecting the final set of publications, an analysis of the different components of the results is presented here, in a relationship of the final set of documents based on the source data. This analysis arises from the evaluation of the full text of the 93 publications eligible for extraction of any relevant information for this research. An overview is also given of which years and categories of publications were selected for full reading.

5.2.1 Gray and white literature number of contributions

The relation of final document set by database reflected in Figure 7 show that 75 results came from Google search

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

with 80,65% of gray literature. For the cases of Scopus, Web Of Science, IEEE, ACM and EBSCO they all contributed with the total sum of 18 (19,35%) of relevant research documents.

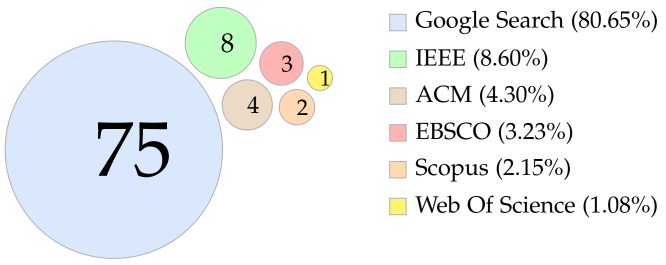


Figure 7. Distribution of the final set of documents per database.

5.2.2 *Distribution of publications over the years*

Important to note the distribution and growth of the selected papers shown over the years in relation to the publication seen in Figure 8. This shows a growing interest in the last three years with an increase in volume of research work related to researching Capabilities, confirming the potential interest and usefulness of this research might have in the area. The first publication from this set was Puppet Labs' 2013 State of DevOps Report [27], that was followed by the next 2014 State of DevOps Report [28] and a webpage on "Six Core Capabilities of a DevOps Practice" from the New Stack website [60]. In 2015, 2016 and 2017, there is a continued growth in webpages and conference papers in 2017.

The interest in gray literature has risen significantly in 2018, coinciding with the release of the book Accelerate [9], as evidenced by the enormous increase of webpages in that year, indicating that practitioner publications have evolved far quicker than scientific research. Despite a brief reduction in publications in 2019, practitioners' publications still continued to grow in 2020 as we can see in Figure 8. It becomes important to note that the values for 2021 are lower because the search that created the database for this paper occurred in March 2021, and therefore it is an incomplete year.

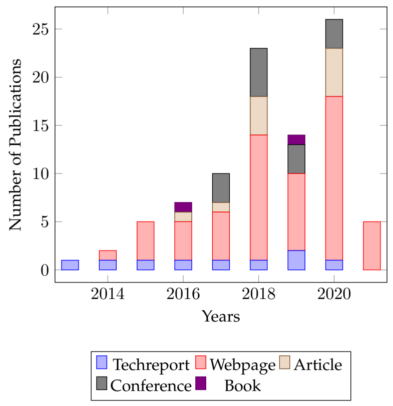


Figure 8. Distribution of publications per type over the years.

6 Reporting the MLR

At this step of the MLR, the DevOps Capabilities and Practices are discussed and reported, and all three research questions are assessed in light of the publications obtained using the research protocol. This leads to improved definitions of DevOps Capabilities and DevOps Practices in Section 6.5, together with a DevOps Capabilities and Practices instantiation of the framework presented in Figure 1. Based on the achieved definitions given in Table 10 and Table 11, this section will focus on using the term "Capability", since that is better aligned with the findings and more applicable for the research in hand.

6.1 Results Analysis

For the systematic coding review a set of base studies were selected that already elicit different Capabilities based on the work of Smeds et al. [55] and Senapathi et al. [61], the scientific book Accelerate [9] and the DevOps Research and Assessment (DORA) [56]. Since most of the Capabilities names have a match, a list was extracted and then used on the full-text analysis of the papers. From this point on, a compilation of Capabilities and Practices (when also mentioned as Capabilities) is gathered, with the attention to only retain the relevant ones that are mentioned and explained in the texts and keep the list manageable.

The publications found were also analyzed with the protocol seen in Table 9 in mind. The first base study dates back to 2015, when Smeds et al. [55], proposed a set of Capabilities that were later referenced and augmented in another research conducted by Senapathi et al. [61] in 2018. In this paper, the author discusses and establishes what the DevOps technology enablers and engineering Capabilities are, here presented in Table 4.

Table 4
List of DevOps Capabilities proposed by Senapathi et al. [61].

1) Collaborative and continuous development
2) Continuous integration and testing
3) Continuous release and deployment
4) Continuous infrastructure monitoring and optimization
5) Continuous user behavior monitoring and feedback
6) Service failure recovery without delay
7) Continuous Measurement

Interestingly, from all the final publications gathered in this research, only two of the scientific literature papers refer to the work of Senapathi et al. [61]. The mentioned papers are "A maturity model for DevOps" [62] and "Managing quality assurance challenges of DevOps through analytics" [63], therefore evidencing the low impact of the first definition proposals had in the long term.

It is noted in literature that these Capabilities bring several types of advantages which can be grouped or categorized into technical, cultural and business benefits, Kim et al. [64]. The technical Outcomes are mostly in the Practices of Continuous Integration, Delivery and Reliability [5]. The cultural benefits reside in the improvement of communication and the creation of stronger feedback and collaboration between different teams [20], [28], [32], [44],

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

[61], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76] and, in result, to improve employee motivation, which usually contributes to achieving the organization goals. Finally, the business benefits are on the customer side and the faster delivery of value, and at the same time ensuring greater business stability and increased innovation [77]. Evolving from the various State of DevOps reports research done over the years [8], [57], 24 Key Capabilities have been published in the book Accelerate [9] in 2018, where the five categories were initially proposed can be seen in Table 5.

Table 5
List of Capabilities in five categories proposed by Accelerate [9], [78].

Continuous Delivery	**Architecture**	**Product and Process**	**Lean Management and Monitoring**	**Culture**
Version control	Loosely coupled architecture	Customer feedback	Change approval processes	Westrum organizational culture
Deployment automation	Empowered teams	Value stream	Monitoring	Supporting learning
Continuous integration		Working in small batches	Proactive notification	Collaboration among teams
Test automation		Team experimentation	WIP limits	Job satisfaction
Test data management			Visualizing work	Transformational leadership
Shift left on security				
Continuous delivery (CD)				

More recently the DevOps Research and Assessment (DORA) [56] team has redefined a set of Capabilities seen in Table 6, that correlate to Accelerate [9], but expands and refines them based on a several year research and data from practitioners worldwide. The correlation can be immediately noticed in the way continuous delivery and architecture categories are merged into a single technical category of Capabilities.

The research here presented also identifies how an important Capability such as **collaboration among teams** was dropped from Table 6 cultural category, when compared to Table 5, despite the fact being the most mentioned from all Capabilities, 81 times out 93 publications seen in **??**. As a result, this Capability was kept and assumed relevant.

Many are high-level Outcome Capabilities, while others are Capabilities in the sense of being able to do a Practice effectively. But in general, it is noticed a growth in the number of technical and measurement Capabilities, such as database change management, monitoring systems to inform business decisions, cloud infrastructure and code maintainability.

Regarding the frequency characteristics of identified Capabilities, it is important to reflect on the growth year-by-year seen in **??** and observe in Table 7 that there is a sudden growth in almost all of them in 2018, coinciding with the release of the book Accelerate [9], from which several practitioner publications have been inspired and evolved [78], [79], [80].

The following Table 7 is a summary of **??** where 2021 is excluded as an incomplete year and the remaining years are grouped together to give better visibility of the Capabilities' growth over the years.

Table 6
List of four Capability categories proposed by DORA [15], [56].

Technical	**Process**	**Measurement**	**Cultural**
Version control	Team experimentation	Monitoring systems to inform business decisions	Westrum organizational culture
Trunk-based development	Streamlining change approval	Monitoring and observability	Learning culture
Continuous integration	Customer feedback	Proactive failure notification	Job satisfaction
Deployment automation	Visibility of work in the value stream	Work in process limits	Transformational leadership
Continuous testing	Working in small batches	Visual management Capabilities	
Continuous delivery			
Architecture			
Empowering teams to choose tools			
Test data management			
Shifting left on security			
Database change management			
Cloud infrastructure			
Code maintainability			

Important to note that since the values gathered in 2021 are only up to March they are residual, and could give a wrong perception of the growth of the several Capabilities. Therefore, they are omitted to not affect the reading.

This gives an overview of how the various Capabilities have grown steadily in the literature over the years, and observe the big leap in 2029-2020, namely in cross team collaboration, continuous integration, continuous delivery, monitoring and test automation.

The most recent rising stars seem to be C20-Shift left on security, C30-Customer focus/feedback and C37-Visual management Capabilities, which have seen a relative jump in 2019-2020. Interestingly, C37-Visual management Capabilities in DevOps have been getting more traction in the latest years, which can be clearly supported by the interest that organizational management has shown in recent years, which has also been observed in the literature.

Finally, from the relation of these and other Capabilities over the years, it is concluded that practitioners are championing these principles, Practices and tools to minimize waste [69], [81], enabling faster feedback cycles [15], [82], exposing invisible technical debt [64], improving value in delivery, maintenance and operational functions [69].

6.2 RQ1 - What are the main Capabilities needed to implement DevOps?

In this section, the MLR provides an answer to the first research question, by revealing a list of 37 Capabilities, categorizing them in table 8 and discussing them in detail.

For the coding process of recognizing these categories into the discovered Capabilities, earlier work from Table 4, Table 5, and Table 6 was first considered. This largely highlighted what categories were already specified by authors [9], [56], [61] into a category list with the majority of the Capabilities

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

Table 7
List of Capabilities identified by number of publications over the years.

	2013-14	2015-16	2017-18	2019-20
C01 - Cross team collaboration and communication	3	9	30	35
C02 - Continuous Integration	2	11	28	35
C03 - Continuous Delivery/Deployment automation	1	10	29	34
C04 - Proactive Monitor, Observability and autoscale	3	10	25	33
C05 - Test Automation and environments	3	7	25	25
C06 - Continuously improve processes/workflows	2	8	18	18
C07 - Version Control System	2	7	14	21
C08 - Support learning culture and experimentation	2	4	15	21
C09 - Empower teams to make decisions/changes	2	4	11	24
C10 - Focus on people, process and technology	1	4	11	15
C11 - Configuration Management	2	5	11	12
C12 - Cloud infrastructure and cloud native	0	4	11	15
C13 - Artifacts versioning and registry	2	5	9	10
C14 - Loosely coupled architecture	0	4	8	14
C15 - Database change management	1	3	7	13
C16 - Infrastructure as Code	0	5	8	11
C17 - Emergency response	0	3	7	14
C18 - Containerization	0	5	9	10
C19 - Open source software adoption	0	4	10	7
C20 - Shift left on security	0	1	5	14
C21 - Transformational leadership	1	3	6	10
C22 - Trunk based development	1	4	5	9
C23 - Monitor systems, inform business decisions	0	4	6	9
C24 - Performance/Westrum organizational culture	1	4	3	11
C25 - Working in small batches	0	2	3	11
C26 - Centralized log management	1	3	3	8
C27 - Lightweight change approval	0	1	3	11
C28 - Visibility of work in the value stream	0	2	4	8
C29 - Working in progress limits	0	2	3	9
C30 - Customer focus/feedback	0	0	3	10
C31 - Blameless Postmortems	1	3	4	4
C32 - Data-driven approach for improvements	0	3	5	4
C33 - Job satisfaction	1	2	1	7
C34 - Test data management	0	1	2	7
C35 - Chaos Engineering	0	4	2	3
C36 - Code maintainability	0	2	2	5
C37 - Visual management Capabilities	0	1	0	8

Legend: This table is a summary of **??** for simplifying the analysis of the Capabilities' growth over the years, excluding 2021's incomplete year results.

Table 8
Categorization of DevOps Capabilities.

Category	ID	DevOps Capability
Cultural	C01	Cross team collaboration and communication
Cultural	C08	Support learning culture and experimentation
Cultural	C19	Open source software adoption
Cultural	C21	Transformational leadership
Cultural	C24	Performance/Westrum organizational culture
Cultural	C31	Blameless Postmortems/reduced fear of failure
Cultural	C33	Job satisfaction
Measurement	C04	Proactive Monitoring, Observability and autoscaling
Measurement	C17	Emergency response/proactive failure notification
Measurement	C23	Monitor systems to inform business decisions
Measurement	C29	Working in progress limits
Measurement	C37	Visual management Capabilities
Process	C06	Continuous Improvement of processes/workflows
Process	C10	Focus on people, process and technology
Process	C25	Working in small batches
Process	C27	Lightweight change approval
Process	C28	Visibility of work in the value stream
Process	C30	Customer focus/feedback
Process	C32	Data-driven approach for improvements
Technical	C02	Continuous Integration
Technical	C03	Continuous Delivery/Deployment automation
Technical	C05	Test Automation and environments
Technical	C07	Version Control System
Technical	C09	Empower teams to make decisions/changes
Technical	C11	Configuration Management
Technical	C12	Cloud infrastructure and cloud native
Technical	C13	Artifacts versioning and registry
Technical	C14	Loosely coupled architecture
Technical	C15	Database change management
Technical	C16	Infrastructure as Code
Technical	C18	Containerization
Technical	C20	Shift left on security
Technical	C22	Trunk based development
Technical	C26	Centralized log management
Technical	C34	Test data management
Technical	C35	Chaos Engineering
Technical	C36	Code maintainability

already mapped. For example, the more generalized "Continuous Measurement" evolved to "Lean Management and Monitoring" and eventually to just "Measurement" which was selected for its clear scope and state of the art. Following this study, the mentioned list was cross-referenced with extracts from all publications into a Capability code with relevant categories leading to table 8.

The Capabilities list is based on the literature review of the 93 publications, also considering the fact that some practitioners mention them as Practices and others as Capabilities as shown in Figure 9. The list includes their name, the number of publications in which they have been mentioned, and, most importantly, their definitions associated with a label of (N) Needed Capabilities for DevOps Practices [9], [56], or (E) Enablers of good practice [61], [72], [83] as shown in Figure 11. This is meant to indicate their relation to the framework of analysis presented in Figure 1 and later discussed in Section 6.5.

C01-Cross team collaboration and communication, mentioned in 81 publications (N). In order to enable cross-functional collaboration between application teams, operations and security teams [20], [30], [61], [63], [72], [73], [80], [81], [84], [85], [86], [87] the organization has to identify the stakeholders, including customers [88], of every project so that they join, have insights about various project phases and processes, and start making valuable contributions. As highlighted by Kim et al. [64], this also signifies an opposition to a culture of fear, with firm embracing DevOps aiming for a high-trust, collaborative culture where individuals are rewarded for taking chances.

C02-Continuous integration (CI), mentioned in 80 publications (N). Continuous integration takes tasks like testing and building, and automates them [15], [32], driving teams to produce high-quality software, to reduce the cost of ongoing software development and maintenance [67], [74], [78], [82], [85], [89], [90], [91], and to increase the productivity of the teams. [61], [82]. The CI process creates canonical builds and packages that are ultimately deployed and released [9]. Continuous integration (CI) and continuous delivery (CD) are the corner stone of software delivery, denoting the huge importance of a Continuous Integration & Continuous

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

Delivery or Deployment (CI/CD) pipeline [61], [63], [69], [72], [76], [92], [93], [94]. Continuous integration is the Capability of multiple developers to commit and merge their code [95], [96].

C03-Continuous delivery and deployment automation, mentioned in 78 publications (N). While continuous delivery automates the entire software release process with a manual step, continuous deployment automates that step, deploying smaller changes [35] to production as soon as they are released from acceptance testing [69], [97], without manual intervention [9], releasing faster and more frequent, reducing the risk of production deployments and providing faster feedback to the teams [56], [81]. Continuous delivery entails deploying code updates to production as frequently as feasible.

C04-Proactive monitoring, observability and autoscaling, mentioned in 74 publications (N). It is critical to monitor the infrastructure [35], [44], [64], [69], [98], whether it is in the cloud or in a local data center. Combining proactive monitoring with autoscaling can automatically solve capacity issues [28], [73], [96], [99] and reduces the need to scale the system manually.

C05-Test automation and environments, mentioned in 62 publications (E). Getting quick feedback on the impact of changes across the Software Development Life Cycle (SDLC) is fundamental to integrate quality into software [27], [64]. It is important to have automated and correctly provisioned test environments along the pipeline [61], [79], [96], [100], reducing long lead times [8], [31], [32], [97], [101].

C06-Continuous improvement of processes and workflows, mentioned in 46 publications (N). Continuous improvement is enabled through a combination of continuous integration, deployment, testing, workflows and monitoring [63], like, implementing branch-naming consistency, where all work originates from the same source while developing on a branch referencing a ticket [65], or applying consistent patterns across multiple applications [31].

C07-Version control system, mentioned in 45 publications (E). Version control and automation are tightly intertwined [15], [78], [95], [102] enabling efficiency and productivity [27], [37], [70]. Version control extends versioning to all production artifacts [64], such as application code, configurations, system settings, and scripts for automating build and environment setup [8], [9].

C08-Support learning culture and experimentation, mentioned in 44 publications (E). Organizations that develop a learning culture [103], [104], [105] and comprehend its impact on organizational performance encourage engineers to have the ability to work alone and experiment [78], [79] to test business concepts and new ideas, to write and update requirements during development [31], [101].

C09-Trust/empower teams to make decisions and changes, mentioned in 42 publications (E). Trust is essential in every relationship, but it is especially critical for DevOps [65] to improve software delivery performance and job satisfaction, empowering them with the ability to make educated decisions about the tools and technologies they employ [28], [44], [80], [90], [101], [103]. This helps to create greater results [61], [96], [106].

C10-Focus on people, process and technology, mentioned in 32 publications (E). People, process and technology are the three pillars of a software development project. There must be a feeling of community, sharing a common goal, and contributing to the common cause [90]. Improving the culture is an ongoing journey [75], [107]. DevOps unifies people, processes, and technology: when all three are aligned toward the same business goals, innovation can be implemented more quickly [31], [62], [94], [108].

C11-Configuration management, mentioned in 30 publications (N). It is practiced in one form or another as part of any software engineering project. A Software Configuration Management (SCM) is a system for managing and controlling the evolution of software products [109], [110] over the life cycle, automating the configuration, monitoring, managing, and maintenance of all entities of infrastructure and systems like servers, applications, storage, networks, and all managed services [89].

C12-Cloud infrastructure and cloud native, mentioned in 30 publications (E). The US National Institute of Standards and Technologies (NIST) defines five essential characteristics of cloud computing [111]: On demand self-service, broad network access, resource pooling, rapid elasticity and measured service [96], [112]. Each service may be deployed individually [113], with flexibility, tool sets, and scalability for applications. Serverless architectures on clouds can dramatically reduce DevOps effort [95]. In a pipeline—for example for worker nodes, deploying artifacts to test or even production environments [92].

C13-Artifacts versioning and registry, mentioned in 28 publications (E). Enable organizations to centrally store artifacts and build dependencies as part of the software delivery process [15], [80], [114]. It is important to version these artifacts in a repository manager [115], [116], [117], either they are promoted containers along the pipeline, bundles, charts, packages or any other kind to make the changes visible, reliable and repeatable [7], [117] for all production artifacts [10], [30], [105].

C14-Loosely coupled architecture/ microservices, mentioned in 27 publications (E). Microservices are an architecture design for building a single application with smaller services that run independently, usually communicating via Application Program Interface (API) calls [118]. Improves agility and helps organizations to easily grow their product at a cheaper cost and in a shorter time [86]. Each service may be deployed separately and decentralized. Produced and delivered using automated tools and automated procedures [81]. This may be achieved using cluster technology like Kubernetes, Apache Mesos or Docker Swarm.

C15-Database change management/ release alignment, mentioned in 25 publications (E). Database change management [11], [15] and release alignment change management processes [67], [119], [120], when well implemented, help developers and IT professionals to easily manage database updates, system configurations, deploy new code quickly and fix incidents faster. The Outcome is release reliability.

C16-Infrastructure as code, mentioned in 25 publications (E). With infrastructure as code [35], [106], [121], [122], [123], it is possible to express procedures in code [124] rather than setup infrastructure or software manually. Using technologies like Ansible, Chef, Puppet, Salt, or Terraform [125], [126]. That way it is possible to use version control to keep track of all infrastructure modifications in a repeatable and more

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

efficient manner [127].

C17-Emergency response/ proactive failure notification, mentioned in 24 publications (N). Proactive failure notification [15] focus on actionable notifications based on the values being monitored and that have known failure thresholds, instead of a reactive system to alert when it has already failed [20], [61], improving emergency response efficiency [128] and reducing risk of customer impact [44].

C18-Containerization, mentioned in 24 publications (E). Containers are efficient for app development and hosting [95]. They allow DevOps, developers, and system administrators to swiftly, securely, and effectively test, build, deploy, and manage applications [69]. This Capability has become a new standard in DevOps pipelines, clusters, and applications [37], [118] by using technologies like Docker, containerd, CRI-O or rktlet.

C19-Open source software adoption, mentioned in 22 publications (E). Open source adoption correlates with DevOps success [32], [76], [96], [129], and the knowledge of open source solutions for testing and deployment is a must for a DevOps engineer [95]. This model is well represented in the DevOps tool set [70], [130] with impact in early DevOps emergence [103]. Organizations assemble and contribute open source parts, which has become a reliant software supply chain [64], [101].

C20-Shift left on security, mentioned in 20 publications (E). Integrating security into the design and testing phases of the software development process is key to driving IT performance with the Outcome of better security & compliance. Including security reviews of applications, including the infosec team [9], [78], [79]. Shift left on security is related to Developer(Dev), Security(Sec) and Operations(Ops) (DevSecOps) [80] concept and emphasizes automating as much as possible security policies in order to accelerate processes, decrease human error and aiding in quality improvement [131] and audits [26], [31].

C21-Transformational leadership, mentioned in 20 publications (E). It is a style in which leaders inspire and encourage teams to attain better levels of performance [30], [81]. Transformational leaders focus on the growth and performance of their followers and organization [8], [64]. Effective leaders [31] impact software delivery performance by pushing the use of technical and product management Capabilities [77].

C22-Trunk based development, mentioned in 19 publications (E). In trunk based development, each developer works in small batches, merges that work into trunk at least once (and potentially several times) a day [7], [120], consistent with commonly accepted continuous integration Practices [16], [17], [30].

C23-Monitor systems to inform business decisions, mentioned in 19 publications (E). With the Outcome of informed business decisions using visual dashboards [31], [61], [98], [102], [132] allows the organization to track configuration changes made to servers along with databases and deployments [62] that have taken place, along with various metrics, logs, and graphs [133], [134] to give a holistic view of changes happening in the system.

C24-Performance/Westrum organizational culture, mentioned in 19 publications (E). Ron Westrum developed a typology of organizational cultures that includes three types of organizations [135]. *Pathological organizations* are characterized by low cooperation across groups and a culture of blame. *Bureaucratic cultures* are preoccupied with rules and positions, and responsibilities are compartmentalized by department. *Generative organizations* are performance oriented [79], with good information flow, high cooperation and trust, bridging between teams [29], [64], which should be the Outcome.

C25-Working in small batches, mentioned in 16 publications (E). The concept comes from Lean manufacturing and is also used in Agile [81]. Working in small batches with a lightweight approval process helps ensure work can get through the system quickly [98] with shorter lead times [15]. It enables fast flow through the development pipeline, fixing errors as they are discovered vs. at the end [136] also allowing to deliver of MVPs, features, and bug fixes sooner, which also helps enable the customer feedback loop above [80].

C26-Centralized log management, mentioned in 15 publications (E). Centralized logs for applications with multiple servers facilitate debugging [96], [137] when sent to a common service that enables easy centralization, rotation, and deletion [64], [87], [138].

C27-Lightweight/streamlining change approval, mentioned in 15 publications (E). Replace heavyweight change-approval systems with peer review [77]. Lead times, with the Outcome of predictability and increased release frequency, improving considerably [30] with negligible impact on system stability [28], [77].

C28-Visibility of work in the value stream, mentioned in 14 publications (E). Is a Lean manufacturing or Lean enterprise approach for documenting, analyzing, and improving the flow of information or materials needed to create a product or service for a client [81]. Understand and visualize the flow of work [30], [32] from idea to customer Outcome in order to drive higher performance. Make the value flow visible for everyone to understand where their piece fits into the whole flow [77], [98] from the business all the way through to customers [29], [78].

C29-Work in progress (WIP) limits or Work in process limits, mentioned in 14 publications (E). WIP is a type of inventory in Lean manufacturing — the materials that are presently being worked on. As a result, it is called "waste" since it ties up value. Prioritize work, limit the number of things that people are working on [1], [8], and focus on getting a few high-priority tasks done [77]. These limits [98] are identified and enforced. Flow is defined [80]. Overload is limited [81].

C30-Customer focus/feedback, mentioned in 13 publications (N). Drive to better organizational Outcomes of customer satisfaction and new products/features by gathering customer feedback [29], [139] and incorporating it into roadmaps and product design [98] for improving customer focus.

C31-Blameless postmortems/reduced fear of failure, mentioned in 12 publications (E). By removing blame, fear is reduced, and by reducing fear, teams are empowered to surface and solve problems more efficiently. Mistakes occur. Holding blameless postmortems [10], [28], [64], [96] is an effective technique of learning from mistakes [77], [140].

C32-Data-driven approach for improvements, mentioned in 12 publications (E). Analyzing factual data [123] can help an organization achieve performance. Sharing application graphs [137], [141], usage patterns with team members

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

to get everyone aligned. Include scalability, testing, and deployment to simplify the entire process [87].

C33-Job satisfaction, mentioned in 11 publications (E). Job satisfaction is the top predictor of organizational performance [28], [84]. DevOps adoption also help avert burnout [8], a common reason why technical people leave jobs [142]. Beecham et al. [143] identifies several motivators for engineers that influence job satisfaction. The higher the team motivation, the more likely DevOps will succeed. A project manager with strong communication skills, a project where risks were reviewed and controlled, a client who trusted the PM and team, a good working environment, good teamwork, and having a nice time working on a project with low burnout as Outcomes are just a few examples [144].

C34-Test data management, mentioned in 10 publications (E). Managing test data can be challenging [98]. Define the right strategies for managing test data effectively along with approaches to provide fast, secure data access for testing [131] like adequate data to run a test suite, acquiring data on demand, conditioning and limiting the amount of test data needed in the pipeline [9].

C35-Chaos engineering, mentioned in 9 publications (E). Contemporary and distributed software programs must be capable of dealing with unexpectedly tumultuous environments [64], [86], [92], [98], [108]. As a result, such systems must be built from the start to withstand unanticipated problems and shortcomings in production contexts [145].

C36-Code maintainability, mentioned in 9 publications (E). Code maintainability [15], [101] is essential for making it simple for developers to identify, reuse, and alter code, as well as keep dependencies up to date [146] leading to high-quality code Outcome.

C37-Visual management Capabilities, mentioned in 9 publications (E). Improves a company's capacity to communicate progress and expectations toward goals, manage change and improvements [14] visually, in a way that takes little or no prior training to interpret. Visualizing work and pulling it through the system [12] is a critical part of the First Way of DevOps [64].

The list of the most studied and approached Capabilities was collected as it was proposed. In extent, it is seen that the Capabilities are, in fact, dynamic and have been changing over the years.

6.3 RQ2 - How frequently are Practices or Capabilities mentioned in gray literature compared to academic literature?

The second research question requires determining how frequently are Practices or Capabilities mentioned in gray literature compared to academic literature. This provides a clearer understanding of the correlations between gray literature and academic literature concerning DevOps Practices over Capabilities, since gray material is more practitioner oriented while academic writing is seen from an academic viewpoint. The goal is to increase academic and practitioner awareness of these relationships in order to reach a consensus or a balanced view and improve the usage of Practices and Capabilities within the proposed framework.

The two terms are described across several types of formal and gray literature, as seen in Figure 9.

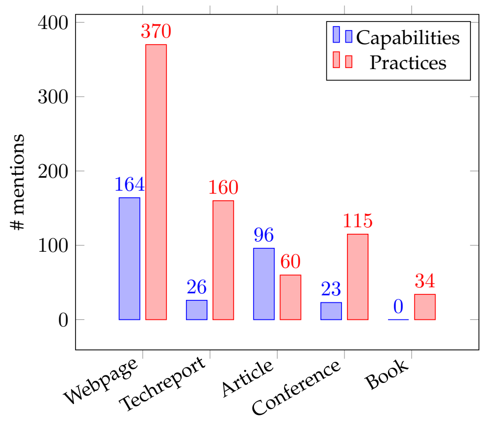


Figure 9. Number of publications mentioning Capabilities and Practices among sources.

Based on the extended research enabled by this MLR, it is noticed that Capabilities have been mentioned interchangeably as Practices in 66 publications Table 9 and eight publications even distinguish Practices from Capabilities [8], [20], [61], [62], [67], [72], [107], [147].

It can be observed in Figure 9, that the webpages, overwhelmingly created by practitioners, have the most mentions as Practices, but also include about half of the number of mentions as Capabilities. In Techreport, Conference and Book, which are closer to the gray literature, we clearly see a tendency to the term “Practice”.

On the other hand, the scientific articles make more mentions as Capabilities, which becomes a very interesting finding of this research, as it reveals that practitioners are more focused on DevOps Practices, while the scientific community tries to organize Capabilities in a way that abstracts more generic concepts applicable to building skills and enablers. Nevertheless, the concepts are interconnected at different stages of the process, as shown in Figure 1.

A different example of this same interchangeability can be observed on a seminal book, “Continuous Delivery” [7] from 2010, that starts on mentioning the term Capabilities. Despite not explicitly mentioning the word “DevOps” it describes, however, in detail the deployment pipeline pattern, which is usually central to DevOps Capabilities. In page 109 of the book, the Capability of deployment and production release is described in detail, explaining how the process is automated, with speed, repeatability and reliability in mind. Jez Humble mentions that when the “Capability” of automating the process as normal events is available, releases are essentially without risk.

Some other books also talk about Capabilities and are frequently cited in gray literature by researchers and practitioners [67], [78], [80], [96], [103], [106], [137], [169] like “The Phoenix Project” [12], “Accelerate: The Science of Lean Software and DevOps” [9], “Lean enterprise: adopting continuous delivery, DevOps, and Lean startup at scale” [77], and “The DevOps Handbook” [64].

In Table 9 it is mentioned that one publication indicates a DevOps Practice definition [62] to be a subset implementation of a Capability. Six publications indicate various Capabilities definitions like a higher level categorization of

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

Table 9
Six publication properties identified from the MLR.

Property	Publications	Total
Interchangeably mentions Capabilities and Practices	[10], [27], [28], [29], [30], [31], [32], [37], [44], [56], [57], [63], [64], [65], [66], [69], [70], [71], [75], [76], [81], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [100], [102], [103], [104], [105], [108], [112], [124], [132], [140], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [160], [161], [162], [163], [164], [165], [166], [167], [168]	66
Mentions Capabilities directly	[8], [11], [20], [32], [35], [56], [61], [62], [67], [72], [78], [79], [80], [91], [101], [107], [129], [147], [169]	19
Presents different or reorganized Capabilities compared to Senapathi et al. [61]	[11], [20], [35], [56], [62], [67], [78], [79], [80], [91], [101], [107], [129], [147]	14
Relates Capabilities to Practices or distinguish them	[8], [20], [61], [62], [67], [72], [107], [147]	8
Indicates a definition for Capability	[8], [20], [30], [35], [61], [62]	6
Indicates a definition for Practice	[62]	1

Practices [35], are important to enhance software company's profitability, productivity and market share [8], Capabilities are the core DevOps aspect which comprises Capabilities such as "continuous planning, collaborative and continuous deployment, continuous integration and testing, continuous release and deployment, continuous infrastructure monitoring and optimization, continuous user behavior monitoring and feedback and service failure recovery without delays" [20].

The previously mentioned research done by Senapathi et al. [61] and 2017 State of DevOps report [30] defines them as a combination set that can change over time, which includes categories of Capabilities like "continuous delivery as the combination of the Capabilities" to be "deployment automation and automated testing, continuous integration and trunk-based development, and version control for all production artifacts". These set of Capabilities have been changing over the years.

It is also an important finding that both concepts and their relations carry out an important role driving implementation and adoption. Therefore, their interaction is later discussed in a framework of analysis on Section 6.5.

6.4 RQ3 - Are authors aligned on the relation between DevOps Capabilities and Practices?

Here, we pursue answering if authors have the same alignment while relating DevOps Capabilities to Practices. To better understand the relations and reach a consensus where a framework can be broadly accepted, we must first clarify how authors see each concept in order to approach the analysis of Figure 1 and its instantiation in Figure 10 with a more strengthened reasoning. It is largely observed that the word "Capability" is used when the observation is external or at a high-level overview. It is then a matter of perspective or context. When talking about a Capability, we see a third-party assessment of something that is being looked at from the outside, while observing a group to see what they are capable of doing. Whereas, a "Practice" is seen from the standpoint of the internal team or group, realizing "I am doing these things". That ability converted to an action is then mentioned with the term "Practice". Therefore, authors will speak about Capabilities from an evaluation standpoint, and Practices from a hands-on approach perspective. The Capability definition points to an organization's "ability" to perform or achieve a certain process, whereas a Practice is referred to more at the level of DevOps practitioners and thus more observed in the gray literature publications, as discussed in Section 6.5. Clear examples of this more formal research concept were presented earlier by Smeds et al. [55], Senapathi et al. [61] and more recently in the book Accelerate [9], in DORA [15], [56], [101] and in several journal articles or proceedings [8], [20], [20], [35], [44], [62], [72], [81].

Organizations should target developing skills, knowledge, and habits in their people [170] as an enabler for continuous improvement. The number of publications mentioning Capabilities or Practices is organized in **??** of Appendix **??**. Notably, there are some publications that mention it more as a Capability like "Trunk based development", "Lightweight change approval", "Working in progress limits", "Customer focus/feedback", "Test data management" and "Visual management Capabilities", while the rest is more mentioned as Practice and this confirms once more the lack of consensus.

It is now clear that there is still some confusion while interchanging these two words as seen in Figure 9 and the table in **??** of Appendix **??**, hence the proposed conceptual framework in Figure 1 tries to clarify the relationship of concepts to help practitioners focus on the DevOps "how-to" implementation sustained by a high-level view. The usage of Capabilities or Practices is not consensual. Therefore, this study proposes achieving some consensus, while mapping the concepts related and proposing clearer definitions of DevOps Capability and another for DevOps Practice, while proposing a framework of analysis in the following Section 6.5.

6.5 DevOps Capabilities Synthesis

Here the most important findings and contributions of this research are summarized, given the purpose of this research of a clearer conceptualization and organization of DevOps Capabilities and Practices, as well as clarifying the definitions of the terms and finally to recenter Figure 1 as a framework of analysis based on the instantiation and relation of these concepts in Figure 10.

6.5.1 Definitions of DevOps Capability and Practice

The definition of DevOps Capability is proposed in Table 10 and the definition of DevOps Capability is suggested in Table 11.

As already mentioned in previous sections, since the misusage of these labels could negatively impact researchers and practitioners in their work, there is benefit of clarifying these definitions, before analyzing their relationships, while addressing DevOps adoption, implementation and research around these topics.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

Table 10
Definition of DevOps Capability.

A **DevOps Capability**, is here defined as the ability to do something [171] in DevOps or by the quality, or state of being capable [172]. It consists of the combined skills [81] accumulated and developed by its members over time. As an example, DevOps technological Capabilities are the information and skills - technical, managerial and institutional - that enable productive enterprises [77], [170] to utilize equipment and technology efficiently [173].

By this definition, "Capability" is the potential or ability to perform something. A person's Capability to use a Practice is determined by their knowledge and skills. An individual, team, organization, or other entity may improve their Capabilities. The use of DevOps Practice or Capability as labels is contextual, which differs depending on the perspective, as can be seen from the definition of DevOps Practice defined in Table 11.

Table 11
Definition of DevOps Practice.

A **DevOps Practice**, is here defined as the action taken rather than thought or ideas [174] or by the act of carrying out [175] a DevOps activity. It can also be see as an enabler [44], [55], [61] of a mentioned Capability by an individual or a group such as the engineering team. As a result, authors will discuss Capabilities from the perspective of evaluation, like an assessment, but refer to Practices from the perspective of a hands-on approach.

In SE, a "Practice" is a technique or methodology for achieving desired goals. It generally includes steps, tools, and a set of concepts and depending on the circumstances, it may be done properly, poorly or regularly. A process is a grouping of Practices. A Practice or group of Practices' desired Outcomes are usually quantifiable deliverables, like new products and features, high-quality code, predictable faster releases or high customer satisfaction.

These proposed definitions can now be used to support explaining the following conceptual relationship between a Practice and a Capability in DevOps.

6.5.2 *Instantiation of concepts*

The Capabilities and Practices of DevOps are the focus of this article. The proposed instantiation of the conceptual map seen in Figure 1 purposes relating in a clear way the roles of both concepts. Following a thorough literature study to cross-check Capabilities and define each of these concepts, the MLR findings are matched to the concept of Capabilities and Practices. The connection between these ideas is instantiated in Figure 10, while also mentioning the relation to Outcomes or results.

The Practices originate from the research done in Section 6.2 where the extracted text already suggest the Practices implicitly used in each Capability. Literature was used to elicit the *Enablers of good Practice* [61], [72], [83], *Outcomes* [18], [30], [83] and *Needed Capabilities for Practices* [9], [56], also contextualized within the same research extraction. It should be noted that while there is a comprehensive list of the main Capabilities, there are probably other Practices that were not perceived as part of this MLR, and likewise there are more inter Practice relations that could be drawn. However, the purpose is to explore how the found concepts relate.

DevOps Capabilities are categorized into Cultural, Measurement, Process and Technical groups and are enabling DevOps Practices. This associates "Practice" with a portion of a process and DevOps "Capability" as a part of skill and knowledge of an individual or team. The team acquires skills required by DevOps Capabilities, resulting in DevOps Outcomes and results, from which we work backwards in order to improve the DevOps assessment and targeted Capabilities.

6.5.3 *Framework of analysis*

This **research framework of analysis**, is based on the MLR findings, reflected in the research questions and leading to a mapping of relations shown in Figure 10. Here it is analyzed how the relation of the several Capabilities and Practices are required by DevOps for generating Outcomes and results.

It is now important to understand the researched Capabilities related to Practices that constitute DevOps as a process, Outcomes, and the related skills needed to enable them. In Figure 10 team Capabilities are mapped to Practices and team Capabilities created as Outcomes (or results) of the DevOps Process. This will be useful for practitioners to improve a particular team Outcome Capability, like *predictable faster releases*, since they could use (C27) *lightweight change approval* Capability to influence this Outcome and try to improve other team's Capabilities in these Practices.

While providing a way for implementers of DevOps to establish a plan for improving the Capabilities by specifying the Practice associated, it will be easier to detect the missing skills that will lead to the expected Outcomes. These Outcomes may be described at a higher level, for instance from the organizational viewpoint like *performance oriented*, or a more concrete level like *release reliability* using database updates and automated system configuration.

Taking a more concrete visualization as reflected in Figure 11 there is a clear relationship between Capabilities, Practices, and Outcomes. This is in line with what was already summarized in Figure 1 for supporting the research framework of analysis in order for researchers and practitioners to benefit from its insights.

The Capabilities, with implicit skills and knowledge, enabling or needed for Practices already seen in Figure 10 offer direction for team members in training, upskilling and defining which Practices are essential to DevOps and how they can be done effectively. Here we can also observe Practices and combinations of Practices resulting in Capabilities for the team or organization, which helps to understand the expected results of Capabilities and which are required to achieve them. Just like the capability to respond to consumer feedback, certain Capabilities are high-level and abstract, requiring the integration of practices, contextual factors, culture and technology. It is also possible to create a new Capability by applying groups of Practices or individually.

DevOps is a set of Practices (and principles) that demands the team and its members to have specified Capabilities and utilize specific Practices to accomplish the desired Outcomes of the Practices, independently and collectively.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

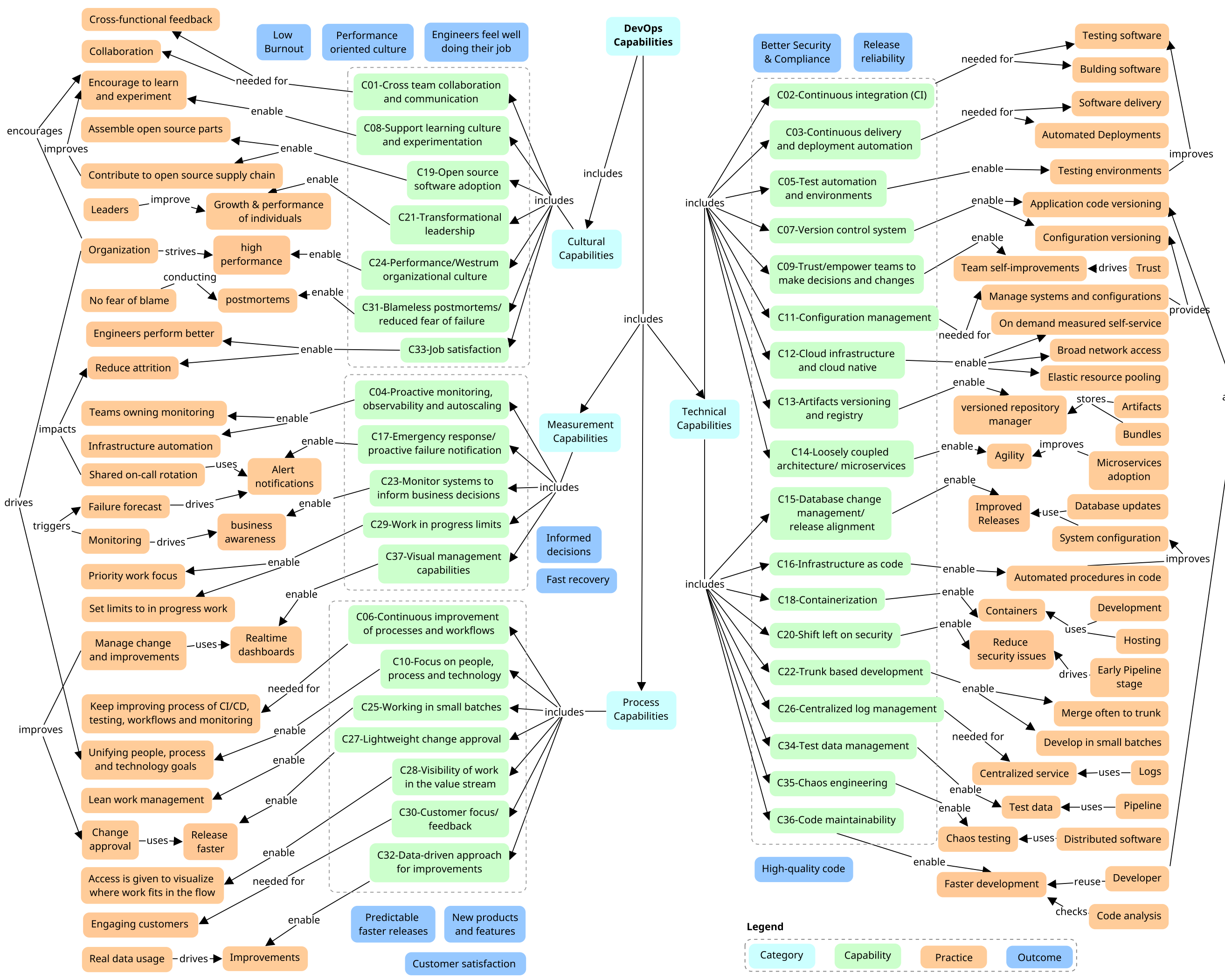

DevOps Capabilities
Cultural Capabilities
Measurement Capabilities
Technical Capabilities
Process Capabilities
includes
C01-Cross team collaboration and communication
C08-Support learning culture and experimentation
C19-Open source software adoption
C21-Transformational leadership
C24-Performance/Westrum organizational culture
C31-Blameless postmortems/ reduced fear of failure
C33-Job satisfaction
C04-Proactive monitoring, observability and autoscaling
C17-Emergency response/ proactive failure notification
C23-Monitor systems to inform business decisions
C29-Work in progress limits
C37-Visual management capabilities
C06-Continuous improvement of processes and workflows
C10-Focus on people, process and technology
C25-Working in small batches
C27-Lightweight change approval
C28-Visibility of work in the value stream
C30-Customer focus/ feedback
C32-Data-driven approach for improvements
C02-Continuous integration (CI)
C03-Continuous delivery and deployment automation
C05-Test automation and environments
C07-Version control system
C09-Trust/empower teams to make decisions and changes
C11-Configuration management
C12-Cloud infrastructure and cloud native
C13-Artifacts versioning and registry
C14-Loosely coupled architecture/ microservices
C15-Database change management/ release alignment
C16-Infrastructure as code
C18-Containerization
C20-Shift left on security
C22-Trunk based development
C26-Centralized log management
C34-Test data management
C35-Chaos engineering
C36-Code maintainability
Low Burnout
Performance oriented culture
Engineers feel well doing their job
Informed decisions
Fast recovery
Predictable faster releases
New products and features
Customer satisfaction
Better Security & Compliance
Release reliability
High-quality code
Cross-functional feedback
Collaboration
Encourage to learn and experiment
Assemble open source parts
Contribute to open source supply chain
Leaders
Growth & performance of individuals
Organization
high performance
No fear of blame
postmortems
Engineers perform better
Reduce attrition
Teams owning monitoring
Infrastructure automation
Shared on-call rotation
Alert notifications
Failure forecast
business awareness
Monitoring
Priority work focus
Set limits to in progress work
Manage change and improvements
Realtime dashboards
Keep improving process of CI/CD, testing, workflows and monitoring
Unifying people, process and technology goals
Lean work management
Change approval
Release faster
Access is given to visualize where work fits in the flow
Engaging customers
Real data usage
Improvements
Testing software
Bulding software
Software delivery
Automated Deployments
Testing environments
Application code versioning
Configuration versioning
Team self-improvements
Trust
Manage systems and configurations
On demand measured self-service
Broad network access
Elastic resource pooling
versioned repository manager
Artifacts
Bundles
Agility
Microservices adoption
Improved Releases
Database updates
System configuration
Automated procedures in code
Containers
Development
Hosting
Reduce security issues
Early Pipeline stage
Merge often to trunk
Develop in small batches
Centralized service
Logs
Test data
Pipeline
Chaos testing
Distributed software
Faster development
Developer
Code analysis
needed for
enable
encourages
improves
improve
strives
conducting
impacts
uses
drives
triggers
stores
provides
access
use
reuse
checks
Legend
Category
Capability
Practice
Outcome

Figure 10. DevOps Capabilities and Practices instantiation.

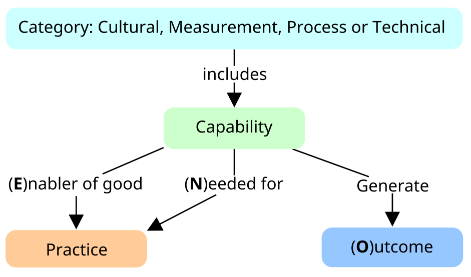


Figure 11. Related Capabilities, Practices and Outcomes.

New organizational capabilities are one of the anticipated results. Increase in DevOps maturity is a consequence of enhancing Capabilities by improving Practices or Processes to more consistently deliver certain idealized outcomes.

Some Capabilities may be thought of as *Enablers* to effective Practice. For example, for DevOps to be successful, the practice of upskilling Dev with Ops Capabilities, is an enabler. Enablers will generally improve the DevOps Capabilities in the various categories, setting the standards of a well performed Practice.

Enablers of **Cultural Capabilities** require all teams and engineers to be aware of the common objectives. People should collaborate on diverse tasks, sharing responsibilities and understanding the overall software development system and common goal of all teams. Disrespecting colleagues and emphasizing personal job success are all examples of poor DevOps practices. The organization should encourage experimenting and learning from errors with no fear of blame. This contributes to growth, performance and creativity of individuals.

Measurement Capabilities Enablers motivate teams to take responsibility for monitoring and sharing the on-call rotation, while performing observability, proactive monitoring and autoscaling of systems improves the expectation on reliability. A good Practice example is making sure there is monitoring for all critical systems, but an opposite situation would be too many alerts causing toil and burnout or even no alerts at all. With proper measurement, the organization could also explore failure forecast or event set limits to in progress work with real-time dashboards.

Process Capabilities Enablers help to improve the process of CI/CD, testing, workflows and monitoring access

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

is given to visualize where work fits in the flow of Lean work management with real data usage for improvements. It is possible to see where the task fits into the overall flow using real-world data collection with the purpose of making changes reliable. A lengthy and heavy change approval process is an example of a bad practice where the Continuous Delivery Capability would be affected leading to increased technical debt, lack of timely security updates or attrition. These Enablers also bringing together the focus of people, processes, and technology.

Technical Capabilities Enablers have main purposes like analyzing and evaluating software, automation of testing, building, deployment using environments for software testing and versioning of software. The use of bundles and artifacts, including configuration management best practices, enable distributing software faster. Also, guiding team self-improvements, leading to many new practices based on engineers creativity. An example of a good technical Practice is using an artifact repository manager for versioning the built artifacts from a pipeline, and a poor Practice would be to not properly scan these artifacts for vulnerabilities or not promoting them along the testing environments.

The intent of each Capability is not only to enable Practices but also to drive Outcomes and results as seen in Figure 12.

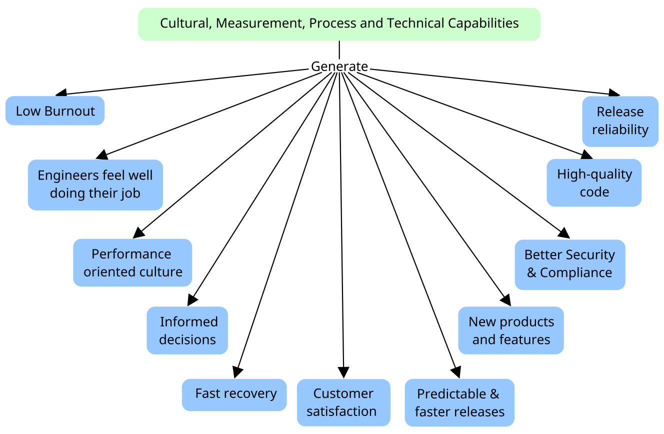

Cultural, Measurement, Process and Technical Capabilities
Generate
Low Burnout
Engineers feel well doing their job
Performance oriented culture
Informed decisions
Fast recovery
Customer satisfaction
Predictable & faster releases
New products and features
Better Security & Compliance
High-quality code
Release reliability

Figure 12. Capabilities generating Outcomes.

The exact combination of Practices and Capabilities that lead to each Outcomes and results [9], [18] are very extensive and not the particular subject of this research in detail. In any case, it is seen from literature that there is a close proximity to each category returned from Section 6.2 and noted in Figure 10. Namely, *Low Burnout*, *Performance oriented culture* and *Engineers feel well doing their job* is most positively impacted by Cultural Capabilities, while business well *Informed decisions* and *Fast recovery* (Mean time to restore), are associated to Measurement Capabilities. On the other hand, it is seen that *Predictable faster delivery/releases* (deployment frequency and lead time to deploy), *New products and features*, and *Customer satisfaction* is related to (but not only) Process Capabilities. Finally, *Better Security & Compliance*, *Release reliability* (Change failure rate), and *High-quality code* are more inclined for improving from Technical Capabilities side.

The **Capabilities needed for Practices** towards the desired outcomes takes a set of essential Capabilities for DevOps that were initially discussed by Smeds and Senapathi et al. [55], [61] together with configuration management [42], [176], seen in the relations of Figure 10. Namely, (C01) Cross team collaboration and communication, (C02) Continuous Integration, (C03) Continuous Delivery/Deployment automation, (C04) Proactive Monitoring, Observability and autoscaling, (C06) Continuous Improvement of processes/workflows, (C11) Configuration Management, (C17) Emergency response/proactive failure notification and (C30) Customer focus/feedback.

To employ a Practice or set of Practices, a person (or team) may *need* certain knowledge as well as skills. This implies they have the Capability to perform a certain Practice to reach the desired Outcome with a high degree of certainty. Any Practice implies a set of Capabilities that a person or team must have in order to utilize that Practice. To utilize the Practice of CI, for example, a team must grasp the process, how to execute a collection of other Practices, and how to use the needed tools recognizing the benefits. Given the essential feedback loop and collaboration, the team will create and enhance Capabilities progressively, bringing the need of the assessment of the result and experimentation with adjustments to their Practice execution, therefore enhancing their Capability. DevOps "Capability" is a (developing) quality of a person or team, whereas "Practice" is part of a process.

These new insights from the presented research have particular implications for practitioners and researchers in this area that should be taken into consideration for a general consensus on how to relate Capabilities and Practices towards improved Outcomes.

7 Conclusion

This study has brought important contributions to both academia and industry on the DevOps topic. A Multivocal Literature Review was conducted, in order to identify and sort and organize an updated list of 37 Capabilities or Practices in Section 6.2, how frequently are Practices or Capabilities mentioned in gray literature compared to academic literature in Section 6.3 and how authors are aligned on these relations in Section 6.4 leading to their definitions, instantiation and analysis in Section 6.5.

Relevant research was done to understand and synthesize the main DevOps Capabilities that are mentioned in publications and how they relate to DevOps Practices. Capabilities were mapped and grouped in an organized taxonomy. A research framework of analysis was used to analyze data in a structured way, relating Capabilities and Practices findings. It was analyzed how the relation of the several Capabilities and Practices are required by DevOps for generating Outcomes and results.

This study will help researchers and practitioners to understand the background and relation of these concepts, giving a clear picture of how to execute them as a description of the Capabilities (skills and knowledge) required to do these Practices well. It was also observed that the Capabilities are dynamic and have been changing and growing throughout time.

Therefore, the list of the most studied and approached Capabilities was collected as it was proposed. Of which,

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

it is highlighted *cross team collaboration and communication*. In this collaboration, developers have access to a self-service platform that provides a foundation for automation, standardization and team autonomy in order to enable the other three most mentioned Capabilities: *continuous integration; continuous delivery and deployment automation; proactive monitoring, observability and autoscaling*.

At a time when DevOps is increasingly more adopted by organizations and examined by researchers, this study assumes a critical role in the consensus of relating the main Capabilities to Practices and their definitions so that the topic can continue to be applied and studied in a more solid and grounded way.

7.1 Threats to validity

We considered types of validity pertaining to qualitative research. Limitations of this study include the fact that it is based on multivocal literature, therefore the majority of the material has not gone through the critical peer-review process that academic research is typically exposed to. To mitigate the impact of this danger, it was chosen to design the review procedure using the recommendations given by Garousi et al. [38] and to conduct each step using this method. Specifically, systematic procedures and logistics, such as clear traceability links between collected data and primary sources, were used throughout data extraction.

The validity from the external environment and the generalization of the research results is threatened by our overreliance on Google search, EBSCO and Web of Science for our data sources, thus academic peer-reviewed literature from IEEE, ACM and Scopus serves as a counterbalance for good measure.

Another restriction is the inclusion of English-only articles, which may exclude significant studies in other languages.

7.2 Future work

In the future work that will be following this MLR, it is intended to move forward with a design science research using the information gathered in this article, in order to create an artifact with relation to the Capabilities gathered here and the main DevOps metrics for each one.

Mapping the possible Outcomes and the metrics for these Outcomes that have most impact is also a targeted work for the near future.

Answering in detail what practices and what combinations of practices provide the team or organization with what capabilities as an outcome.

After this, a formalized ontology could be compiled to systematize the other concepts found.

References

[1] N. Forsgren, M. C. Tremblay, D. VanderMeer, and J. Humble, "DORA Platform: DevOps Assessment and Benchmarking," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, 2017, pp. 436–440.

[2] J. P. L. . K. C. Laudon, *Management Information Systems: Managing the Digital Firm, Global Edition*, 15th ed. Pearson Education, 2017.

[3] A. Dyck, R. Penners, H. Lichter, and IEEE, "Towards Definitions for Release Engineering and DevOps," in *2015 IEEE/ACM 3RD INTERNATIONAL WORKSHOP ON RELEASE ENGINEERING*, no. 3rd International Workshop on Release Engineering. IEEE Press, may 2015, p. 3.

[4] N. Forsgren and J. Humble, "The Role of Continuous Delivery in it and Organizational Performance," *SSRN Electronic Journal*, pp. 1–15, 2015. [Online]. Available: http://www.ssrn.com/abstract=2681909

[5] D. N. Blank-edelman, *Seeking SRE: Conversations About Running Production Systems at Scale*. O'Reilly Media, Inc., 2018.

[6] R. W. Macarthy and J. M. Bass, "An Empirical Taxonomy of DevOps in Practice," in *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, M. A., W. M., and S. A., Eds. University of Salford, School of Science, Engineering and Environment, Manchester, United Kingdom: Institute of Electrical and Electronics Engineers Inc., aug 2020, pp. 221–228.

[7] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Addison-Wesley Professional, dec 2010.

[8] K. Mikko, "DevOps Capability Assessment in a Software Development Team," *Science*, vol. 135, pp. 408–415, 2018. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/334710/MikkoKurkela_DevOpsCapabilityAssessmentInASoftwareDevelopmentTeam.pdf

[9] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and Devops: Building and Scaling High Performing Technology Organizations*. IT Revolution, 2018.

[10] Puppet Labs, "2015 State of DevOps Report," Puppet Labs, Tech. Rep. 877, 2015. [Online]. Available: http://puppetlabs.com/2015-devops-report

[11] M. E. Pasten, M. Siyam, and D. Academy, "Role-based DevOps Capability Model," pp. 1–12, 2020. [Online]. Available: https://devonacademy.com/enterprise-devops-capability-model/

[12] G. Kim, K. Behr, K. Spafford, and G. Spafford, *The phoenix project: A novel about IT, DevOps, and helping your business win*. IT Revolution, 2014. [Online]. Available: https://books.google.pt/books?id=H6x-DwAAQBA

[13] R. Pereira and J. Serrano, "A review of methods used on IT maturity models development: A systematic literature review and a critical analysis," *Journal of Information Technology*, vol. 35, no. 2, pp. 161–178, jun 2020.

[14] U. S. Bititci, P. Garengo, A. Ates, and S. S. Nudurupati, "Value of maturity models in performance measurement," *International Journal of Production Research*, vol. 53, no. 10, pp. 3062–3085, may 2015.

[15] Google, "DevOps capabilities | DORA - Google Cloud," 2020. [Online]. Available: https://cloud.google.com/solutions/devops/capabilities

[16] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7884954/

[17] L. E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo, C. Lassenius, V. Heikkila, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo, and C. Lassenius, "DevOps in practice: A multiple case study of five companies," *Information and Software Technology*, vol. 114, no. March 2017, pp. 217–230, Oct. 2019.

[18] J. Díaz, D. López-Fernández, J. Pérez, and Á. González-Prieto, "Why are many businesses installing a DevOps culture into their organization?" *Empirical Software Engineering*, vol. 26, no. 2, 2021.

[19] G. Adzic and M. Bisset, *Impact Mapping: Making a big impact with software products and projects*. Provoking Thoughts Limited, 2012.

[20] S. Badshah, A. A. Khan, and B. Khan, "Towards Process Improvement in DevOps: A Systematic Literature Review," in *24th Evaluation and Assessment in Software Engineering Conference, EASE 2020*, ser. EASE '20. Comsats University Islamabad, Islamabad, Pakistan: Association for Computing Machinery, apr 2020, pp. 427–433.

[21] F. M. A. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," *Journal of Software: Evolution and Process*, vol. 29, no. 6, p. e1885, 2017.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

[22] M. Sánchez-Gordón and R. Colomo-Palacios, "A multivocal literature review on the use of DevOps for e-learning systems," in *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM'18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 883–888.

[23] B. B. N. de França, H. Jeronimo, and G. H. Travassos, "Characterizing DevOps by Hearing Multiple Voices," in *Proceedings of the 30th Brazilian Symposium on Software Engineering*, ser. SBES '16, E. DeAlmeida, Ed., Unicesumar; Colivre; Espweb; Tasa Eventos. New York, NY, USA: Association for Computing Machinery, 2016, Proceedings Paper, pp. 53–62.

[24] L. Prates, J. Faustino, M. Silva, and R. Pereira, "DevSecOps Metrics," in *Lecture Notes in Business Information Processing*, M. J. Wrycza S., Ed. Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal: Springer, 2019, vol. 359, pp. 77–90.

[25] L. E. Lwakatare, T. Karvonen, T. Sauvola, P. Kuvaja, H. H. Olsson, J. Bosch, and M. Oivo, "Towards DevOps in the embedded systems domain: Why is it so hard?" in *49th Annual Hawaii International Conference on System Sciences, HICSS 2016*, S. R.H. and B. T.X., Eds., vol. 2016-March. University of Oulu, Finland: IEEE Computer Society, 2016, pp. 5437–5446.

[26] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," *Communications in Computer and Information Science*, vol. 770, no. 1, pp. 17–29, 2017.

[27] Puppet Labs, "2013 State of DevOps Report," Puppet Labs, Tech. Rep., 2013.

[28] ——, "2014 State of DevOps Report," Puppet Labs, Tech. Rep., 2014. [Online]. Available: http://puppetlabs.com/2014-devops-report

[29] ——, "2016 State of DevOps Report," Puppet Labs, Tech. Rep. 7, 2016. [Online]. Available: https://puppetlabs.com/solutions/devops/

[30] ——, "2017 State of DevOps Report," Puppet Labs, Tech. Rep., 2017. [Online]. Available: https://puppetlabs.com/solutions/devops/

[31] ——, "2018 State of DevOps Report," Puppet Labs, Tech. Rep., 2018. [Online]. Available: https://media.webteam.puppet.com/uploads/2019/11/Puppet-State-of-DevOps-Report-2018_update.pdf

[32] ——, "2020 State of DevOps Report," Puppet Labs, Tech. Rep., 2020. [Online]. Available: https://puppet.com/resources/report/2020-state-of-devops-report/

[33] N. PR, "IT Revolution Announces Second Round of Speakers for DevOps Enterprise Summit London 2019." *PR Newswire US*, 2019.

[34] C. Cook and A. M. Fred, "DevOps Enterprise Summit / Metrics We Love / 24 June 2020 / © 2020 IBM Corporation 1," pp. 1–42, 2020. [Online]. Available: https://videolibrary.doesvirtual.com/?video=431640000

[35] D. Teixeira, R. Pereira, T. A. Henriques, M. Silva, and J. Faustino, "A Systematic Literature Review on DevOps Capabilities and Areas," *International Journal of Human Capital and Information Technology Professionals*, vol. 11, no. 2, pp. 1–22, apr 2020.

[36] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–35, nov 2019.

[37] G. B. Ghantous and A. Q. Gill, "DevOps: Concepts, practices, tools, benefits and challenges," in *21st Pacific Asia Conference on Information Systems: Societal Transformation Through IS/IT, PACIS 2017*. School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia: Association for Information Systems, 2017, p. 12.

[38] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, no. September 2018, pp. 101–121, feb 2019.

[39] S. Mäkinen, M. Leppänen, T. Kilamo, A.-L. Mattila, E. Laukkanen, M. Pagels, and T. Männistö, "Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises," *Information and Software Technology*, vol. 80, pp. 175–194, dec 2016.

[40] P. Debois, "Agile infrastructure and operations: How infra-gile are you?" *Proceedings - Agile 2008 Conference*, pp. 202–207, 2008.

[41] J. Allspaw and P. Hammond, "10+ deploys per day: Dev and ops cooperation at Flickr," in *Velocity: web performance and operations conference*, 2009. [Online]. Available: https://www.youtube.com/watch?v=LdOe18KhtT4

[42] J. I. Olszewska, "IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021," IEEE Standards Association, Tech. Rep. 16 Apr 2021, 2021.

[43] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps?" in *Proceedings of the Scientific Workshop Proceedings of XP2016*, ser. XP '16 Workshops. New York, NY, USA: ACM, may 2016, pp. 1–11.

[44] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Computer Science Review*, vol. 38, no. 1, p. 100308, nov 2020.

[45] J. Humble and J. Molesky, "Why enterprises must adopt devops to enable continuous delivery," *Cutter IT Journal*, vol. 24, no. 8, pp. 6–12, 2011.

[46] J. Willis, "DevOps Culture (Part 1) - IT Revolution," 2012. [Online]. Available: https://itrevolution.com/devops-culture-part-1/

[47] Gene Kim, "The Three Ways: The Principles Underpinning DevOps," 2012. [Online]. Available: https://itrevolution.com/the-three-ways-principles-underpinning-devops/

[48] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Software*, vol. 33, no. 3, pp. 94–100, may 2016.

[49] B. Tessem and J. Iden, "Cooperation between developers and operations in software engineering projects," *Proceedings - International Conference on Software Engineering*, pp. 105–108, 2008.

[50] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, "DevOps adoption benefits and challenges in practice: A case study," *17th International Conference on Product-Focused Software Process Improvement, PROFES 2016*, vol. 10027 LNCS, pp. 590–597, 2016.

[51] B. A. Kitchenham, "Systematic review in software engineering," in *Proceedings of the 2nd international workshop on Evidential assessment of software technologies - EAST '12*. New York, New York, USA: ACM Press, 2012, p. 1.

[52] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering - EASE '16*. New York, New York, USA: ACM Press, 2016, pp. 1–6.

[53] R. T. Ogawa and B. Malen, "Towards Rigor in Reviews of Multivocal Literatures: Applying the Exploratory Case Study Method," *Review of Educational Research*, vol. 61, no. 3, pp. 265–286, sep 1991.

[54] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th international conference on Software engineering*, Keele University and Durham University Joint Report. New York, NY, USA: ACM, may 2006, pp. 1051–1052.

[55] J. Smeds, K. Nybom, and I. Porres, "DevOps: A Definition and Perceived Adoption Impediments," in *Lecture Notes in Business Information Processing*. Springer, 2015, vol. 212, pp. 166–177.

[56] DORA, "DORA research program," 2020. [Online]. Available: https://www.devops-research.com/research.html

[57] BMC, "State of DevOps 2020: A Report Roundup," *BMC*, 2020. [Online]. Available: https://www.bmc.com/blogs/state-of-devops/

[58] J. Mitlöhner, S. Neumaier, J. Umbrich, and A. Polleres, "Characteristics of open data CSV files," in *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE, 2016, pp. 72–79.

[59] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," *ACM International Conference Proceeding Series*, 2014.

[60] A. Novak, "Six Core Capabilities of a DevOps Practice – The New Stack," 2014. [Online]. Available: https://thenewstack.io/six-core-capabilities-of-a-devops-practice/

[61] M. Senapathi, J. Buchan, and H. Osman, "DevOps Capabilities, Practices, and Challenges," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 - EASE'18*, ser. EASE'18, no. June. New York, USA: ACM Press, jun 2018, pp. 57–67.

[62] D. Teixeira, R. Pereira, T. Henriques, M. M. D. Silva, and J. Faustino, "A maturity model for DevOps," *International Journal of Agile Systems and Management*, vol. 13, no. 4, p. 464, 2020.

[63] M. M. A. Ibrahim, S. M. Syed-Mohamad, and M. H. Husin, "Managing Quality Assurance Challenges of DevOps through Analytics," in *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, ser. ICSCA '19, vol. Part F1479.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

New York, NY, USA: Association for Computing Machinery, feb 2019, pp. 194–198.

[64] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook : How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016.

[65] B. I. Staff, "10 DevOps Best Practices To Know | Built In," pp. 1–7, 2020. [Online]. Available: https://builtin.com/software-engineering-perspectives/ devops-best-practices

[66] D. Linthicum, "DevOps tools best practices: A 7-step guide," 2016. [Online]. Available: https://techbeacon.com/devops/7-steps-choosing-right-devops-tools

[67] R. de Feijter, S. Overbeek, R. van Vliet, E. Jagroep, and S. Brinkkemper, "DevOps competences and maturity for software producing organizations," *Lecture Notes in Business Information Processing*, vol. 318, pp. 244–259, 2018.

[68] J. Groll, "What is a DevOps 'Best Practice'? - DevOps.com," 2016. [Online]. Available: https://devops.com/devops-best-practice/

[69] I. Karamitsos, S. Albarhami, and C. Apostolopoulos, "Applying devops practices of continuous automation for machine learning," *Information (Switzerland)*, vol. 11, no. 7, pp. 1–15, jul 2020.

[70] K. Eby, "The Tools and Technology of DevOps | Smartsheet," pp. 1–26, 2017. [Online]. Available: https://www.smartsheet.com/ devops-tools

[71] K. Kuusinen, V. Balakumar, S. C. Jepsen, S. H. Larsen, T. A. Lemqvist, A. Muric, A. Ø. O. Nielsen, and O. Vestergaard, "A large agile organization on its journey towards DevOps," in *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018*, B. T. and A. L., Eds. University of Southern Denmark, Odense, Denmark: Institute of Electrical and Electronics Engineers Inc., aug 2018, pp. 60–63.

[72] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, H. Shen, L. Chen, and K. Lu, "Preliminary Findings about DevSecOps from Grey Literature," in *20th IEEE International Conference on Software Quality, Reliability, and Security, QRS 2020*. Nanjing University, State Key Laboratory for Novel Software Technology, Nanjing, China: Institute of Electrical and Electronics Engineers Inc., dec 2020, pp. 450–457.

[73] M. D. Z. Imam, "What are the Best Practices For Successful Implementation Of DevOps?" 2018. [Online]. Available: https://www.knowledgehut.com/blog/devops/best-practices-for-successful-implementation-of-devops

[74] Netapp, "What Is DevOps? - Practices and Benefits Explained | NetApp," pp. 1–12, 2019. [Online]. Available: https: //www.netapp.com/devops-solutions/what-is-devops/

[75] T. Hall, "DevOps Best Practices | Atlassian," 2020. [Online]. Available: https://www.atlassian.com/devops/what-is-devops/ devops-best-practices

[76] L. Yin and V. Filkov, "Team Discussions and Dynamics during DevOps Tool Adoptions in OSS Projects," in *Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*, ser. ASE '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 697–708.

[77] J. Humble and B. O'Reilly, *Lean enterprise : adopting continouos delivery, devops, and lean startup at scale*. O'Reilly Media, Inc., 2014.

[78] I. T. Revolution, "24 Key Capabilities to Drive Improvement in Software Delivery - IT Revolution," 2020. [Online]. Available: https://itrevolution.com/24-key-capabilities-to-drive-improvement-in-software-delivery/

[79] H. T. C. G. Services, "24 DevOps Capabilities - DevOps Efficiency Matrix," 2020. [Online]. Available: https://devopsefficiency.com/ list.html

[80] E. DeBoer, "Accelerate: A Principle-based DevOps Framework," 2019. [Online]. Available: https://blog.sonatype.com/principle-based-devops-frameworks-accelerate

[81] P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen, and P. Kuvaja, "Advances in Using Agile and Lean Processes for Software Development," in *Advances in Computers*, ser. Advances in Computers, M. A.M., Ed. Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland: Academic Press Inc., 2019, vol. 113, pp. 135–224.

[82] H. Ketterer, "Leaner, Faster, and Better with DevOps," 2017. [Online]. Available: https://www.bcg.com/publications/2017/ technology-digital-leaner-faster-better-devops

[83] W. P. Luz, G. Pinto, and R. Bonifácio, "Adopting DevOps in the real world: A theory, a model, and a case study," *Journal of Systems and Software*, vol. 157, no. July, p. 110384, Nov. 2019.

[84] PuppetGuideCIOs, "The 5 Stages of DevOps Evolution: A Guide for CIOs," 2018. [Online]. Available: https://www.thinkahead.com/wp-content/uploads/2018/ 12/puppet-5-stages-devops-evolution-cio-guide.pdf

[85] O. Mikhalchuk, "5 core DevOps principles for a painless culture shift | Forte Group," 2020. [Online]. Available: https://fortegrp.com/what-are-the-core-devops-principles/

[86] Infopulse, "6 DevOps Best Practices to Launch Enterprise-Wide Transformations | Continuous Delivery, Microservices Architecture, Collaboration | Infopulse," 2018. [Online]. Available: https://www.infopulse.com/blog/6-devops-best-practices-to-launch-enterprise-wide-transformations/

[87] Puppet Labs, "2019 State of DevOps Report," Puppet Labs, Tech. Rep., 2019.

[88] S. Team, "Understanding 6 Essential DevOps Principles - SKILLOGIC Official Blog," 2018. [Online]. Available: https://skillogic.com/blog/understanding-6-essential-devops-principles/

[89] J. Wade, "Devops Best Practices Checklist · GitHub," pp. 1–8, 2017. [Online]. Available: https://gist.github.com/jpswade/ 4135841363e72ece8086146bd7bb5d91

[90] Vilmate, "7 DevOps Best Practices for a Project Success | Vilmate," 2020. [Online]. Available: https://vilmate.com/blog/devops-best-practices/

[91] RedHat, "What is DevSecOps?" 2018. [Online]. Available: https: //www.redhat.com/en/topics/devops/what-is-devsecops

[92] T. F. Düllmann, C. Paule, A. van Hoorn, T. F. Dullmann, C. Paule, A. van Hoorn, and IEEE, "Exploiting devops practices for dependable and secure continuous delivery pipelines," in *Proceedings - International Conference on Software Engineering*, ser. RCoSE '18, no. 4th ACM/IEEE International Workshop on Rapid Continuous Software Engineering (RCoSE), IEEE Comp Soc; Assoc Comp Machinery; SIGSOFT; IEEE Tech Council Software Engn. New York, NY, USA: Association for Computing Machinery, 2018, Proceedings Paper, pp. 27–30.

[93] C. S. Tov, "DevOps Best Practices: Take Your Pipeline to the Next Level - Codemotion," 2020. [Online]. Available: https://www.codemotion.com/magazine/dev-hub/ devops-engineer/devops-best-practices/

[94] M. Zulfahmi Toh, S. Sahibuddin, and M. N. Mahrin, "Adoption issues in DevOps from the perspective of continuous delivery pipeline," in *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, ser. ICSCA '19, vol. Part F1479. New York, NY, USA: Association for Computing Machinery, 2019, pp. 173–177.

[95] I. Pavlenko, "DevOps: Principles, Practices, and DevOps Engineer Role," 2021. [Online]. Available: https://www.altexsoft.com/blog/engineering/devops-principles-practices-and-devops-engineer-role/

[96] L. Bass, Len; Weber, Ingo; Zhu, *DevOps: A Software Architect's Perspective*, 1st ed. Addison-Wesley Professional, 2015.

[97] J. Faustino, "DevOps Practices in Incident Management Process," Thesis, ISCTE-IUL, 2018. [Online]. Available: https://repositorio.iscte-iul.pt/bitstream/10071/18294/1/ Master_Joao_Carvalho_Faustino.pdf

[98] N. Ceresani, "The 4 Core Capabilities of DevOps," 2017. [Online]. Available: https://dzone.com/articles/the-4-core-capabilities-of- devops

[99] I. Sacolick, "15 KPIs to track devops transformation," 2018. [Online]. Available: https://www.infoworld.com/article/ 3297041/15-kpis-to-track-devops-transformation.html

[100] C. A. Technologies, "How Can DevOps Practices Help You Increase - Broadcomdocs.broadcom.com," 2017. [Online]. Available: https://docs.broadcom.com/doc/how-can-devops-practices-help-you-increase-innovation-velocity-and-business-agility-on-the-mainframe

[101] DevOps Research and Assessment (DORA), D. Research, and A. (DORA), "State of DevOps 2019 - DORA," DORA, Tech. Rep., 2019. [Online]. Available: https://services.google.com/fh/files/ misc/state-of-devops-2019.pdf

[102] C. Us and StarAgile, "Top 7 DevOps practices for Successful Implementation of DevOps," pp. 1–8, 2020. [Online]. Available: https://staragile.com/blog/devops-best-practices

[103] C. Crowley, L. McQuillan, and C. O'Brien, "Understanding DevOps: Exploring the origins, composition, merits, and perils of a DevOps Capability," in *Proceedings of the 4th International Conference on Production Economics and Project Evaluation, ICOPEV 2018, Guimarães, Portugal*, ser. ICOPEV International Conference

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

on Project Economic Evaluation, M. Araujo, Ed., Univ Minho, Sch Engn, Ind Engn and Management, Algoritmi Res Ctr. Guimaraes: Universade do Minho, 2018, Proceedings Paper, pp. 29–37.

[104] Scaledagile, "DevOps - Scaled Agile Framework," 2021. [Online]. Available: https://www.scaledagileframework.com/devops/

[105] Wikipedia, "DevOps - Wikipedia," 2021. [Online]. Available: https://en.wikipedia.org/wiki/DevOps

[106] E. Mueller, "What Is DevOps? | the agile admin," 2019. [Online]. Available: https://theagileadmin.com/what-is-devops/

[107] Veritis, "DevOps Capabilities: 6-point Principles for Business Success," 2018. [Online]. Available: https://www.veritis.com/blog/devops-capabilities-a-6-point-principle-that-drives-business-success/

[108] A. Lichtenberger, "Blog: Agile: Dead End? Taking the next step by applying DevOps Practices effectively - impact matters Blog," 2019. [Online]. Available: https://www.impactmatters.ch/blog/agiledevops-deadend/

[109] Department of Energy Quality Managers: Software Quality Assurance Subcommittee, "Software Configuration Management (SCM) A Practical Guide," United States Department of Energy, Tech. Rep., 2000. [Online]. Available: http://energy.gov/sites/prod/files/cioprod/documents/ scmguide.pdf

[110] ANSI/IEEE, "IEEE Std 1042-1987 Guide to Software Configuration Management," American National Standards Institute, Tech. Rep., 1987.

[111] P. M. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[112] D. Stahl, T. Martensson, J. Bosch, D. Ståhl, T. Mårtensson, J. Bosch, D. Stahl, T. Martensson, and J. Bosch, "Continuous practices and devops: beyond the buzz, what does it all mean?" in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, F. M., O. H.H., and S. A., Eds., vol. 2017-Janua. Ericsson AB, Linköping, Sweden: Institute of Electrical and Electronics Engineers Inc., aug 2017, pp. 440–448.

[113] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software*, vol. 33, no. 3, pp. 42–52, may 2016.

[114] ——, "Migrating to Cloud-Native Architectures Using Microservices: An Experience Report," *Communications in Computer and Information Science*, vol. 567, pp. 201–215, jul 2015.

[115] M. Stillwell and J. G. F. Coutinho, "A DevOps approach to integration of software components in an EU research project," in *Proceedings of the 1st International Workshop on Quality-Aware DevOps*, Imperial College London United Kingdom. New York, NY, USA: ACM, sep 2015, pp. 1–6.

[116] R. Souza, F. Silva, L. Rocha, I. Machado, F. Silva, and I. Machado, "Investigating Agile Practices in Software Startups," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, ser. SBES 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 317–321.

[117] A. Steffens, H. Lichter, and J. S. Döring, "Designing a next-generation continuous software delivery system: Concepts and architecture," *Proceedings - International Conference on Software Engineering*, pp. 1–7, 2018.

[118] BoxBoat, "What is DevOps? Exploring DevOps Principles & Benefits | BoxBoat," 2018. [Online]. Available: https://boxboat.com/2018/12/26/what-is-devops/

[119] C. Marnewick and J. Langerman, "DevOps and Organisational Performance: The Fallacy of Chasing Maturity," *IEEE Software*, pp. 0–0, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9190017/

[120] R. Feijter, R. Vliet, E. Jagroep, S. Overbeek, and S. Brinkkemper, "Towards the adoption of DevOps in software product organizations: A maturity model approach," *Technical Report Series*, no. UU-CS-2017-009, 2017.

[121] A. Valdes, "5 DevOps Metrics and KPIs that CTOs Must Monitor," pp. 1–9, 2020. [Online]. Available: https://www.clickittech.com/devops/devops-metrics-and-kpis/

[122] V. Fedak, "DevOps metrics: what to track, how and why do it." pp. 1–11, 2020. [Online]. Available: https://medium.com/@FedakV/devops-metrics-what-to-track-how-and-why-do-it-e08dc6864eab

[123] E. Lock, "Measure DevOps Metrics That Matter," 2020. [Online]. Available: https://www.devopsdigest.com/measure-devops-metrics-that-matter

[124] D. Sato, "Practices for DevOps and Continuous Delivery," 2015. [Online]. Available: https://www.infoq.com/articles/book-DevOps-continuous-delivery/

[125] Devopedia, "DevOps Metrics," 2019. [Online]. Available: https://devopedia.org/devops-metrics

[126] G. Motroc, "Key DevOps metrics that matter: How well does your team sleep?" pp. 1–13, 2018. [Online]. Available: https://jaxenter.com/devops-influencers-interview-series-4-142312.html

[127] AWS, "What is DevOps? - Amazon Web Services (AWS)," 2021. [Online]. Available: https://aws.amazon.com/devops/what-is-devops/

[128] CMMI Product Team, "CMMI for Development, Version 1.3," *Software Engineering Process Management Program*, 2010.

[129] N. Forsgren, J. Humble, G. Kim, A. Brown, and N. Kersten, "Accelerate: State of DevOps 2018 Strategies for a New Economy," *Report. DevOps Research & Assessment (DORA)*, p. 78, 2018. [Online]. Available: https://cloudplatformonline.com/rs/248-TPC-286/images/DORA-State of DevOps.pdf

[130] Intellipaat, "What is DevOps - Introduction to DevOps Architecture & Benefits," 2016. [Online]. Available: https://intellipaat.com/blog/what-is-devops/

[131] H. Packard, "Measuring DevOps success," 2016. [Online]. Available: http://www.baldrover.com/wp-content/uploads/Measuring-DevOps-Success.pdf

[132] A. R. Chowdhury, "DevOps Best Practices: A Complete Guide," 2019. [Online]. Available: https://stackify.com/devops-best-practices-a-complete-guide/

[133] K. Wakayama, "How to Ensure the Success of DevOps in Your Organization," pp. 1–10, 2020. [Online]. Available: https://codersociety.com/blog/articles/devops-success-in-organization

[134] Ubiq, "Top DevOps Metrics and KPIs To Monitor Regularly - Ubiq BI Blog," pp. 1–12, 2020. [Online]. Available: http://ubiq.co/analytics-blog/top-devops-metrics-kpis-to-monitor-regularly/

[135] R. Westrum, "A typology of organisational cultures," *Quality and Safety in Health Care*, vol. 13, no. SUPPL. 2, pp. 22–27, 2004. [Online]. Available: www.qshc.com

[136] A. Wiedemann, N. Forsgren, M. Wiesche, H. Gewald, and H. Krcmar, "Research for practice: The Devops phenomenon," *Communications of the ACM*, vol. 62, no. 8, pp. 44–49, 2019.

[137] M. Outlaw, "The DevOps Handbook – The Technical Practices of Flow," 2020. [Online]. Available: https://www.codingblocks.net/podcast/the-devops-handbook-the-technical-practices-of-feedback/

[138] D. Cukier, "DevOps patterns to scale web applications using cloud services," in *SPLASH 2013 - Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, and Applications: Software for Humanity*. New York, New York, USA: ACM Press, 2013, pp. 143–152. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508075.2508432

[139] Veritis, C. Services, and Services, "Measuring DevOps: Key 'Metrics' and 'KPIs' That Drive Success!" pp. 1–10, 2020. [Online]. Available: https://www.veritis.com/blog/measuring-devops-key-metrics-and-kpis-that-drive-success/

[140] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of DevSecOps," in *5th International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2019*. Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway: Institute of Electrical and Electronics Engineers Inc., jun 2019, pp. 1–8.

[141] S. I. Mohamed, "DevOps Maturity Calculator DOMC -Value oriented approach," *International Journal of Engineering Research and Science*, vol. 2, no. 2, pp. 2395–6992, 2016.

[142] P. Waterhouse, "DevOps Practitioner Series - Metrics That Matter," 2015. [Online]. Available: https://docs.broadcom.com/doc/devops-practitioner-series-metrics-that-matter-developing-and-tracking-key-indicators

[143] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in Software Engineering: A systematic literature review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 860–878, 2008.

[144] J. M. Verner, M. A. Babar, N. Cerpa, T. Hall, and S. Beecham, "Factors that motivate software engineering teams: A four country empirical study," *Journal of Systems and Software*, vol. 92, no. 1, pp. 115–127, 2014.

©2022 IEEE. Reprinted, with permission, from Amaro, R., Pereira, R. and Mira da Silva, M., "Capabilities and Practices in DevOps: A Multivocal Literature Review", IEEE Transactions on Software Engineering, April 2022

[145] S. Smith, "High performing DevOps metrics," pp. 1–11, 2020. [Online]. Available: https://samlearnsazure.blog/2020/04/30/high-performing-devops-metrics/

[146] V. Gupta, P. K. Kapur, and D. Kumar, "Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling," *Information and Software Technology*, vol. 92, no. 1, pp. 75–91, 2017.

[147] RDC Partner, "6 core capabilities of DevOps practice," pp. 2–6, 2018. [Online]. Available: https://rdcpartner.nl/6-core-capabilities-of-devops- practice/

[148] C. Rudder, "10 ways DevOps helps digital transformation | The Enterprisers Project," 2019. [Online]. Available: https://enterprisersproject.com/article/2019/8/devops-role-digital-transformation

[149] S. Yusuf, "Ebbs and Flows Of DevOps Debugging PART 1 - Security Boulevard," pp. 1–32, 2021. [Online]. Available: https://securityboulevard.com/2021/02/ebbs-and-flows-of- devops-debugging-part-1/

[150] Puppet, "5 Foundational DevOps Practices: How to Establish & Build on Them | Puppet," 2018. [Online]. Available: https://puppet.com/resources/whitepaper/5-foundational-devops-practices-how-establish-build-them/

[151] T. Jachja, "CI/CD Patterns and Practices — DevOps Institute," 2020. [Online]. Available: https://devopsinstitute.com/ci-cd-patterns-and-practices/

[152] C. Patel, "DevOps Best Practices - DZone DevOpsdzone.com," 2020. [Online]. Available: https://dzone.com/articles/devops-best-practices

[153] R. Powell, "Essential DevOps Principles for 2021 | CircleCIcircleci.com," 2020. [Online]. Available: https://circleci.com/blog/essential-devops-principles/

[154] M. Patil, "Maximizing DevOps ROI with Modern Application Practices | HCL Blogs," 2020. [Online]. Available: https://www.hcltech.com/blogs/maximizing-devops-roi-modern-application-practices

[155] D. Online, "The Practice of DevOps," 2020. [Online]. Available: https://www.dqindia.com/the-practice-of-devops/

[156] C. Solutions, "8 Best Practices for Successful Implementation of DevOps," 2019. [Online]. Available: https://dev.to/credencys/8-best-practices-for-successful-implementation-of-devops-in-your-enterprise-2k93

[157] G. Pousseo, "DevOps Practices: Agility without DevOps is pointless," 2019. [Online]. Available: https://www.pentalog.com/blog/agility-without-devops-is-pointless

[158] A. D. Rayome, "5 foundational DevOps practices your enterprise needs to succeed - TechRepublic," 2018. [Online]. Available: https://www.techrepublic.com/article/5-foundational-devops-practices-your-enterprise-needs-to-succeed/

[159] P. CIO, "Building a Strong DevOps Foundation | CIO," 2018. [Online]. Available: https://www.cio.com/article/3319077/building-a-strong-devops-foundation.html

[160] N. Stamenkovic, "DevOps - principles, practices and why should you care? - Ingsoftware Blog," pp. 1–11, 2018. [Online]. Available: https://www.ingsoftware.com/devops-principles-and- practices

[161] Pinkelephant, "DevOps: Why The 15 Essential Practices Are Key To Your Organization's Survival | Pink Elephant Blog," 2018. [Online]. Available: https://blog.pinkelephant.com/blog/devops-why-the-15-essential-practices-are-key-to-your-organizations-surviva

[162] PuppetSplunk, "The 5 foundational devops practices," 2018. [Online]. Available: https://www.1sttechguide.com/wp-content/uploads/2019/09/The-5-Foundational-DevOps-Practices.pdf

[163] A. Crouch, A. Gaspari, and A. Crouch, "6 Steps to a Successful DevOps Adoption | AgileConnection," 2017. [Online]. Available: https://www.agileconnection.com/article/6-steps-successful-devops-adoption

[164] C. Pang, A. Hindle, and IEEE, "Continuous Maintenance," in *Proceedings - 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016*, no. 32nd IEEE International Conference on Software Maintenance and Evolution (ICSME). Department of Computing Science, University of Alberta, Edmonton, AB, Canada: Institute of Electrical and Electronics Engineers Inc., oct 2016, pp. 458–462.

[165] A. Wadhera, "10 Key DevOps Practices to Improve IT Efficiency," 2016. [Online]. Available: https://www.tothenew.com/blog/10-key-devops-practices-to-improve-it-efficiency/

[166] L. Zhu, L. Bass, and G. Champlin-Scharff, "DevOps and Its Practices," *IEEE Software*, vol. 33, no. 3, pp. 32–34, may 2016.

[167] K. Horvath, "DevOps Part 2: Methods, Practices and Tools," 2015. [Online]. Available: https://content.intland.com/blog/agile/devops/devops-part-2-methods-practices-and-tools

[168] M. Croker, "DevOps: innovative engineering practices for continuous software delivery," 2015. [Online]. Available: https://www.accenture.com/_acnmedia/PDF-18/Accenture-DevOps-brochure-new.pdf

[169] M. A. Silva, J. P. Faustino, R. Pereira, M. M. da Silva, and M. Mira da Silva, "Productivity gains of DevOps adoption in an IT team: A case study," in *Proceedings of the 27th International Conference on Information Systems Development: Designing Digitalization, ISD 2018*, B. C. L. M. L. H. S. C. Andersson B. Johansson B., Ed. Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal: Association for Information Systems, jul 2018, p. 12. [Online]. Available: https://repositorio.iscte-iul.pt/handle/10071/16388

[170] M. Rother, *Toyota Kata: Managing people for improvement, adaptiveness and superior results*. MGH, New York, 2019.

[171] Cambridge, "CAPABILITY | meaning in the Cambridge English Dictionary," 2021. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/ capability

[172] Merriam-Webster, "Capability | Definition of Capability by Merriam-Webster," 2021. [Online]. Available: https://www.merriam-webster.com/dictionary/capability

[173] T. Biggs, M. Shah, and P. Srivastava, *Technological capabilities and learning in African enterprises*. The World Bank, 1995.

[174] Cambridge, "PRACTICE | meaning in the Cambridge English Dictionary," 2021. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/ practice

[175] "Practice | Definition of Practice by Merriam-Webster," 2021. [Online]. Available: https://www.merriam-webster.com/dictionary/practice

[176] IEEE Standards Association, "IEEE Standard for Configuration Management in Systems and Software Engineering: IEEE Std 828™-2012 (Revision of IEEE Std 828-2005)," IEEE Standards Association, Tech. Rep., 2012.

CHAPTER 3

Article #2

This article (A2) presents the second MLR, which led to finding and categorizing 22 main DevOps metrics, providing definitions, importance, and practical implementation guidelines to enhance DevOps adoption and performance in organizations [91]. The second analysis vector of this thesis is related to DevOps metrics, which results in an informed approach to improving DevOps adoption in a structured way. Therefore, it aims to assist researchers and practitioners in understanding and implementing DevOps metrics.

As a result, the article provides relevant DevOps metrics organized in KPIs to facilitate the efficiency of DevOps adoption and better assess DevOps performance in organizations. The results also indicate that by conducting an extensive MLR on DevOps metrics, it resulted in their definitions, importance, and categorization. The results also highlight the need for DevOps metrics to improve software delivery, performance and discuss several aspects, like business improvements and challenges. It guides how to implement DevOps metrics in organizations by establishing a baseline, setting targets and goals, automating data collection, releasing changes, using dashboards and reports, taking action, continuously monitoring, and using a feedback loop for continuous improvements. The outcomes assist researchers and practitioners in understanding and implementing DevOps metrics to improve profitability, productivity, quality, operational efficiency, customer feedback, and achieving organizational goals.

Article details:

- Title: DevOps Metrics and KPIs: A Multivocal Literature Review
- Date: March 2024
- Journal: ACM Computing Surveys
- Scimago Journal Rank: Q1
- Publisher: Association for Computing Machinery (ACM)

DevOps Metrics and KPIs: A Multivocal Literature Review

RICARDO AMARO, Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
RÚBEN PEREIRA, INOV INESC Inovação, Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
MIGUEL MIRA DA SILVA, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

Context: Information Technology organizations are aiming to implement DevOps capabilities to fulfill market, customer, and internal needs. While many are successful with DevOps implementation, others still have difficulty measuring DevOps success in their organization. As a result, the effectiveness of assessing DevOps remains erratic. This emphasizes the need to withstand management in measuring the implementation process with suitable DevOps Metrics. But what are these metrics?

Objective: This research seeks to provide relevant DevOps Metrics to facilitate the efficiency of DevOps adoption and better analyze DevOps performance in enterprises.

Method: A Multivocal Literature Review (MLR) is conducted, with 139 documents gathered and thoroughly examined from throughout the community, including books, scientific articles, white papers, conferences, among others.

Results: This article conducts an extensive and rigorous MLR, contributing with a definition of DevOps Metrics, 22 main metrics, their definitions, importance, and categorization in sets of Key Performance Indicators, as well as exposing clear indicators on how to improve them. It is also discussed how metrics could be put into practice and what constitutes a change in the context of DevOps Metrics. The study's outcomes will assist researchers and practitioners understand DevOps Metrics and how to better implement them.

CCS Concepts: • **Software and its engineering** → *Software creation and management*; *Software development process management*; *Programming teams*; *Software post-development issues;*

Additional Key Words and Phrases: DevOps, metrics, performance, adoption, software development life cycle, information system

ACM Reference Format:
Ricardo Amaro, Rúben Pereira, and Miguel Mira da Silva. 2024. DevOps Metrics and KPIs: A Multivocal Literature Review. *ACM Comput. Surv.* 56, 9, Article 231 (April 2024), 41 pages. https://doi.org/10.1145/3652508

1 INTRODUCTION

Information technology (IT) organizations are constantly impacted by ever-changing consumer expectations, industry regulations, competitors, and advanced external threats [58, 155]. Consequently, in those organizations where software development is part of the core business, they seek a competitive advantage [101], such as improving the user experience, increasing productivity, and team collaboration [80, 93].

Authors' addresses: R. Amaro and R. Pereira, Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal; e-mails: ricardo_amaro@iscte-iul.pt, ruben.filipe.pereira@iscte-iul.pt; M. Mira da Silva, Instituto Superior Técnico, Universidade de Lisboa, Portugal; e-mail: mms@tecnico.ulisboa.pt.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

© 2024 Copyright held by the owner/author(s).
ACM 0360-0300/2024/04-ART231
https://doi.org/10.1145/3652508

However, because of the silos that exist between development and operations [212], this approach produces complexity and inefficiency. The request for more frequent software delivery, in the absence of sustained builds, adequate testing, and release automation [56], causes burnout and toil in the engineers doing operations, deteriorating software delivery performance and reliability.

As a result of these challenges, we observe the rise of DevOps, an organizational model that emphasizes empathy and fosters more cooperation among technical teams involved in software delivery [80], to improve key performance indicators like development time, deployment rate, reliability, mean time to recover and the overall cost of product implementation and deployment [146]. Furthermore, research has been specifically focusing on **Mean Time To Recover (MTTR)**, **Mean Lead-time for Changes (MLT)**, **Deployment Frequency (DF)**, and **Change Failure Rate (CFR)**, four key metrics [53].

But it is still common to observe inconsistent results in the adoption of DevOps practices and capabilities [6, 172, 177], confirming the need to have a consensual improvement on expanding the metrics being gathered and used to improve those results. It is agreed that DevOps, with the right metrics, can help applications and teams perform at their best [84] and should also present relevant data, clearly and understandably, showing where improvements can be made in the deployment and change process [118].

Moreover, it is a pertinent approach to regulate and evaluate the level of success of DevOps adoption by using precise DevOps Metrics that assess the success of DevOps capabilities adoption, indicating, for instance, if a certain software delivery process within the pipeline [155] is at optimal levels or might need enhancements. This information will support management's decision-making process to have a clear vision of the steps to take ahead based on information systems and metrics to increase efficiency [101].

Therefore, it would be valuable to comprehend the DevOps Metrics that comprise the DevOps assessment process, as well as the main key metrics required to carry out the adoption of DevOps capabilities[5, 6]. From previous related work, it is understood that DevOps Metrics help drive outcomes [38], generated by DevOps capabilities and controlled by the usage of a periodic DevOps assessment [35]. Some authors already imply a few key concepts for arranging the DevOps Metrics like *Organizational Culture* [53, 84, 103, 118, 152], *Operational Performance* [23, 84, 109, 165], *Business Focus* [73, 85, 87, 114, 135] and *Incremental Change* [76, 186, 217]. This study also proposes discussing these concepts to categorize DevOps metrics in Section 5. This may be useful for practitioners and organizations, since it will improve comprehension and subsequently improve the success of DevOps implementation tightly connected with Outcomes [71, 93, 109, 148]. Especially if it would be possible to extract the Key Performance Indicators out of all the discussed DevOps Metrics in literature, that can help define the organization's DevOps strategy and clear focus.

Hence, the success of DevOps could be improved if the main DevOps Metrics are known, targeting a successful implementation and operationalization of the complex [92] process of DevOps. Not just during the implementation phase but also later in controlling the execution, since it demands a rigorous systematization for self-assessment. This, in turn, should lead to increased performance levels, according to Ravichandran et al. [155]. Unfortunately, there is a lack of a broader study, joining practitioners and researchers knowledge, proposing an extended list of metrics, their clear categorization, and relating them to outcomes for the effective use of DevOps in organizations.

Consequently, the *research problem* is the following: While there is important research done in DevOps over the years around a few measures, the assessment of the desired success of DevOps adoption is still unpredictable, because, despite the fact that metrics are being discussed in the industry, there is still a need for more consensus regarding definitions and benefits, due to the lack of a broader study that synthesizes and clarifies the main key metrics from both practitioners and researchers.

This article proposes to understand and synthesize the main DevOps Metrics that are mentioned in publications and how they relate to each other. Because this subject has received more attention and scrutiny from the industry than from the scientific community, with major technological organizations issuing frequent reports, this article suggests undertaking a MLR [63], to expand the few academic papers and include voices from the industry, to research and elicit the main DevOps Metrics that are mentioned in publications from both the practitioners and researchers communities. It is also intended to gather the definition and categorization of each of these DevOps Metrics. Based on the primary objective of this research, a MLR is undertaken to look for scientific and "gray" literature that discusses or examines the subject of DevOps Metrics, which may then be translated into the following research questions:

— **RQ1**. What are the main DevOps Metrics, and where are they referenced?
— **RQ2**. What is the precise definition and importance of each main DevOps Metric?

This article also provides business leaders with a concise DevOps metrics definition in Section 5.1, a categorization in Section 6 and a discussion of findings of the research, aiming to improve the adoption of DevOps, facilitating an alignment strategy with business goals. Thus, helping on data-driven decisions[5, 92], for enhancing change[76, 186, 217], operations [23, 84, 109, 165], and promoting a culture of continuous improvement.

The remainder of this article is organized as follows: Section 2 presents a review of the core concepts of DevOps, its collaborative culture, practices, and the importance of measuring its impact on software delivery performance. Section 3 discusses MLR methodology to investigate DevOps metrics from the academic and gray literature and analyze them. Section 4 outlines the steps in the literature review process to identify and analyze relevant studies on DevOps Metrics and **Key Performance Indicators (KPIs)**, culminating in 139 publications for data extraction. Section 5 addresses the research questions and provides a detailed analysis of key DevOps metrics and KPIs, their relevance, expected trends, and improvement strategies. Finally, the discussion of our findings is located in Section 6, where we also provide a categorization of DevOps metrics, impact on software delivery, and their practical implementation challenges. The article ends with the conclusion, future work, and limitations in Section 7.

2 DEVOPS

This section provides a theoretical foundation for the study area of this research, namely, *DevOps*.

DevOps is an abbreviation for the **Developer (Dev)** and **Operations (Ops)** teams, which collaborate to eliminate the so-called engineering silos [162, 217]. There is no universal definition of DevOps. Blog articles on the subject are widespread, although they usually vary on a specific definition of the phrase. DevOps emerged as an evolution of the agile paradigm for IT service management [32], which focused on developing new processes for the continuous deployment of rapidly changing software. According to Jabbari et al. [85], DevOps is a development approach that emphasizes communication and cooperation, continuous integration, quality assurance, and delivery with automated deployment through the use of a set of development techniques. It can also be seen as a conceptual framework that is based on certain capabilities, focused on the acronym CAMS (Culture, Automation, Measurement, and Sharing) [81]. Later, Jez Humble added to these four pillars, the **Lean (L)** pillar, becoming the acronym **Culture, Automation, Lean principles, Measuring and Sharing (CALMS)** [215].

DevOps uses a combination of cultural changes and technology-enabled strategies to achieve higher levels of throughput and stability, even in the face of high unpredictability [81, 213]. Sousa et al. [182], in his article on DevOps foundations and views, emphasizes the new approach to software delivery that happens through cooperation between development teams and operations,

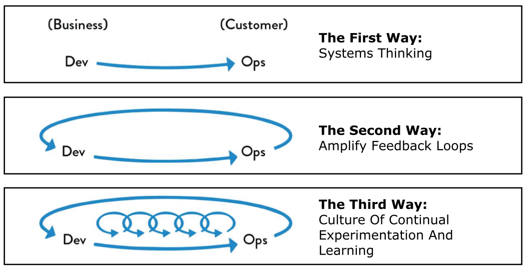


Fig. 1. The Three Ways: The principles underpinning DevOps (adapted) [65].

as demonstrated in Figure 1, rather than the conventional method that is isolated in organizational silos. This trait of effective interaction between IT Development and IT Operations teams is critical for ensuring successful IT system deployment and operation [193].

DevOps is also a culture, movement, or practice that stresses cooperation and communication [13] with the objective of speeding up the software release cycle to production and automating the creation of new software components while maintaining high quality, as mentioned by Lwakatare et al. [110], where a literature review on the term DevOps indicates that DevOps is a mentality shift backed by a set of automated procedures.

For Riungu-Kalliosaari et al. [159], DevOps is a collection of techniques aimed at reducing the time it takes for a change made to a system to move into regular production, while guaranteeing high quality and the least friction and blame between teams rather than trust and empathy. A comprehensive DevOps performance research book, "Accelerate" [53], investigates how most modern organizations are using DevOps principles and practices, using statistical methods to assess software delivery performance and providing a new understanding of software delivery and organizational performance.

Finally, as mentioned by Forsgren et al. [56], There are two approaches for collecting metrics about DevOps performance: Survey data and System data. Both have advantages and disadvantages, but knowing what metrics to collect first has been described by several authors [48, 53, 58, 141, 192] as a primary goal for determining the effectiveness of DevOps implementation within organizations [150, 155]. However, there has been minimal academic research on measuring DevOps capabilities and practices [6, 103].

3 MULTIVOCAL LITERATURE REVIEW

A MLR is a type of **Systematic Literature Review (SLR)** [62], designed to include gray literature such as blogs, videos and websites, as well as white papers constantly produced by SE practicing professionals outside academic forums, despite publishing (peer-reviewed) writings such as journals and conference papers. MLRs are therefore useful for research extension by integrating material that would ordinarily not be collected because of its "gray" character [63], as shown in the Figure 2.

While examining the specific subject of DevOps, a number of academics have already noticed that "enlarging the scope" and including **Gray Literature (GL)** will add value and advantage to the review study. There are already some successful MLR-based DevOps research examples in the same area [59, 141]. Therefore, it confirms the practical application of this approach in the research presented and increases the diversity of accessible sources in many ways, including the representation of various goals and viewpoints [129].

The MLR has a few objectives for this study, including a thorough mapping of the main DevOps Metrics from scientific and gray literature, as well as a summary of the definitions and references of each metric.

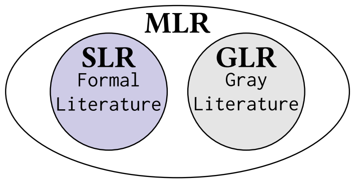


Fig. 2. Relationship of SLR, GLR, and MLR [63].

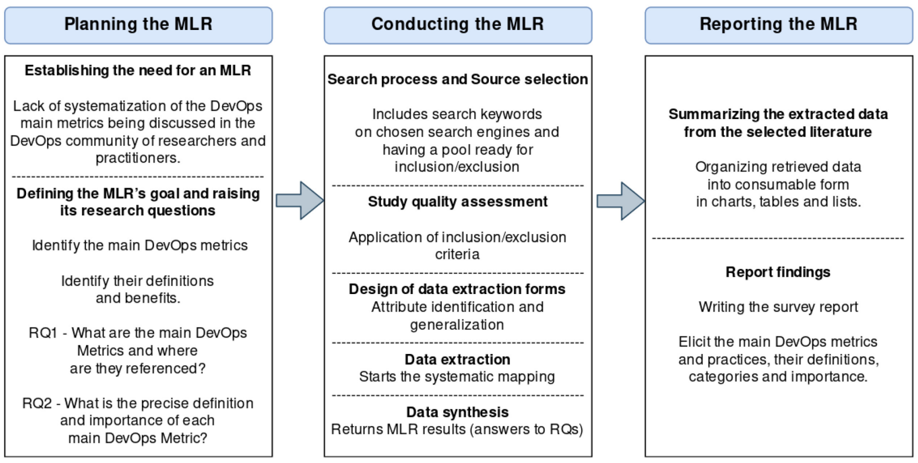

Planning the MLR
Establishing the need for an MLR
Lack of systematization of the DevOps main metrics being discussed in the DevOps community of researchers and practitioners.
Defining the MLR's goal and raising its research questions
Identify the main DevOps metrics
Identify their definitions and benefits.
RQ1 - What are the main DevOps Metrics and where are they referenced?
RQ2 - What is the precise definition and importance of each main DevOps Metric?
Conducting the MLR
Search process and Source selection
Includes search keywords on chosen search engines and having a pool ready for inclusion/exclusion
Study quality assessment
Application of inclusion/exclusion criteria
Design of data extraction forms
Attribute identification and generalization
Data extraction
Starts the systematic mapping
Data synthesis
Returns MLR results (answers to RQs)
Reporting the MLR
Summarizing the extracted data from the selected literature
Organizing retrieved data into consumable form in charts, tables and lists.
Report findings
Writing the survey report
Elicit the main DevOps metrics and practices, their definitions, categories and importance.

Fig. 3. MLR steps based on Garousi et al. [63].

There are several guidelines in the scientific literature for doing SLR research in Software Engineering [11, 95, 96]. MLRs, however, has multiple stages that are distinct from regular SLRs. Specifically, the process of investigating and evaluating the source's quality. Therefore, SLR guidelines are only partially useful for conducting MLR studies as seen in Figure Figure 3. This process shows the planning, conducting, and reporting as proposed by Garousi et al. [63].

While this strategy is predicted to generate substantial information in some areas of DevOps research, using such data will surely pose challenges, as the evidence provided is typically based on experience and opinion. For that reason, **systematic guidelines** [62] will be used for performing this MLR.

3.1 Planning

3.1.1 Motivation. Management in software development businesses that wish to adopt DevOps internally must have access to appropriate supporting information and metrics regarding this implementation to assess the success and enhance the efficiency [101] of applying DevOps [13, 87, 109, 114, 135, 141, 155, 159].

To achieve targets and goals, organizations adopting DevOps can then quickly measure and demonstrate the effectiveness of new DevOps processes like software development and continuous delivery, align cross-functional teams around business value creation and continuous improvement [76], pinpoint capability gaps and initiate remediation strategies. It is hard to improve DevOps without metrics [56, 141, 179]. An organization must constantly qualify its DevOps processes, and achieving excellence requires measurement to remove subjectivity.

However, there exists a lack of systematization of the main DevOps Metrics being debated in the DevOps community of scholars and practitioners. To provide systematization and clarity to the current DevOps Metrics, a broader range of sources, including practitioners and industry perspectives, must be collected. With the inclusion of gray literature to review, a comprehensive survey on

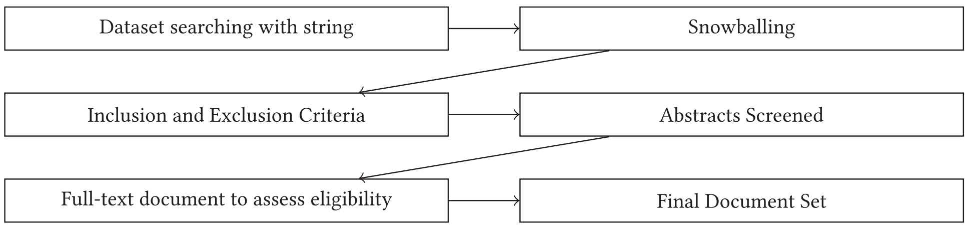

Dataset searching with string
Snowballing
Inclusion and Exclusion Criteria
Abstracts Screened
Full-text document to assess eligibility
Final Document Set

Fig. 4. Review protocol performed in this research.

not only what the scientific literature specifies about Metrics but also what the industry produces dynamically and utilizes internally, can be undertaken. Combining both points of view will help to improve the research topics given in Section 1.

3.1.2 Review Protocol. The first batch of papers has been procured as seen in Figure 4 following the completion of the search and snowballing, inclusion and exclusion criteria will be used to refine the search results in the first phase.

The review protocol used the workflow shown in Figure 4.

In January 2022, a publication search using various keywords was conducted, in an attempt to locate further studies relevant to this research that may provide answers to the indicated research questions. This section lists the search string used to get the most studies and datasets.

- **Search String**: `(devops AND (metrics OR measures OR kpi OR indicator))`.
- **Datasets**: The search engines used were, Google search, Scopus, Web of Science, IEEE, ACM and EBSCO.

To make finding and gathering large amounts of gray literature easier, some code was written, as shown in the source code in Appendix A (Python code for fetching Google search results), to parse the data into CSV files [120]. This way, it is ensured that clean results are not specific to the researcher, but rather reproducible for peer review. Thus, addresses the issue of consistency in the returned results, since Google delivers customized results that are tailored differently for different users based on their previous search history and preferences. Finally, it simplifies the job of converting the results into spreadsheet files that are used in the MLR process.

In this MLR inclusion and exclusion criteria focused on both gray literature and quality peer reviewed work reporting work found on DevOps Metrics. For this focus, it is developed the following **inclusion criteria** present in Table 1. Criteria 1 and 2 have the goal of ensuring focus on relevant quality publications. Criteria 3 and 4 were also used to assess evidence of quality and report on the area of this study. **Exclusion criteria** also reflect on the transparency of quality relevant work that contributes to DevOps Metrics discussion. We excluded papers with the following features. Criteria 1–3 exclude non-relevant work for this study. Criteria 4–8 exclude incomplete publications, out of boundaries, are lacking identification or not written in English. The inclusion and exclusion criteria used in Table 1 has been adapted from authors who already approached the use of similar methods [62, 63, 126, 129, 161, 170].

Within this scope, abstracts are screened to determine their relevance to the investigation. After which, the relevant articles are reviewed to arrive at the final study selection for the coding review.

4 CONDUCTING THE MLR

4.1 Selection of Studies

For reference, the complete summary of the review process is shown in the diagram in Figure 5 with a visual representation of the applied MLR selection process. This reflects all the selection work done through the methodical process of MLR.

Table 1. Inclusion and Exclusion Criteria Applied in This Research

Inclusion Criteria	Exclusion Criteria
1. Full publications from databases and snowballing: (a) Including full books, webpages, or papers. (b) Related research (qualitative and quantitative). (c) Practitioners or industry-relevant contributions. 2. Publications matching the search string and date: (a) Title and abstract of peer-reviewed work. (b) Full content of gray literature. (c) Date between 2010 and January 2022. 3. Explicitly stated and described DevOps Metrics subject. 4. Clearly described Metrics related to DevOps performance.	1. Does not contribute to answering any research questions: (a) Does not elicit, discuss and list DevOps Metrics. (b) Not focused on DevOps. (c) Only focused on Agile. (d) Focused on IoT systems, not DevOps Metrics. (e) Focused on monitoring systems, not DevOps Metrics. (f) Focused on microservices, not DevOps Metrics. (g) Focused on Quality Assurance, not DevOps Metrics. 2. Advertisement, Product promotion or Job Post. 3. Conference Announcement, Review or Summary. 4. Full-text not accessible. 5. Published before 2010. 6. No publication date. 7. Unidentified author. 8. Not Written in English.

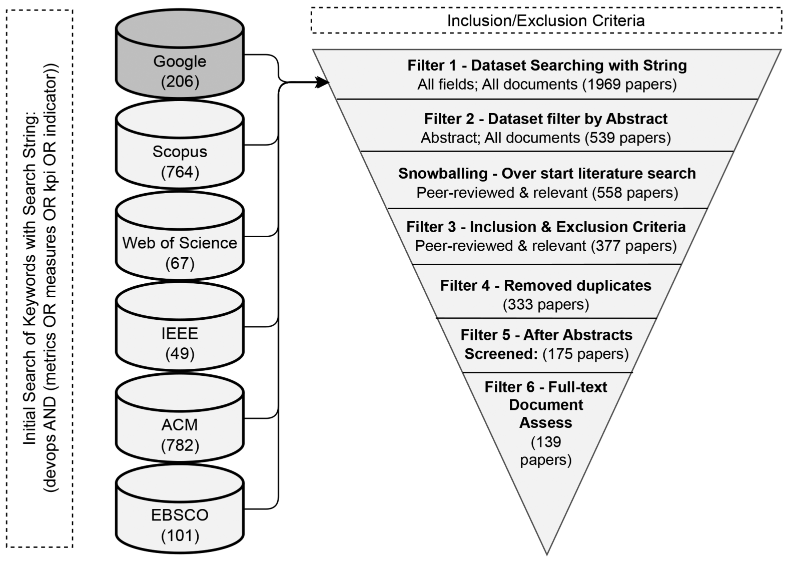

Initial Search of Keywords with Search String: (devops AND (metrics OR measures OR kpi OR indicator))
Google (206)
Scopus (764)
Web of Science (67)
IEEE (49)
ACM (782)
EBSCO (101)
Inclusion/Exclusion Criteria
Filter 1 - Dataset Searching with String All fields; All documents (1969 papers)
Filter 2 - Dataset filter by Abstract Abstract; All documents (539 papers)
Snowballing - Over start literature search Peer-reviewed & relevant (558 papers)
Filter 3 - Inclusion & Exclusion Criteria Peer-reviewed & relevant (377 papers)
Filter 4 - Removed duplicates (333 papers)
Filter 5 - After Abstracts Screened: (175 papers)
Filter 6 - Full-text Document Assess (139 papers)

Fig. 5. Followed multivocal literature review process (adapted) [63].

In the first phase of the search, *filter 1* (All fields; All documents) was combined with the search term, both of which were found in Table 2. The difference between *filter 1* and *filter 2* is justified by the fact that previously, keywords could be found throughout the retrieved material, and some search engines return more literature than academic articles, such as newspapers or reports. In the case of the Google search engine, however, this is not the case. Thus, the results stay the same. In the second iteration of search results, *filter 2* (Abstracts, All papers) was utilized and therefore the number of articles that include an abstract mentioning the keywords was reduced to 539 publications in total. For gray literature, it is always considered the entire text, since there is no abstract. The resulting articles were added to Zotero[1] reference manager.

[1]https://www.zotero.org

Table 2. Filters Used in the MLR Protocol

Database	Filter 1	Filter 2	Snowballing	Filter 3	Filter 4	Filter 5	Filter 6
Google	206	206		170	165	132	127
Scopus	764	140		87	69	11	2
Web Of Science	67	61	+19	46	38	9	2
IEEE	49	38		35	25	8	3
ACM	782	28		19	17	7	3
EBSCO	101	66		21	19	8	2
Total	1,969	539	558	377	333	175	139

Filter 1 = Query All fields, All documents.
Filter 2 = Query Abstracts, All documents.
Snowballing = Applied over starting literature search [63].
Filter 3 = Relevant (inclusion/exclusion criteria).
Filter 4 = Remove duplicates.
Filter 5 = After Abstracts Screened.
Filter 6 = Full-text Document Assess.

In the following stage, forward and backwards *snowballing* [216] is performed, taking as seed all relevant conference papers, scientific articles, and the 2019 State of DevOps Report, which appears in Google search when using the search string. The motivation for snowballing was to increase the number of relevant quality papers. This process resulted in an increase in the overall quantity and quality of articles to 19 more relevant publications identified. These papers were added to the first row in Table 2.

We remain with 377 publications after applying inclusion and exclusion criteria *filter 3*, present in Table 1. This leads to *filter 4*, which is defined to eliminate duplicates from the list of results. Since Zotero already detects duplicate documents, while keeping the tags of where the document came from, the effort of counting the duplicates done from *filter 3* to *4* is reduced due to this capability. Nevertheless, during manual scanning, a few duplicates were still found, therefore the total number of publications with no duplicates is only accounted for in *filter 4* as a consolidated number in the process. In *filter 5* after abstracts are screened, 175 documents remain. Because there is no abstract for instances pertaining to gray literature, the entire text was skimmed, allowing a better assertion of the quality of the publications. After screening all full-text documents, 139 publications are left for the extraction phase of the MLR.

4.2 Data Extraction Analysis

Following the selection of the final collection of publications, this section analyzes the various components of the search results in connection to the final set of articles based on the source data. This study is based on an evaluation of the whole text of 139 articles that are appropriate for extracting significant data for this research. Additionally, a summary of the years and genres of articles selected for thorough reading is included.

4.2.1 Gray and White Literature Number of Contributions. The relationship of the final document set by database shows that 127 results originated from Google, with 91,37 percent of gray literature. Six publications are contributed by Web of Science, Scopus, and EBSCO databases. ACM and IEEE each provided three items, leading to a total of six publications of relevant research peer-reviewed articles. This validates the expectation that the practitioner community will produce a wider range of findings when contrasted with the scientific literature.

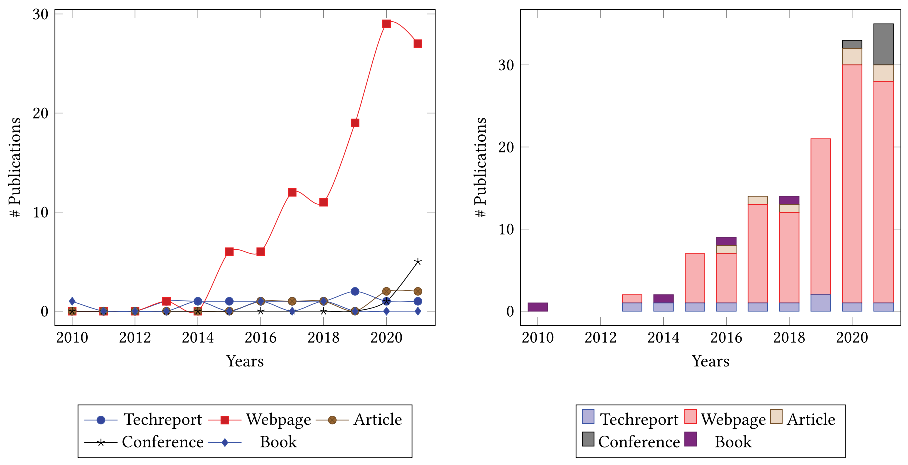


Fig. 6. Distribution of publications per type over the years.

4.2.2 Distribution of Publications Over the Years. In Figure 6 it is reflected how publications have been evolving over the years with the biggest amount of generated literature appearing in webpages in the year 2020, where three of which include relevant video content. The rising number in 2020 and 2021 reflects the fact that this topic has been gaining more relevance. Another interesting aspect to note is the early appearance in 2010 of the book “Continuous Delivery” [80] mentions that metrics help improvements and efficiency in the continuous integration and delivery processes. According to Jezz Humble, a well-implemented deployment pipeline should make determining the Cycle Time simple. It should also display how long it takes from check-in to each stage of the procedure. This is an effective method for identifying bottlenecks in operations.

Since 2013, the same author has been consistently contributing to the topic, including in numerous State of DevOps reports [35, 144–149, 151] and in other relevant books. These important contributions to DevOps Metrics are also shared with Joanne Molesky and Barry O’Reilly in “Lean enterprise : continuous delivery, DevOps, and lean startup at scale” [82] Gene Kim, Patrick Debois and John Willis in the “The DevOps Handbook” [93] and congregating efforts with Nicole Forsgren in “Accelerate: The science of lean software and DevOps” [53] where important metrics are discussed based on the investigation done in yearly *State of DevOps Reports* [144–148], which have been consistent since 2013. In Figure 6 it is also observed an increasing interest in the sector in the last years, despite a relatively low amount of research work done on studying metrics, demonstrating the potential appeal and utility of this research in the field.

The Tech reports from 2013 and 2014 had a special importance in ramping up the interest on DevOps Metrics topic and raising awareness for the fact that measurements are visible and actionable [144, 145], while only focusing on four top key metrics as seen in Section 5.1.1. Thus, making this MLR research important to expand the main metrics to a list that is still useful, broader and manageable. Finally, the number of gray literature articles increased considerably in 2020, as demonstrated by the massive increase in web pages related content in that year, showing that practitioner writings grew far faster than academic research. Later in 2021, we see a growth in conference publications that address the DevOps Metrics and KPIs topic. The research in this survey tries to contribute to this growth by congregating different voices in a MLR.

5 REPORTING THE MLR

The reporting is taken out from qualitative coding done with the selected publications using Qualcoder[2] OpenSource tool. Qualitative coding [167] is a process of organizing and categorizing data

[2]https://qualcoder.wordpress.com/

Table 3. Definition of DevOps Metric from Literature

A **DevOps Metric** is here defined as a quantifiable, business-relevant, trustworthy, actionable and traceable [42, 49, 107, 118, 202] indicator that aids organizations in making data-driven decisions to continuously improve their DevOps and software delivery process [16, 33, 55, 72, 78, 89, 97, 98, 106, 119, 173, 180, 194].

collected in qualitative research. For this study, emerging codes were applied using repeating ideas and metrics found from all selected publication texts to organize the data into meaningful categories. After that, the data was analyzed manually within Qualcoder. Using the resulting coded data, and also taking into account the conceptual framework proposed in Section 1, patterns and relationships were identified. In conclusion, qualitative coding, characterized by deep data scrutiny and pattern recognition, is crucial in revealing the presented findings, all stemming from the analysis of qualitative data.

From the extensive Multivocal Literature Review done, it is understood that the main DevOps Metrics should aim to quantify the right elements to understand if a DevOps process is working [72, 140, 204]. It was also seen in Section 5.2 that authors distinguish five qualities of good DevOps key performance indicators to clearly define their usefulness. This is important to know so that DevOps adoption can be measured toward success. While metrics and KPIs are frequently used by authors interchangeably, the distinction is clear: In the context of DevOps [84], KPIs are a set of the measures or indicators that have the greatest impact on an organization's DevOps progress [115, 116], thus the reason this study uses this concept. They articulate and provide insight into the metrics and outcomes that the organization must track and achieve to accomplish long-term goals. Key Performance Indicators (KPIs) are metrics that help understand how an entity is doing against its objectives [46, 84, 118]. These kinds of metrics are fundamental to leverage the rigor of measurement, not only in DevOps but also in Software Engineering and Information Systems [101]. This MLR benefited from the fact that more than half the publications (88) mention metrics and try to organize and explain each stated metric, as seen in Table 6. There are 49 publications that also try to define what are DevOps Metrics. Following those dispersed definitions, this MLR can now propose a unified definition of DevOps Metrics in Table 3.

5.1 RQ1—What Are the Main DevOps Metrics and Where Are They Referenced?

5.1.1 Main Metrics Found Over the Years. In this study, 22 main DevOps Metrics seen in Figure 7 are set for discussion. These metrics are listed from M01 to M22, within the following figure, in descending order, by the total number of references, from the publications accounted for in Section 4.1.

It can also be observed in Figure 7 how the various metrics have grown in the literature over the years.

Since this MLR found 10 years with relevant publications out of the 12 years, it was chosen accordingly to use the metrics that are cited 10 or more times as the main DevOps metrics for this research, present in Figure 7. Therefore, in Figure 7 are shown the top 22 metrics, from M01 to M22. In this figure, a significant jump in 2020 can be perceived, namely, in *MTTR*, *MLT*, *DF*, and *CFR*. These four metrics were the most frequently stated, indicating that practitioners have agreed on their higher importance based on the articles reviewed. If we compare MTTR (114 mentions) with CFR (82 mentions), there is only and a difference of 32 mentions, while if we compare MTTR with *Service Availability and Uptime* (42 mentions), there is a notable difference of 72 mentions. Therefore, while each of these 22 metrics have 10 or more mentions, there is a clear tendency of increased interest in M01, M02, M03, and M04, widespread and driven largely by the research

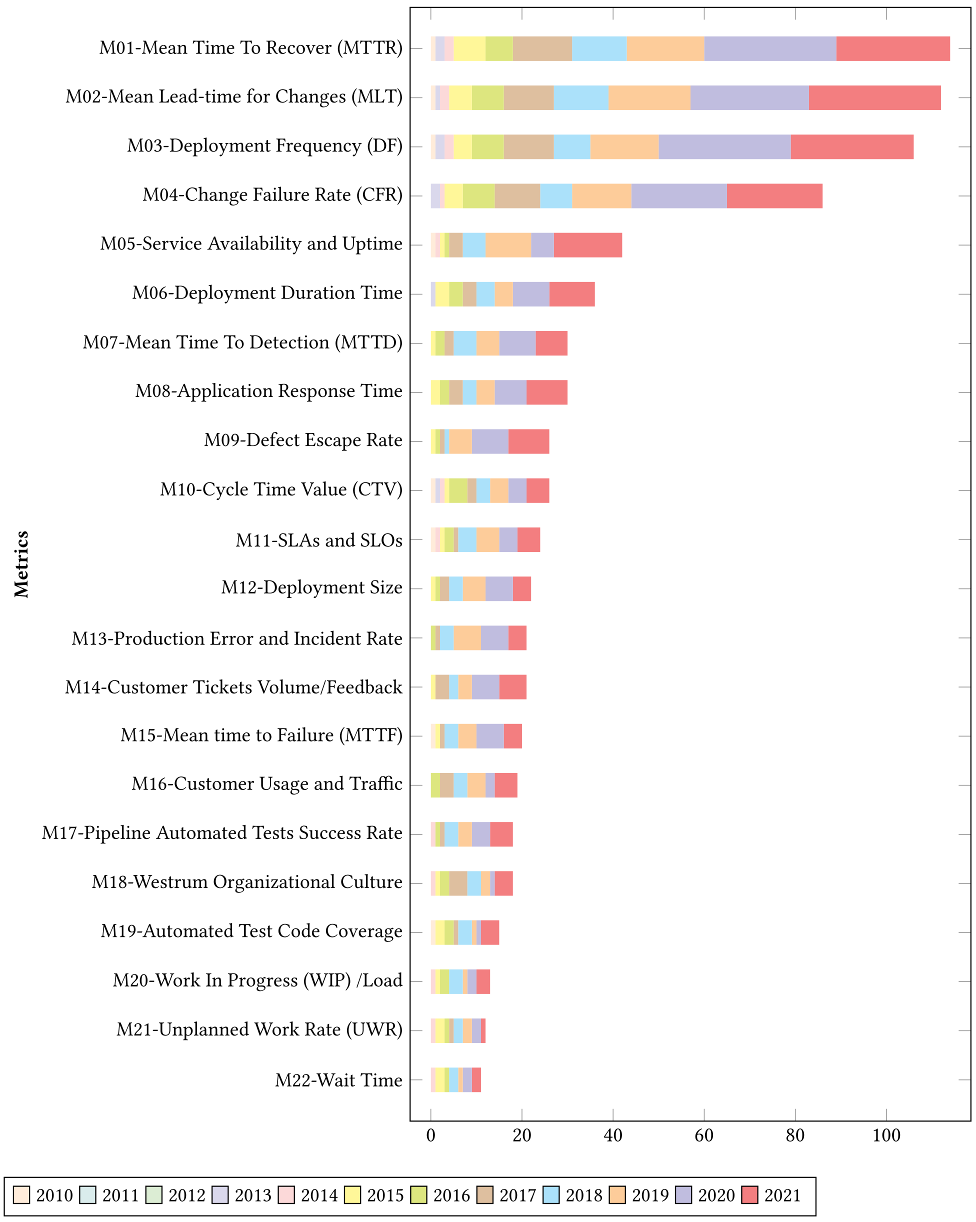


Fig. 7. Top main metrics mentioned in publications over the years.

conducted in the various State of DevOps Reports over the years [1, 35, 144–151] and the work published by Forsgren et al. [51, 52, 54, 55, 57, 58].

Looking at the reasons pointed out during the 2020's jump in gray literature, it is found that ***Mean Time To Recover (MTTR)*** *(M01)* is one of four most distinguished key metrics for DevOps teams [67, 107, 178]. The average cost of downtime for companies rises year after year [189]. MTTR emphasizes critical business outcomes that are directly related to customer experience, acquisition, and retention [3]. ***Mean Lead-time for Changes (MLT)*** *(M02)* is fundamental, because it measures the time it takes for a code change to reach production. It gives insight into the DevOps process's efficiency, complexity, and the team's ability to meet customer needs [29, 50]. Short lead times suggest immediate feedback, while long lead times indicate inefficiency. ***Deployment Frequency (DF)*** *(M03)* approaches infinite in just-in-time manufacturing as batch size approaches zero, therefore deployment frequency of software is a KPI for software delivery teams [7, 57, 64, 128, 130, 143]. A team that deploys more than once per week can fix outages in production faster and deliver value to customers more frequently [60, 91, 156]. *Change Failure Rate (M04)* represents changes in production that require an immediate fix to resolve, thus a more complex metric [178]. An increasing failure rate reveals processes problems in the delivery pipeline [70, 102, 154]. From the relation of these and the other metrics over the years, it is shown that practitioners are championing these metrics in their publications. Given that we are reporting the MLR in Section 5, the main metrics will be further discussed and defined as part of that reporting.

5.1.2 Purpose and References for Each Main DevOps Metric. Following the research, there are 22 main DevOps Metrics found in this Multivocal Literature Review, gathered from all the publications, selected for review. It is observed that, alongside the pure academic work, there has been a growing impact from the State of DevOps Reports, DORA, and the work of Forsgren et al. on the data being collected over the years [53], which shows the importance of the primary sources collected. The MLR intends to give voice to all, including academia, practitioners, and the DevOps community in general.

In Section 4 the found list of the main DevOps Metrics is shown, along with their purpose, mentions, and the total of references. A summarized description mentioning their purpose is also listed, taken out from the qualitative coding already described. This list is a subset with the most important metrics from all the metrics collected in Section 5.1.1, where we saw that most tend to be “business as usual” measures that would still add value to the organization but are not a critical measure needed to focus on. As a result, every KPI listed here is a metric, but not every metric that has been found is a KPI.

For these 22 selected main DevOps Metrics, it was considered that they were referenced ten or more times in relation to the others shown in Appendix B, and always keeping in mind how DevOps focuses on the impact of things such as profitability [22, 155], productivity [37, 59], quality [13, 118], product or service improvements [103, 172], operational efficiency [6, 48], customer feedback [5, 191], and achievement of organizational goals [82, 158]. Therefore, this last stage in the refinement process is based on organizing the metrics by the number of references, which gives the reading in Section 4 based on the number of times the selected publications mention them.

5.2 RQ2—What Is the Precise Definition and Importance of Each Main DevOps Metric?

After having identified the main DevOps Metrics, some questions may remain as to their usefulness and applicability for increasing DevOps performance. This research question intends to answer to what is the precise definition of each key metric, why it matters, and discuss how a team can improve the metrics and the expectations without confusion or ambiguity. To clarify these questions, the following definitions gathered from qualitative research aim to provide details about the practicality of each metric, as well as its relevance to organizations. Table 4 shows the purpose and references for each main DevOps Metric and Table 5 shows a summary of definitions for each main DevOps Metric and their optimal trend.

M01. ***Mean Time To Recover (MTTR).*** *Definition:* How quickly can teams restore service in case of a production outage? MTTR is an essential metric that indicates the ability to recover appropriately from identified issues. It is measured as the time from when impairment occurs until the time it is resolved, then the averaging of all those values [7, 26, 58, 74, 144, 145, 183].

Why it matters: For organizations and individuals, time is money, and wasting time on false positives or difficult issues frustrates development teams and slows down the automation. MTTR is a good way for a manager to assess the capabilities of their teams [4, 50, 199, 206]. This is a good indicator of how well the team handles change and responds to problems. Failures are unavoidable, but how we respond is a far more important indicator of our team’s agility. MTTR

Table 4. Purpose and References for Each Main DevOps Metric

ID	Metric	Purpose	References	Total
M01	**Mean Time To Recover (MTTR)**	Measures the mean of the time required to recover or restore service from a failure in production.	[1–4, 7–10, 12, 14, 16, 19, 21, 24, 26–28, 30, 33–36, 40–43, 45, 47, 49–55, 57, 58, 61, 66, 67, 70, 72, 74, 75, 77–80, 82, 86, 88–90, 93, 97, 98, 102, 106–108, 117–119, 123, 124, 127, 128, 130, 132, 133, 137–140, 143–149, 151, 154, 156–158, 160, 163, 166, 168, 169, 171, 173, 175, 176, 178, 183–185, 187, 190, 194, 196, 198–202, 204–206, 208, 209]	114
M02	**Mean Lead-time for Changes (MLT)**	Indicates how long it takes for a change to go from code committed to code successfully running in production.	[1, 2, 4, 7, 8, 10, 12, 14, 16, 19–21, 24, 26–28, 30, 33–36, 39–43, 46, 47, 49–54, 57, 58, 64, 66–68, 70, 72, 74, 75, 77–80, 82, 83, 86, 88, 89, 91, 93, 97, 98, 100, 102, 106–108, 115, 117, 119, 123–125, 127, 128, 130, 131, 137–139, 142, 144–149, 151, 154, 156–158, 160, 163, 166, 168, 169, 173, 176, 178, 183–185, 188, 190, 194, 198, 199, 201, 203–205, 207–210]	112
M03	**Deployment Frequency (DF)**	Checks how often changes are deployed to production.	[1, 2, 4, 7–10, 12, 14, 16, 19–21, 24, 26–28, 30, 33–36, 40–43, 45, 47, 49, 50, 52–54, 57, 58, 60, 61, 67, 70, 72, 74, 75, 78–80, 82, 83, 86, 88–90, 93, 97, 98, 100, 102, 106–108, 115, 117–119, 123, 125, 127, 128, 130, 132, 137, 139, 143–149, 151, 154, 156, 158, 160, 163, 166, 168, 169, 173, 175, 176, 178, 183, 184, 190, 196, 198, 200, 201, 203–209]	106
M04	**Change Failure Rate (CFR)**	Informs how often a change in production fails and must be immediately remedied.	[1, 4, 7, 9, 12, 16, 19, 21, 26–28, 30, 34–36, 40–43, 45, 47, 50, 52–54, 58, 64, 67, 70, 72, 74, 75, 78, 79, 83, 86, 89, 93, 97, 98, 102, 106–108, 115, 117, 119, 123, 125, 127, 128, 130, 132, 137, 139, 143–149, 151, 154, 156, 158, 160, 163, 168, 169, 173, 175, 178, 183, 188, 190, 198–201, 206–210]	86
M05	Service Availability and Uptime	Shows the percentage a service is available during a period of time.	[1, 4, 8, 19, 24, 30, 33–35, 40, 42, 49, 51, 53, 54, 60, 61, 77, 80, 82, 93, 116, 118, 122, 124, 125, 128, 138, 142, 150, 156, 157, 169, 173, 175, 176, 180, 185, 200, 201, 205, 207]	42
M06	Deployment Duration Time	Informs on how long it takes to deploy a set of changes.	[9, 14, 20, 28, 33, 40, 42, 45, 49, 51, 52, 66, 79, 83, 100, 115, 119, 123, 124, 128, 132, 134, 138, 142, 154, 156, 169, 175, 176, 183, 196, 198, 200, 201, 205, 206]	36
M07	**Mean Time To Detection (MTTD)**	Measures the mean of the time required to detect a failure in production.	[3, 4, 19, 30, 33, 34, 40, 42, 49, 51, 61, 78, 88–90, 93, 115, 119, 123, 128, 143, 156, 157, 160, 166, 176, 200, 201, 205, 210]	30
M08	Application Response Time	How an application responds to increases or decreases in user traffic and activity.	[4, 24, 26, 30, 33, 40, 49, 53, 61, 93, 119, 123, 128, 133, 138, 142, 151, 157, 160, 173, 176, 185, 190, 194, 196, 200, 203–205, 207]	30
M09	Defect Escape Rate	Indicates the number of defects discovered in production versus the number of defects found during development.	[7, 9, 10, 33, 40, 42, 46, 49, 78, 119, 128, 132, 154, 156, 160, 163, 166, 171, 175, 176, 188, 196, 200, 203, 205]	26
M10	**Cycle Time Value (CTV)**	Provides information on the full Cycle Time Value, beginning with deciding to make a change and finishing with delivering to the end user.	[10, 14, 28, 39, 42, 53, 66, 74, 80, 82, 91, 93, 108, 118, 121, 132, 143, 144, 171, 175, 187, 201, 202, 204, 208, 209]	26
M11	**Service Level Agreements (SLAs)** and Objectives (SLOs)	Sets customer expectations for service availability with SLA and internal teams with SLO.	[1, 4, 8, 19, 30, 33, 35, 42, 49, 51, 53, 80, 82, 83, 93, 100, 121, 150, 156, 157, 176, 201, 203, 205]	24
M12	Deployment Size	Shows the number of changes incorporated in each production release.	[7, 26, 33, 40, 42, 46, 49, 115, 119, 123, 124, 128, 131, 154, 168, 173, 175, 176, 201, 204, 205, 207]	22

(Continued)

Table 4. Continued

ID	Metric	Purpose	References	Total
M13	Production Error and Incident Rate	Measures the frequency of faults and incidents in production following a deployment.	[10, 19, 33, 49, 77, 88, 90, 100, 107, 124, 128, 154, 157, 171, 175, 176, 194, 196, 200, 205, 210]	21
M14	Customer Tickets Volume and Feedback	Indicates the level of satisfaction of customers using their feedback.	[7, 10, 33, 42, 46, 49, 78, 83, 89, 100, 117, 119, 124, 125, 128, 156, 173, 176, 201, 205]	21
M15	**Mean time to Failure (MTTF)**	Exposes the average time a flawed deployment into a system will manage to run until it fails.	[2–4, 19, 33, 34, 42, 49, 60, 80, 88, 90, 117, 119, 123, 156, 176, 200, 201, 205]	20
M16	Customer Usage and Traffic	Measures usage and traffic of customer-facing applications when there are defined business goals to increase.	[29, 33, 49, 66, 93, 132, 138, 142, 150, 156, 157, 160, 166, 173, 176, 190, 194, 205, 207]	19
M17	Pipeline Automated Tests Success Rate	Shows the rate of successful pipeline automated test jobs.	[10, 14, 24, 33, 53, 66, 82, 93, 119, 128, 131, 138, 140, 154, 175, 176, 205, 207]	18
M18	Westrum Organizational Culture Measures	Result of the Westrum cultural assessment [211]	[1, 11, 35, 51–55, 82, 93, 108, 116, 124, 125, 147, 150, 206, 208]	18
M19	Automated Test Code Coverage	Measures how many lines, statements, or blocks of code are tested using the suite of automated tests.	[9, 14, 24, 49, 51, 60, 66, 80, 93, 131, 132, 156, 175, 180, 202]	15
M20	**Work In Progress (WIP)** /Load	Presents the number of open issues of each type (story, defect, task).	[20, 29, 39, 51, 53, 60, 64, 66, 82, 91, 93, 125, 157]	13
M21	**Unplanned Work Rate (UWR)**	Indicates the amount of time spent on tasks that were not in the initial plan.	[19, 42, 53, 82, 107, 117, 119, 125, 146, 148, 150, 210]	12
M22	Wait Time	Measures the amount of time spent waiting for the next step to add value.	[29, 39, 51, 64, 82, 91, 93, 117, 123, 125, 202]	11

should always be a focus for DevOps KPI monitoring, as improving MTTR contributes to better customer satisfaction, faster application delivery, and better cost control [33, 35, 50, 57, 118, 151].

Metric expectations: This metric should trend down or remain stable over time [42, 77, 196].

How to improve: To reduce MTTR, it is imperative to use good alerting and monitoring tools to identify an issue on time and promptly fix it. An effective collaboration between operations and developers is needed, which can help teams find root causes and deploy solutions quickly [14, 140, 206].

M02. ***Mean Lead-time for Changes (MLT)***. *Definition:* The time it takes to go from code committed to code successfully running in production. MLT in DevOps, can be seen as CTV including beginning development and time for delivering the finished product. Measuring MLT requires starting the clock when there is code committed and stopping it when said code enters production [30, 35, 75, 131, 144, 145, 149–151]. The book "Accelerate" attempts to clarify the confusion around CTV and MLT terms, frequently used interchangeably, even though they measure different things. "Lead Time" (time between code commit and deployable code) and "Cycle Time" (which some define as the time from code starting to be worked on by development to code in a deployable state) are two examples [53].

Why it matters: MLT offers valuable insight into the efficiency of the entire development process. Projects that have not successfully implemented agile will frequently have lengthy lead times. Tracking Mean Lead Time for Changes and the process' subsequent components can assist the team in identifying areas for improvement. Keeping track of the total time it takes from source code commit to production release can help indicate the speed with which software is delivered [67, 70, 102, 106, 173, 183].

Table 5. Summarized Definition for Each Main DevOps Metric and Their Optimal Trend

ID	Metric	Definition	Trend
M01	**Mean Time To Recover (MTTR)**	The time to recover appropriately from identified issues in production.	*down* ↓
M02	**Mean Lead-time for Changes (MLT)**	Average time for the code committed to be running in production.	*down* ↓
M03	**Deployment Frequency (DF)**	The number of software deployments to production over a period of time.	*up* ↑
M04	**Change Failure Rate (CFR)**	Percentage of deployments that result in impairment or outage in production.	*down* ↓
M05	Service Availability and Uptime	The percentage of continuous availability of service over a period of time.	*up* ↑
M06	Deployment Duration Time	Duration of deploying a previously built artifact into environments and production.	*down* ↓
M07	**Mean Time To Detection (MTTD)**	Time between the start of an issue and the detection of the issue in production.	*down* ↓
M08	Application Response Time	The duration for an application to respond to a user's request or input.	*down* ↓
M09	Defect Escape Rate	The number of defects or issues discovered after releasing to production.	*down* ↓
M10	**Cycle Time Value (CTV)**	The time it takes from the decision to make a change to having it in production.	*down* ↓
M11	**Service Level Agreements (SLAs)** and Objectives (SLOs)	Customer service quality agreements, and internal performance objectives.	*up* ↑
M12	Deployment Size	The number of changes incorporated in each production release.	*down* ↓
M13	Production Error and Incident Rate	The frequency at which errors or incidents occur in a live production environment.	*down* ↓
M14	Customer Tickets Volume and Feedback	The number of customer support tickets and feedback to be addressed.	*down* ↓
M15	**Mean time to Failure (MTTF)**	The average time a flawed, non-recoverable asset will manage to run until it fails.	*up* ↑
M16	Customer Usage and Traffic	The amount of traffic from active users in the system.	*up* ↑
M17	Pipeline Automated Tests Success Rate	The success rate of pipeline automated test jobs.	*up* ↑
M18	Westrum Organizational Culture Measures	Categorize organizations into three types: Pathological, Bureaucratic, or Generative [211]	→ *generative*
M19	Automated Test Code Coverage	The percentage of the relevant codebase that is tested by automated test cases.	*up* ↑
M20	**Work In Progress (WIP)** /Load	The number of tasks, projects, or items that have been initiated but are not yet completed.	*down* ↓
M21	**Unplanned Work Rate (UWR)**	The time spent on addressing unexpected tasks and incidents.	*down* ↓
M22	Wait Time	The delay experienced before work items progress through the value stream.	*down* ↓

Metric expectations: This metric should trend down or remain stable over time. To calculate the Lead Time, the time of the request and respective delivery need to be known [74, 144, 151, 158]. Elite performers have a lead time for changes of less than 1 hour and Low performers have a lead time for changes that is between 1 and 6 months according to various State of DevOps [1, 35, 149, 151].

How to improve: Lower lead times indicate that the team is adaptable, responsive, and can respond quickly to feedback. Version control and automated testing are highly correlated with lead time. Working in small batches improves process efficiency, but to measure MLT the team needs a clear start and end of work defined [106, 147, 148, 173, 205]. Shorter lead times indicate a team's agility, responsiveness, and ability to adapt to feedback.

M03. ***Deployment Frequency (DF)***. *Definition:* The number of software deployments to production over a period of time. The number of times a piece of code/software is pushed (released) to production [41, 74, 143]. It can be measured using various methods, such as automated deployment pipelines and API calls. How often does the team deploy a new version of a specific product or service? In other words, DF shows how often the organization deploys code into production [16, 35, 58, 75, 147, 154, 180, 202].

Why it matters: The frequency with which a team deploys changes is critical to the success of DevOps. Increasing the frequency of deployments has been a powerful motivator for changes in development practices [30, 140, 166]. Frequent deployments can help resolve production outages faster, because they have the automation to quickly and easily deploy changes. The quicker teams can deploy, the sooner they can provide value to end users. [66, 150, 183, 194, 198, 205, 205].

Metric expectations: Every time a deployment occurs, a counter will increase. The frequency can be measured daily or weekly. One approach to quantify DevOps value is to track deployment frequency over time. It could be measured via an automated deployment pipeline, API requests, or manual scripts. Many organizations choose daily tracking to increase productivity [16, 35, 39, 46, 67, 72, 89, 106, 107, 147, 173, 199].

How to improve: A well-designed CI/CD pipeline enhances deployment frequency. Engineering teams may deliver products and minor enhancements faster by segmenting deployments. We want to perform as many smaller deployments as feasible. Smaller deployments make testing and releasing easier. These pipelines assist remove errors and increase product confidence [9, 16, 28, 34, 35, 70, 100, 117, 127, 150, 202].

M04. ***Change Failure Rate (CFR)***. *Definition:* The percentage of deployments that result in service impairment or an outage in production. If KPIs show an increasing rate of failure as deployments increase, then it is time to slow down and investigate problems in the development and deployment pipeline. The change fail percentage is the ratio of failed to successful changes. How frequently we fail over time can measure both production failures and failures in our testing, deployment, or the DevOps pipeline [27, 41, 47, 51, 53, 132, 133, 176, 180].

Why it matters: This metric should reveal the flaws in the deployment strategy and not the outside world. Failed deployments can take services down, resulting in lost revenue and frustrated customers. Well-implemented DevOps capabilities can make a big difference in failure rate. A high change failure rate suggests poor application stability, which can lead to negative end-user outcomes. Failed deployments cause revenue losses and unsatisfied customers [19, 21, 27, 61, 74, 82, 147, 176].

Metric expectations: This metric should be as low as possible or remain stable over time. The Change Failure Rate is a measure of the quality of the release process. It is calculated by dividing the total number of failed deployments and the number of deployments that resulted in production failures [30, 42, 106].

How to improve: If we want to reduce deployment failures, then we need to add more automated tests. If the change failure rate is increasing, then teams should consider reducing the deployment frequency. Automation should be used for security testing, unit testing, and integration testing. A low failure rate for changes indicates rapid and frequent deployment, whereas a high failure rate indicates an unstable DevOps practice and process [36, 102, 127, 130, 183]. While a failure rate of

0 would be ideal, a failure rate of less than 5% is considered workable by most authors [88, 128]. Fixing five issues in 100 deployments is far easier than fixing 50 issues in 1,000 deployments in the same amount of time [67, 190].

M05. ***Service Availability and Uptime***. *Definition:* Uptime (or Availability) is the percentage of continuous availability of service over a period of time. The uptime percentage is straightforward to quantify and track. It can be calculated from the data center or cloud service downtime and availability data [61, 175]. The amount of downtime the service experiences, as well as the level of service availability for end users, demonstrates the dependability of the applications and infrastructure [30, 54, 61, 77, 156, 205].

Why it matters: As DevOps capabilities are adopted, availability should increase and downtime should reduce. This is a critical component to monitor for software as a service businesses. The availability of a service is critical for sustaining customer satisfaction [61, 207]. It also serves as a significant statistic for determining the success of changes, the reacting speed on infrastructure issues and the overall success of projects [30, 33, 49, 77, 118, 128, 156].

Metric expectations: This metric should trend upward or remain stable over time. A team calculates availability by adding up all reported outages by our primary production monitor for each service, subtracting them from the total time, and then dividing by the total time [34, 169, 175, 176, 207]. A Hundred percent availability is unlikely, as scheduled maintenance would require planned downtime. The team calculates availability by adding up all reported outages and dividing them by the total downtime. [33, 51, 80, 82, 93, 128, 173, 180].

How to improve: DevOps delivery value can be tracked by measuring downtime and availability as KPIs, which are somewhat related to the total number of incidents. With less downtime and greater availability, DevOps organizations can likely promise more enticing **Service-level Agreements (SLAs)**, **Service-level Indicators (SLIs)**, and **Service-level Objectives (SLOs)** to customers [77, 123, 201].

M06. ***Deployment Duration Time***. *Definition:* The total time it takes to deploy a previously built artifact in infrastructure environments and production [45, 119, 206]. This metric quantifies the time required to promote an application or service from one environment to another [57, 100, 128, 196]. Specifically, after the change is approved or automated deployment of the artifact starts.

Why it matters: Given that the goal of DevOps is to accelerate software delivery, tracking the time to deploy that software is a useful metric. Monitoring deployment time can reveal delivery challenges. Increasing error rates may indicate badly planned releases [49, 100, 156]. It can help identify potential problems and allow a dramatic increase in revenue by using that extra time to develop more value-added services [123, 205, 206].

Metric expectations: This metric should trend down or remain stable over time. When measuring this metric, it is vital to pay attention to any sudden and dramatic rise in deployment time [52, 58, 80, 83, 119, 124, 205, 206].

How to improve: Measure the time taken to roll out deployments after they are approved. Track how long it takes to do an actual deployment and investigate bottlenecks [80]. Dramatic increases in deployment time warrant further investigation, especially if they are accompanied by reduced deployment volume [42, 49, 128, 183].

M07. ***Mean Time To Detection (MTTD)***. *Definition:* The amount of time between the start of an issue and the detection of the issue in a production environment, ideally at which point some action is taken [61, 90, 166]. It's an indication of how effective are incident management tools and processes [30, 34, 34, 42, 88, 88, 176].

Why it matters: While the ideal solution is to minimize or even eradicate failed changes, it's essential to catch failures quickly if they do occur [30, 166]. Time to detection can determine whether current response efforts are adequate. A High MTTD could raise bottlenecks capable of interrupting the entire workflow [42, 51, 93].

Metric expectations: This metric should trend down or remain stable over time. Teams should work to keep their MTTD as short as possible. With proper instrumentation, alerting, and notification channels, teams will be able to more quickly respond to any error detection [42, 49, 78].

How to improve: In addition to MTTR, the team needs to track how long their average incident response is. At what stage of the incident lifecycle is the most time spent [77]? Having robust application monitoring and good coverage will help detect issues quickly. Once a team detects them, they also have to fix them quickly [119, 128, 205]. In contrast, a more mature team that has monitoring implemented can detect issues faster through the data that team members capture, such as logs or performance data [90, 90, 128, 143, 156].

M08. ***Application Response Time***. *Definition:* The response time and performance of an application in production. It is a good practice to look for performance issues before deploying an application [123, 203, 207]. However, it is equally important to track application performance during and after deployment. This is crucial for DevOps success, since the performance of parameters such as web service calls, queries, and other dependencies can change after application deployment [119, 205].

Why it matters: It is vital to maintain a good user experience. Before deploying, the team should run a tool to check for performance and hidden errors [26, 203]. The functionality of an application is examined more frequently. Optimizing services benefits customers and the organization as a whole [49, 61, 207]. Application performance may be included in a SLA [133, 203].

Metric expectations: This metric should trend down or remain stable over time. Time to first byte, error rates, and response time are common performance metrics for web applications [40, 119, 151]. The user experience that a service or application provides can be easily quantified. It shows that the software/application is working properly within the defined parameters [33, 196].

How to improve: It is difficult to simulate all the different network paths and hardware configurations that a client might use during a session. Therefore, techniques like blackbox monitoring [203] can be effective in helping get a good measurement, since it has no knowledge of the interior metrics or design. Before performing the deployment, a team should check for performance faults, unknown bugs, and other problems [119, 201].

M09. ***Defect Escape Rate***. *Definition:* The percentage of defects that are not caught during testing and are discovered in production. This can be reviewed by time period or as a ratio to the number of deployments. Less than one percent of customers or users find defects in production, while QA finds most bugs in pre-production [166, 203]. Bug tickets or support tickets are typically used to track these defects [78, 157, 205].

Why it matters: It shows how many defects were found in production, during, and after deployment. Preventing a bug from reaching production is easier if it is discovered during pipeline development or testing. They may, however, not be detected and pass tests so deployments can still introduce bug fixes [42, 113, 113, 171].

Metric expectations: This metric should trend down or remain stable over time. Trying to achieve a defect-free operation can lead to DevOps anti-patterns like change reluctance or feature upgrade delays. Make SLAs sane error budgets. Abnormally high defect rates could be the first sign of problems in testing, qualification, or in team performance [33, 156, 176].

How to improve: The Defect Escape Rate is a useful DevOps Metric that counts software bugs found in production during and after deployment. This indicates when the quality process needs to be improved. To ship code quickly, a team must be confident in its software defect detection ability [7, 49, 119, 128, 175].

M10. ***Cycle Time Value (CTV)***. *Definition:* The time it takes from deciding to make a change to have it in production [10, 28, 171]. How long does it take to deploy a change that only involves one line of code? In software development, typically it is the time from code starting to be worked to code in a delivered state [35, 53]. The CTV metric provides a broad overview of application deployment.

Why it matters: In many organizations, Cycle Time is measured in weeks or months, and the release process is manual. Teams must be able to achieve a Cycle Time of hours or even minutes for any critical fix. This is possible using a fully automated, repeatable, reliable process for taking changes through the various stages of the build, deploy, test, and release process [26, 42, 201]. It is manual and often requires a team of people to deploy the software even into a testing or staging environment, let alone into production [80].

Metric expectations: This metric should trend down or remain stable over time. This process should be managed using a single ticketing system that everybody can log into and that generates useful metrics such as average cycle time per change [80, 131, 171].

How to improve: The end-of-cycle security testing and assessments frustrate developers and business owners. The earlier automated testing is the better. Avoid manual security testing of new code and builds to reduce CTV [91, 143, 175]. Cycle Time is a key process efficiency indicator. Value stream mapping aids in identifying waste removal and automation requirements. Defect detection and SLA adherence are prioritized over Cycle Time reduction [42, 171].

M11. ***Service Level Agreements (SLAs) and Objectives (SLOs)***. *Definition:* SLOs are specific measurable characteristics of the SLA such as response time, throughput, availability, frequency, or quality indicators. The SLA is the entire agreement that specifies a provided service, how it is supported, times, locations, costs, performance, and responsibilities of the parties involved [49, 80, 157]. There are two types of SLAs: soft (ideal goal) and legal (contractual obligation) [203]. Based on technical reality, a SLO target should reflect what the team or organization actually can support [100].

Why it matters: To increase transparency, most companies operate according to SLAs and SLOs. Often, teams have customer facing service-level metrics, based on SLIs [15, 18, 35, 42, 121, 201], they are accountable for, thus aligning expectations between providers, clients or internally[15, 18, 42, 157]. This is a fundamental aspect, since it enables DevOps teams to release and experiment improvements [35, 53] within defined boundaries and without fear, contributing to a culture of psychological safety [98, 158, 211].

Metric expectations: These metrics are defined to set stakeholders expectations and should remain within boundaries. Service levels can change over time. For example, if a system is immature, then initial modest SLOs [157] can be increased over time [42, 201, 203, 205].

How to improve: Ensure to be compliant with SLOs and SLAs [146, 149, 201]. Any disagreement with SLAs causes issues at a later stage, hampering the workflow. It is important to operate within defined service levels, therefore keeping track of Service Level Indicators (SLIs) and error budgets [15, 18] is key. A good starting point for SLOs is using an open specification like OpenSLO.[3]

M12. ***Deployment Size***. *Definition:* Total new user stories and new lines of code that are shipped in each deployment in production. Volume of code changes focus on the new lines of code deployed per build. Change Volume refers to the lines of code are pushed to production per deployment [119, 207]. Measuring this is crucial to measuring the success of deployment in terms of value, time and frequency. This DevOps Metric is critically comparing the static code and ratio changes within the organization [7, 119, 175].

[3]OpenSLO uses YAML to define reliability and performance targets. https://openslo.com/

Why it matters: This DevOps KPI determines the extent to which code is changed versus remaining static. Improvements in deployment frequency should not have a significant impact on change volume [7, 42, 119]. Tracking how many stories, feature requests, and bug fixes are being deployed is another good DevOps Metric. A team could also track how many story points or days' worth of development work is being deployed. Some companies use this metric to find out the total new stories and new lines of code they ship for each deployment [123, 173, 201, 205].

Metric expectations: This metric should trend down or remain stable over time. The end goal shouldn't be a constant stream of minor but insubstantial changes; instead, focus on impactful updates that provide a better experience with less disruption [49, 173]. Tracking the amount of change with each deployment allows for a more accurate representation of progress [42, 123, 201, 205].

How to improve: In DevOps, changes should come often and be in small batches. But the sizes of those pieces can vary. For each deployment, monitoring the volume of change makes for a more precise depiction of development. Finding a good average of frequent and impactful changes leads to success rates [42, 119, 207]. Keep track of the number of bug fixes and feature requests delivered in each release's deployment artifacts [42, 49, 176].

M13. *Production Error and Incident Rate*. *Definition:* Rate percentage of production incidents and errors. The error rate tells the team how often new problems appear in running applications [90, 205]. Bugs after a new release, database connection issues, query timeouts, other issues all contribute to the uptime and system performance metrics of operations [77, 90, 128, 130, 133, 196].

Why it matters: It is vital to track application error rates. They indicate not only quality issues but also performance and uptime issues. Less time between deployments means more production incidents [49, 100, 205]. Constant testing policies, release management processes, and monitoring and alerting improvements are all common. The goal is to keep production incidents below the value delivered to customers [77, 88]. Not all errors are equally impactful for customer's trust [33, 90, 194].

Metric expectations: This metric should trend down or remain stable over time. The error rate is calculated as a function of the transactions that result in an error during a particular time window [88, 128]. A few intermittent errors throughout the application life cycle is a normal occurrence, but any unusual spikes that occur need to be looked out for [90, 205].

How to improve: Errors are common in most applications. Apply good exception handling. Errors are part of a busy system's noise [205]. Keep an eye on the error rates and look for spikes via log analysis [88, 90]. Error rate spikes must be captured, because they can indicate a problem. The raw incident count can also help compare the team's incident load to the organization's average [130, 133, 196].

M14. *Customer Tickets Volume and Feedback*. *Definition:* Amount of customer support tickets and feedback on how many problems are filed as support tickets. This metric is a measure of end user satisfaction and a good indicator of production problems [100, 113, 173, 205]. Bugs and errors can frequently pass through the testing phase and only be detected by the end user. As a result, the number of customer tickets labeled as problems or bugs is a key indicator of application reliability [49, 78, 119, 128].

Why it matters: This reflects how many problems users find and how they are solved, which surfaces quality and performance issues. Ideally, customers should not be finding problems [26, 49, 78]. Most companies track user ticket generation to assess performance, since customer satisfaction is vital to product survival. Satisfied clients increase sales. So, customer support tickets reflect DevOps improvements needed [78, 100, 119, 128].

Metric expectations: This metric should trend down or remain stable over time. It is important to note that not all defects are disastrous, but they should be caught early. Customer satisfaction leads to a competitive advantage [33, 49, 78, 172].

How to improve: Keep track of customers happiness and prioritize improvements. Happy customers translate into business growth and fewer expenses on addressing issues [78]. Use a production debugger to prevent issues from getting to production, and fix them as quickly as possible if they do. While organizations should qualify which customer tickets to include in this metric, it can be a good overall measure of DevOps success [7, 42, 113].

M15. ***Mean time to Failure (MTTF)***. *Definition:* MTTF is the average time a flawed, non-recoverable asset will manage to run until it fails. The duration starts when a major flaw occurs in the system and ends when the mechanism fails [90, 119]. This may reveal how often system components generate flaws leading to failures, which may imply routine maintenance. This metric relates to improving system uptime [34, 49]. Since this failure is non-recoverable, the asset or component needs to be replaced. Examples are a hard disk, a microservice Kubernetes pod or a flawed virtual machine. MTTF is associated with costs, capacity planning and risk management [15, 84, 195, 197]

Why it matters: MTTF is used to monitor non-repairable, disposable system assets or components and determine their lifespan, allowing a team to monitor the health of mission-critical components [60, 119] and forecast automated replacement. A high MTTF rate can also indicate software quality issues. For example, tests may not be covering all possible scenarios [90, 201]. Reliability is a function of MTTF and MTTR [15].

Metric expectations: This metric should trend upward or remain stable over time. It is an indication of how long on average the system or a component can run before failing [34, 88, 90, 205].

How to improve: Based on the available data, forecast the eventual failure of the asset or component. Compare different versions and perform a preventive maintenance [34]. Improve the time elapsed between installation and the first failure [49] and aim for continuous service availability and correct system behavior even if a failure of some kind occurs [42, 49].

M16. ***Customer Usage and Traffic***. *Definition:* The amount of traffic from active users in the system. Following a deployment, teams should check to see if the number of transactions or users accessing the system appears to be normal [205]. Something could be wrong if teams suddenly see a massive spike or no traffic [49, 194, 201, 207].

Why it matters: A good way to ensure the deployments are successful in the eyes of users and customers is to track changes in usage across services. If usage drops after a change, then something is wrong or the changes are not working for customers [49, 205]. Teams should constantly monitor usage and traffic to identify general trends and sudden deviations that may be controlled [173, 207].

Metric expectations: This metric should trend upward or remain stable over time. Increase usage metrics, for customer-facing applications, when there are defined business goals for it [49, 176, 205].

How to improve: Teams want to see normal service usage after a new version is released. This measure affects system uptime. It is important to get new or improved features into production quickly, but that does not mean customers will use them [46, 166]. After we announce a feature's availability, teams can compare its actual popularity to previous predictions. Use hypothesis-driven development [82, 94, 181, 194] to influence feature prioritization via experimentation.

M17. ***Pipeline Automated Tests Success Rate***. *Definition:* The percentage of successful tests per build. Test pass rate will combine the percent success of the unit tests, functional tests, performance tests, and security tests [128, 207]. This entails effectively utilizing unit and functional testing to determine how frequently changes in code result in test failures [175].

Why it matters: To maximize velocity, it is advised that the team make effective use of unit and other automated testing. Because DevOps is heavily reliant on automation, measuring how well automated tests perform is a useful DevOps Metric [201]. It is useful to know how many code changes cause the tests to fail. Because DevOps is a highly automated process, keeping track of the automated test pass percentage is critical for maintaining an upward trend in deployment velocity [82, 128, 207].

Metric expectations: The success viewpoint of this metric should trend upward or remain stable over time. A team can get this data from an orchestration tool such as Jenkins or the respective test tool such as Selenium, JMeter, JUnit, and others [33, 175, 176, 205].

How to improve: Automation is an important DevOps practice that should be used extensively to avoid repetitive tasks. Controlling automated test performance indicators helps focus on key results. Measuring the results of automation tests can also help ensure that ongoing efforts are paying off. Automated test cases must be fully utilized to achieve superior performance in DevOps. Keep an eye on code changes that affect test cases [119, 128, 207].

M18. *Westrum Organizational Culture Measures*. *Definition:* Measures the performance indicators of the organization. In contrast to other broad measures of culture, such as national culture, an organization's culture can be seen and observed in both its formal and informal states, such as mission statements, goals, shared values, and social norms. Westrum et al. [211] proposed a model of organizational culture evaluation that emphasized the importance of information flow in complex, high-risk work environments. For this reason, the State of DevOps related studies selected this framework for inclusion in their research, and adapted it for use in research to assess DevOps capabilities [51, 54, 98].

Why it matters: Culture measurements must be an integral part of any DevOps process. It shows that elite performers are more likely to meet or exceed their goals for organizational performance [35]. These outcomes are measured by many factors, including productivity, profitability, and market share as well as non-commercial measures such as effectiveness (value addition), efficiency (faster cadence), and customer satisfaction [54, 91]. To thrive within a DevOps ecosystem, the team must be encouraged in innovation and focuses on integrating lean principles and shorter implementation cycles [98, 166, 206].

Metric expectations: This metric can position the organization in one of three types—pathological, bureaucratic, and generative. The output should trend toward generative type over time. These measurements are of particular interest to software developers, operations engineers, project managers, and engineering leadership in DevOps [54, 147, 206].

How to improve: Leaders can help their teams gain a culture of high mutual trust with autonomy in their work by establishing and communicating goals, but letting the team decide how the work will be done. Allowing the team to remove obstacles for achieving the goals and letting the team prioritize good outcomes for customers [82]. Research finds that more autonomy fosters trust in the leader—that is, the team believes its leader is fair, honest, and trustworthy. This trust in transformational leadership contributes to a stronger organizational culture [53, 54, 150]. Other key improvements are the culture of psychological safety, dependability, sense of meaningful work, impact and climate of learning in the company[18, 35, 53].

M19. *Automated Test Code Coverage*. *Definition:* The percentage of code associated with automated test scripts. Code Coverage indicates the number of lines of code that are executed while the automated tests are running [9]. It can be further broken down into Unit Test Coverage or Automated Test Coverage. The faster the automation, the more tests that can be incorporated as continuous testing in the CI/CD pipeline [131, 166, 205].

Why it matters: To increase velocity, it is highly recommended that a team makes extensive usage of unit and functional testing [205]. Since DevOps relies heavily on automation, tracking how well the automated tests work is a good DevOps Metrics, and it is good to know how often code changes are causing tests to break [9, 175].

Metric expectations: This metric should trend upward or remain stable over time. The higher the percentage, the lower the risk of performing refactoring exercises. Three metrics also worth tracking are the number of test cases that have been developed, the percent of these that are automated, and the duration it takes to run different tests [49, 175].

How to improve: It is critical to track the Code Coverage percentage while transitioning from time-consuming peer review processes to automated ones [9]. Every new batch of code should be tested against the automated unit and integrity tests, and the percent of coverage should be tracked [49, 131].

M20. ***Work In Progress (WIP) /Load***. *Definition:* The amount of work that has been started but not yet completed. A similar number of incoming and outgoing work requests allows teams to balance their workloads [60, 64, 91].

Why it matters: The team can simply count the number of open issues of each type with the work-in-progress metric added to a dashboard (story, defect, task). When the number exceeds a certain threshold, it is time to stop taking on new projects and concentrate on those that have already begun. This improves the team's overall velocity [51, 82, 91, 93].

Metric expectations: This metric should have a ceiling and remain within normal levels over time. The load is the number of active or waiting work items in a value stream at any given time. Load is a metric that measures the utilization capabilities of value streams in relation to productivity in the process flow. This increases total velocity [29, 60, 64].

How to improve: The Toyota Production System of Lean Manufacturing [164] taught us that limiting work in progress (batch sizes) helps teams improve overall throughput. In other words, it is preferable to complete one project today rather than to work on ten projects and not complete any of them [20, 53, 60, 91, 93].

M21. ***Unplanned Work Rate (UWR)***. *Definition:* The unplanned work rate tracks unexpected efforts in relation to time spent on planned work. This exposes how much time is dedicated to unexpected efforts. Ideally, the **unplanned work rate (UWR)** should not exceed 25 percent [42, 119, 201].

Why it matters: A high UWR may reveal wasted efforts on unexpected errors that were likely not detected early in the workfloow. The UWR is sometimes examined alongside the **rework rate (RWR)**, which relates to the effort to address issues brought up in tickets [53, 119, 150, 201].

Metric expectations: This metric should trend down or remain stable over time. This is another crucial DevOps Metric that speaks the effective utilization of efforts. This calculates tracks the time spent on an unplanned work to that spent on a planned one. Most authors see it as the difference between acting on warning signs or having an unexpected outage [42, 146].

How to improve: This is the amount of time spent on tasks that were not originally planned. The UWR in standard projects should not exceed one-quarter of the work. A high UWR could reveal efforts that were squandered on unanticipated errors that were obviously missed early in the workflow. In addition to the RWR, which refers to the attempt to resolve issues raised in tickets, the UWR is an important indicator [42, 82, 107, 119, 146].

M22. ***Wait Time***. *Definition:* Wait Time is a "non-value-adding" process time, wherein the process is idle and waits for the next step. Also mentioned as queuing or waste, it is an estimate of the time that the work item spends idle in a non-productive state during its processing by the value stream. Wait time is in opposition to touch time when value is created [64, 91, 93, 123].

Why it matters: The goal is increasing efficiency, equivalent to touch time and opposed to wait time. These two metrics are related to time to develop a feature and time waiting until deploying it in production [25, 82, 93, 181]. It usually occurs when one stakeholder is waiting for another stakeholder to perform an action or hand over an artifact [174].

Metric expectations: This metric should trend down or remain stable over time. Code reviews, QA testing, security testing, and release cycles are examples of waiting time that value stream managers must reduce or eliminate to maximize customer and product value.[39, 82, 93, 123, 202]

How to improve: The wait time spent in "review" or "ready for release" delays delivery, because the development team might be unable to obtain feedback on the waiting stories [25]. There is a need to improve by using non-bottleneck resources, processing, triaging and limiting rework [15, 53, 82].

Finally, while answering this research question, it was found that some authors mention a set of important characteristics, in which the above listed main DevOps Metrics are considered useful [34, 49, 113], namely: (1) **Measurable**—metrics must have consistent and standardized values over time. (2) **Relevant**—should measure aspects that are important to the business. (3) **Actionable**—analysis should provide data for possible improvements. (4) **Reliable**—should be free of the influence of teams and team members. (5) **Traceable**—should point to a root cause rather than a general issue.

By further cross-checking these five qualities of good DevOps key performance indicators, it was found that they are based on the concepts of SMART [17], which technically provides some increased validity and guidelines to metrics for DevOps adoption success. But, organizations should also consider time, context, and resources when tracking these metrics. As well as using them per service and team to identify strengths and weaknesses [72]. Last, using a broader set of metrics allows organizations to quickly assess the effectiveness of DevOps capabilities. Focusing on business value creation via continuous improvement, identifying capability gaps toward achieving goals and objectives, and eliminating existing practices that undermine a strong DevOps culture and impede value flow to businesses and customers [204].

6 DISCUSSION AND FINDINGS

This article presents survey results, highlighting the need for DevOps metrics to enhance software delivery performance. It discusses aspects of metrics like categorization and competition, and introduces new topics on implementing these metrics and understanding changes in the context of DevOps metrics.

6.1 DevOps Metrics Categorization

It was found that DevOps Metrics categorization is still dispersed and only a few authors try to categorize metrics (39), represented in Table 6, therefore our categorization proposal shown in Section 1 could help achieve consensus. The same is observed while trying to understand what metrics are associated with each practice, capability, or principles of DevOps (25). This other missing piece of structured knowledge is intriguing and a possible source of investigation in forthcoming studies.

In Table 6 it is also shown a few highly relevant properties that this MLR has identified from the publications. The most important factor is that almost all authors state DevOps Metrics help improvements and efficiency (124) and a high number (93) associate the need for metrics with having pipeline automation in place. Like in the case of "value stream mapping" [74, 105], organizations have begun to adopt measurement techniques that will help identify areas that need improvement [80] and ensure produced software offers continuous improvements to the customer experience. To assess if DevOps efforts are successful, managers need consumable information based on a clear list of DevOps Metrics comparing similar value streams across a common set of KPIs [64, 82, 206].

Authors mention that selecting a categorization for DevOps Metrics is challenging [117, 166, 175, 206]. A fundamental issue with DevOps Metrics is that their significance is relative to a stakeholder's perspective. What the senior manager considers critical is likely to be somewhat different from what the software engineer producing the code considers important. Indeed, occasionally, the

Table 6. Six Publication Properties Identified from the MLR

Property	Publications	Total
Mentions that metrics help improvements and efficiency	[1–4, 7–11, 16, 19–21, 24, 26, 27, 29–31, 33–36, 39–43, 45–47, 49–55, 57, 58, 60, 61, 64, 67, 68, 70, 72, 74, 77, 79, 80, 82, 83, 88–91, 93, 97, 98, 100, 106–108, 113, 115, 116, 118, 119, 121–125, 128, 130, 132, 133, 137–140, 142–151, 154, 156–158, 160, 166, 168, 169, 171, 173, 175, 176, 178, 180, 183, 184, 190, 194, 196, 198–201, 204–210]	124
Relates Pipeline Automation to Metrics	[1–3, 9, 12, 14, 16, 29, 34–36, 39, 40, 42, 43, 46, 47, 49–51, 53–55, 57, 58, 60, 61, 64, 66–68, 70, 72, 74, 75, 77–80, 82, 83, 86, 88, 90, 93, 97, 100, 102, 107, 108, 113, 116, 118, 121, 123, 125, 127, 128, 131–133, 137, 139, 140, 144–151, 154, 158, 166, 168, 169, 171, 173, 175, 176, 178, 180, 183, 190, 196, 198, 201–204, 207, 208]	93
Organizes and Explains each Metric	[1, 2, 4, 7–10, 16, 21, 26–28, 30, 31, 33–36, 40, 42, 43, 45, 46, 49, 53, 54, 57, 58, 61, 64, 70, 72, 74, 75, 77, 78, 83, 86, 88–91, 98, 100, 102, 106, 107, 113, 119, 123, 128, 131, 132, 138–140, 142, 144–149, 151, 156, 157, 160, 163, 166, 173, 175, 180, 183–185, 188, 190, 198–200, 204–210]	87
Defines what are DevOps Metrics	[1, 3, 12, 14, 16, 24, 33, 35, 42, 46, 49, 51–55, 58, 67, 72, 74, 77, 78, 82, 89–91, 93, 97, 98, 100, 102, 106, 107, 118, 119, 123, 132, 139, 143, 145–149, 151, 171, 173, 176, 178, 180, 184, 194, 199, 200, 202, 208]	56
Mentions or Groups KPIs in DevOps context	[2, 3, 7, 10, 19, 24, 26, 29, 30, 34, 42, 45, 46, 49, 57, 77–79, 89, 90, 98, 100, 108, 113, 115, 116, 119, 121–124, 128, 132, 143, 156, 160, 163, 166, 173, 184, 187, 188, 196, 198, 200–203, 206–208, 210]	52
Tries to categorize metrics	[1, 8, 21, 24, 31, 33–36, 49, 51, 53, 54, 57, 98, 115–118, 125, 128, 132, 144–149, 151, 156, 157, 163, 166, 168, 175, 187, 188, 206, 210]	39
Mentions associated Practice, Capability or Principles	[1, 4, 11, 24, 35, 42, 51, 53, 54, 58, 72, 74, 77, 115, 121, 125, 144–149, 151, 166, 168]	25

importance of one measure is contingent on the values of other metrics: The metrics that matter are relevant to the observer's orientation and even to the values of other metrics.

Gathering metrics effectively is also another debated problem. Forsgren et al. [55] mention that it is best to start by capturing a system baseline with survey measures while continuing to build out system-based metrics, which should normally use data that comes from the various systems of record in the software delivery value stream. Both metrics have their limitations, but if used in complement, organizations can gain a superior view of their software delivery value chain and DevOps transformation work. In the various State of DevOps Reports [35, 144–149, 151] a few grouped important metrics have already been used over the years, namely, some IT performance metrics. DevOps metrics typically measure throughput, stability, or quality [107], while quantifying a faster cadence (efficiency) and value addition (effectiveness) [51, 91]. Metrics also enable DevOps teams to monitor and analyze collaborative workflows, as well as track progress toward high-level goals such as higher quality, quicker release cycles, and improved application performance [74]. Specifically, in the State of DevOps 2019 report[35], metrics are mentioned to mirror the effectiveness of the development and delivery process, and they can be grouped in terms of *throughput* and *stability*. The throughput of the software delivery process is measured using *lead time for code changes* from check-in to release, along with *deployment frequency*. As for stability, it can be measured using *mean time to restore* a system and *change fail rate*, a measure of the quality of the release process. They provide a solid basis for an organization's metrics activities [67].

However, these high-level metrics can be drilled down into a more refined state or expanded to include others like *Service Availability and Uptime*—the time an application is available, *Defect Escape Rate*—the number of defects that are found during a given unit of time, or *Mean Time To Detection*—the average time between when a problem arises in production and when it is detected. As part of this research, the main metrics are being expanded, listed, and defined in Sections 5.1 and 5.2, as mentioned in the objectives.

As seen in Section 1 there are a few organizational concepts in literature structuring DevOps Metrics into the categories of *Organizational Culture* [38, 53, 84, 103, 118, 152], *Operational Performance* [23, 84, 109, 165], *Business Focus* [73, 85, 87, 114, 135], and *Incremental Change* [76, 186, 217]. In the literature review's coding done over the found publications, shown in Table 6 it was observed that each of the main DevOps metrics found in Section 5.1 are being mentioned or grouped into key indicators matching the category proposal.

Nevertheless, it is known that structuring key indicators is an art to determine which are most relevant for the organization with DevOps objectives [24, 121, 166] in mind. Therefore, in Figure 8 this study summarizes and organizes DevOps Metrics into the four KPIs categories: (1) **Business KPIs**, that have a direct impact on business goals. (2) **Change KPIs**, that reflect engineering's capacity to improve applications, infrastructure, or services. (3) **Operational KPIs**, that reflect a team's operational excellence. (4) **Cultural KPIs**, which are used to assess an organization based on Westrum's Organizational Culture Measures.

As a result, there are ten *Change KPIs* focused on measuring the different aspects of development and delivery, while there are nine *Operational KPIs* focused on reliability, stability, and supporting applications running in production. These dimensions seek to collect data about service delivery and operationality [24, 166]. Some of these indications may be high level for new DevOps initiatives. This could be due to the extra time required to adopt and implement new processes, as well as exposing and resolving current technological debt and waste [53, 204]. The *Business KPIs* are focused on providing customers the created value, getting feedback and making sure there is traffic and users for the system. These measurements help quantify the impact of DevOps on business objectives like increased customer loyalty and time to market [48, 115, 186, 206].

DevOps *Cultural KPIs* measures the cultural impact of DevOps implementation to address the cultural gaps that have traditionally existed between developers, operational administrators, and other engineers. This can be measured using employee surveys and other employee engagement metrics. Team happiness, meeting efficiency, and learning opportunities are examples of enablers for this category aiming to capture the organizational culture and its impacts [146, 195, 204]. Authors largely agree that culture is an important ingredient of DevOps. The challenge for most IT leaders is defining and communicating a vision of beneficial culture for their organizations, and then facilitating the changes needed to achieve that.

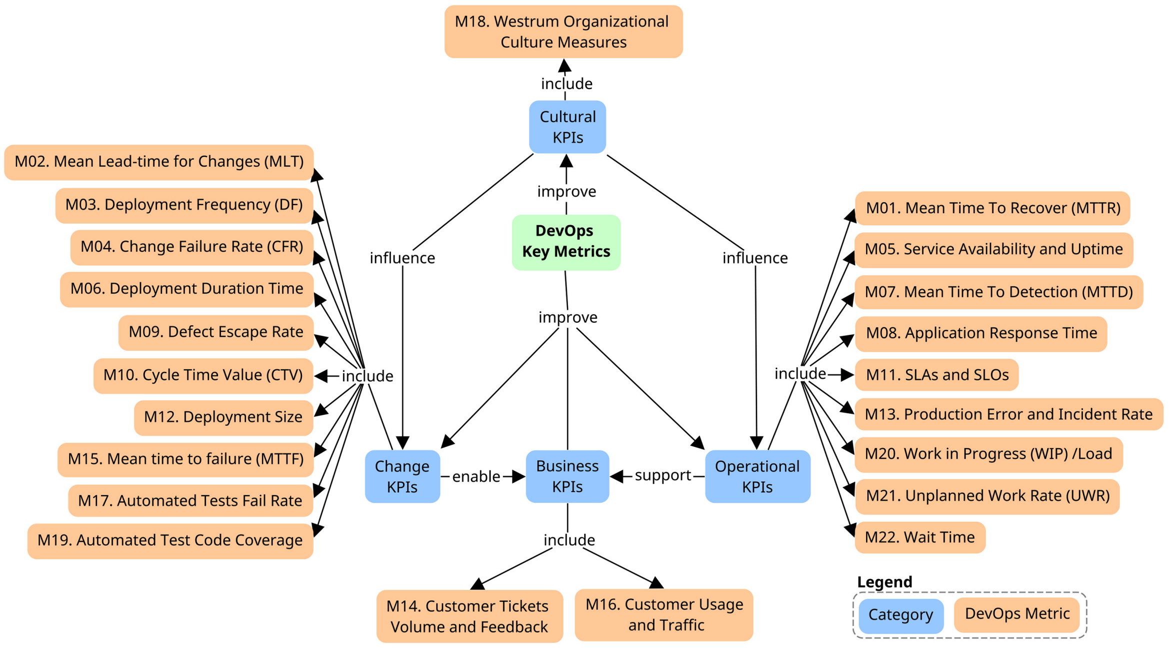

M18. Westrum Organizational Culture Measures
include
Cultural KPIs
improve
DevOps Key Metrics
influence
influence
improve
M02. Mean Lead-time for Changes (MLT)
M03. Deployment Frequency (DF)
M04. Change Failure Rate (CFR)
M06. Deployment Duration Time
M09. Defect Escape Rate
M10. Cycle Time Value (CTV)
include
M12. Deployment Size
M15. Mean time to failure (MTTF)
M17. Automated Tests Fail Rate
M19. Automated Test Code Coverage
Change KPIs
enable
Business KPIs
support
Operational KPIs
include
M01. Mean Time To Recover (MTTR)
M05. Service Availability and Uptime
M07. Mean Time To Detection (MTTD)
M08. Application Response Time
M11. SLAs and SLOs
M13. Production Error and Incident Rate
M20. Work in Progress (WIP) /Load
M21. Unplanned Work Rate (UWR)
M22. Wait Time
include
M14. Customer Tickets Volume and Feedback
M16. Customer Usage and Traffic
Legend
Category
DevOps Metric

Fig. 8. DevOps metrics categorization and relation of concepts.

The State of DevOps report 2015, an annual survey based on the responses from more than 20,000 technical professionals, mentions that many companies claim to be data-driven, in the sense of gathering a lot of data, but relatively few can really use that data to make educated choices [146]. Thus, posing the questions if DevOps analytics is being done correctly and regularly and if any action is being taken on them. This MLR proposes to improve performance clarity by linking measurements to systematic actionable goals and to turn measurement data into meaningful, visible information that provides feedback to leadership and teams on important quality, performance, and productivity criteria.

6.2 Competition and Vanity Metrics

Following the reporting process in Section 5.1, it is noticed that metrics among the not strongly mentioned, therefore not included in the final list of 22 metrics, are those that could lead to internal competition. Because if the top performers are the "winners" and everyone else "loses," then communication or collaboration within and between teams is difficult to expect, which should be a top DevOps capability [51, 53, 80, 82, 118, 145, 148, 149]. Metrics that are based on competition among team members or between teams go against DevOps values [13, 217]. Teams will become obsessed with improving metrics rather than identifying and resolving real problems. Examples are builds per day, number of code commits, or features released per quarter seen in Appendix B.

Finally, there could also be some vanity metrics [34, 46, 49, 53, 82, 113, 146, 156, 204]. These metrics may even indicate some ability, but they do not accurately reflect business effectiveness. For example, the number of lines of code written each week is meaningless, because code can vanish completely during refactoring, and less code is sometimes better for the organization. The number of builds per day is irrelevant unless each build adds value to the end-user experience [34, 49, 113].

6.3 Implementing DevOps Metrics in Organizations

A more involved, emerging question not yet fully answered by the literature is how an organization could put these DevOps metrics into practice. It seems the results are incomplete and even

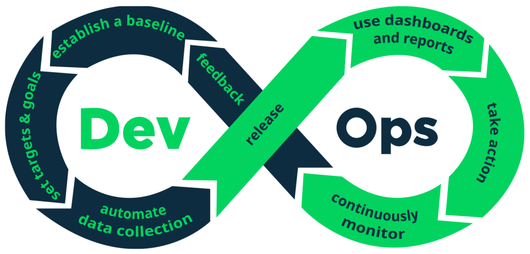


Fig. 9. DevOps metrics in practice infinity loop.

dispersed on this matter. However, it can be discussed within the scope of a few related studies that have already touched on this topic. Forsgren et al. [56] states that measurement is the most important part of making a software value stream that works, while Snyder et al. [179] recommends eliminating measurement silos and aligning analytics from all enterprise tools, to have a complete picture of ongoing transformation. Therefore, in measuring and improving software delivery performance, both the measurement method and the results are important to be exposed [53]. Sallin et al. [168] analyzed how DevOps metrics could be used to measure software delivery by having the metrics automatically calculated and shown to a team of practitioners in production.

Given this context, it is intended in Figure 9 to contribute more to the discussion by suggesting a practical process based on the DevOps infinity loop [80]. This is a conceptual representation of the continuous feedback and improvement process that is central to the DevOps philosophy [6, 118, 125]. It is proposed that the steps for putting the main DevOps metrics into practice can be represented as a series of nodes or stages, with each stage connected to the next, forming a feedback loop.

The proposed steps to put DevOps metrics into practice are described as follows:

1. **Establish a baseline**: Organizations should start by setting a baseline for each metric by collecting data over a period of time to gain a clear understanding of current performance and identify areas for improvement [19, 80, 140]. Set a starting point by measuring the current state of processes and systems, providing a basis for comparison and progress tracking [41, 75, 178]. Identify which metrics are most important and establish a baseline for future improvements [57, 140]. This step is the first in the DevOps infinity loop, establishing the current state of the system.

2. **Set targets and goals**: Governance should set clear goals and targets for each DevOps metric based on their desired outcomes, such as reducing downtime, increasing efficiency, or improving customer satisfaction [2, 35, 56]. These goals should align with the organization's business objectives and be focused on improving business outcomes [80, 92, 168]. This aligns with the second step in the loop, where the goals for improvement are set.

3. **Automate data collection**: The data collection process should be automated by using tools such as monitoring, logging, observability platforms, and online surveys to ensure accurate and up-to-date data [44, 55, 144]. Automating the data collection, analysis, and monitoring process saves time, reduces human error, and improves the accuracy of metrics [14, 57]. Survey data can show important cultural, perceptual and whole-person information that cannot be collected through system measurement[56, 186]. This step aligns with the third step in the DevOps infinity loop, where automation is prepared to collect data and gather feedback.

4. **Release**: The release stage of the DevOps infinity loop involves making new code or updates available to customers and stakeholders [50, 51, 115, 116]. For this specific process, the data collected is used to improve and release new features, updates, and improvements to systems and processes to maintain continuous feedback [11, 118, 121]. The book "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation" [80] highlights

the importance of frequent software releases and how it can be done in a reliable and efficient manner.

5. **Use dashboards and reports**: Utilize dashboards and reports to visualize DevOps metrics, identify trends and patterns, track progress toward targets and goals, and assess performance [106, 196, 197, 208]. The book "Continuous Delivery and DevOps: A Quickstart guide" [186] explains how dashboards and reports can be used to track and improve the performance of a DevOps organization. There are also academic experiments on using visual aids such as dashboards and reports [24] and authors [13, 69, 112] recommend using visualizations to communicate metrics to team members and stakeholders [52]. This step aligns with the fifth step in the DevOps infinity loop, where data visualization is used to make sense of the data and identify areas for improvement.

6. **Take action**: Leverage the insights from the metrics to pinpoint opportunities for improvement and take action [53, 58, 149], such as process changes, tooling changes, or adding team members [179]. Use data to make informed decisions about process and technology improvements for systems, processes, and customer experience [44, 80]. In the book "The Phoenix Project," Kim et al. [92] provide a view of how an organization can take action to improve its IT performance through the implementation of DevOps capabilities. This step aligns with the sixth step in the DevOps infinity loop, where you act based on the insights gained from the data.

7. **Continuously monitor**: Continuously monitor metrics and make adjustments to improve DevOps processes, practices, and collaboration [18]. Regular monitoring helps identify potential issues and track progress against established objectives [10, 44]. Use insights from metrics to improve systems, processes, and customer experience by continuously monitoring customer feedback [74, 104, 109, 151]. This step aligns with the seventh step in the DevOps infinity loop, where you continuously monitor the system to ensure that improvements are sustained.

8. **Feedback and Communicate**: Share metric data and insights with all involved to promote transparency and collaboration toward common goals [14, 78, 94]. Encourage feedback and communication between teams, stakeholders, and customers to identify new improvement opportunities and foster a data-driven, continuous improvement culture [69, 88]. Share results with relevant stakeholders to use data to drive continuous improvement [80]. This aligns with the eighth step in the loop, where feedback is gathered and results are communicated to stakeholders.

These guidelines propose a structured and comprehensive approach to DevOps metrics implementation. The stages in Figure 9 are in a specific order to ensure that each step builds upon the previous one and provides the necessary foundation for the next step.

6.4 Change in the Context of DevOps Metrics

Change is a concept that has a few different interpretations depending on the DevOps metric being evaluated and the context being observed. During this research, it was noticed that what constitutes change can vary for some metrics like M02. MLT is a metric that shows how much time is needed to implement a change in the full development cycle process [198]. But it can also be how long it takes to go from code committed to code running successfully in production [98, 198]. Which also raises the discussion: does the variation of change across different contexts matter?

Expanding from the current study to a wider state of the art, there are a few papers that, despite not explicitly, address the question of what constitutes a change in the context of DevOps metrics. They provide some relevant information. Gupta et al. 2017 refers to Continuous Delivery as the ability of the system to release changes or fixes to the production environment, also suggesting that a framework can be used to assess this and other aspects of DevOps implementation [71]. Lwakatare et al. 2015 suggests that measurement in DevOps is achieved by measuring the effort of the software process beyond QA using real-time performance [111]. Forsgren et al. 2017 implies that it is challenging for teams to understand the wider dynamic context in which they operate

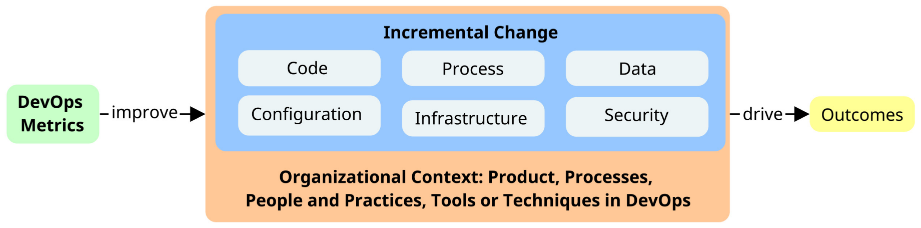


Fig. 10. DevOps metrics and incremental changes in the DevOps context.

due to their inability to measure change over time in relation to changes in the rest of the industry [58].

To better organize this discussion, it is proposed that changes in DevOps, gathered from literature, can be broadly divided into these categories: (1) **Code changes**: These are modifications to the source code of a software application [18, 80, 92, 99, 103, 174]. (2) **Configuration changes**: These are modifications to the configuration files, settings, or environment variables that govern how a software application operates [18, 41, 82, 84, 92, 149]. (3) **Infrastructure changes**: These are modifications to the underlying infrastructure that supports a software application, including servers, databases, and network components [13, 82, 99, 104, 186]. (4) **Process changes**: These are modifications to the development and delivery processes that support a software application, including changes to continuous integration, testing, and deployment workflows [53, 82, 84, 151, 155, 197]. (5) **Data changes**: These are modifications to the data that is stored, processed, or used by a software application, including changes to database schema and data structures [1, 80, 121, 174, 186]. (6) **Security changes**: These are modifications to the security measures that protect a software application and its underlying infrastructure, including changes to access controls, encryption algorithms, and network security settings [53, 93, 149, 150, 174].

Each of these types of changes can impact the stability, reliability, and security of a software application, and it is important to manage them carefully to ensure a high-quality software delivery process [18, 93]. In DevOps, the emphasis is on automating and streamlining changes to minimize the risk of errors and improve the speed and efficiency of software delivery [35, 53, 168]. To contextualize change from multiple perspectives, Petersen et al. proposed a checklist, given that context is critical in software engineering [136]. The context of incremental change seen in Figure 10 is also not often discussed or considered when approaching DevOps metrics. For instance, in the change category items mentioned in Section 6.1, does the variation of change across different contexts matter?

DevOps performance includes measuring incremental change, introduced in Section 1, which can be placed within context facets [136]. Namely, product, processes, practices and techniques, people, organization, and market facets. The *product* context considers the size and complexity of changes and requirements for integration [15, 84]. The *process* context involves the DevOps pipeline and procedures in place for measurement [80, 174]. The *practices and techniques* context covers tools and methodologies used in DevOps [103, 118]. The *people* context refers to team members' roles, responsibilities, and skills [115, 214]. The *organizational* context covers the company's structure and culture [153, 177, 186], and the *market* context refers to competitive pressures and customer demands [101, 155]. These domains impact the efficiency of a pipeline, use of automation, quality of collaboration and communication, availability of resources, level of organizational support, and ability to respond to market conditions and customer requirements.

Therefore, variation of change across different contexts matters, since the success of organizational change depends on the specific change context or the environment in which it is implemented. Assessing internal and external factors can help identify DevOps enablers and challenges, leading to specific actions [111, 114]. Analysis of change context involves gathering diverse perspectives and acting on the findings [118]. The context of change in DevOps is related to the change management capability and to how well an organization handles these contexts to successfully manage change [13, 151]. Every time the business wants a change, there is an investment in the development process to deliver that change [186, 199]. Therefore, for the example of Mean Lead-time for Changes it is indeed dependent on what the context is and to what we are changing, code, configurations, and infrastructure.

Furthermore, in the context of the organizational process, MLT will be influenced by the need to respond quickly to changing market conditions and customer requirements. Leading to measure the throughput of the software delivery process using lead time of code changes from check-in to release, along with deployment frequency [1, 35]. Last, in the context of DevOps metrics, change refers to the modifications made to systems, processes, or practices based on the data and insights gained from the metrics [111]. These changes aim to improve the performance, efficiency, and customer satisfaction of the systems and processes being monitored. Change may involve fixing issues, implementing new tools, or making process changes to address any challenges or areas for improvement identified through the metrics. The goal of these changes is to continuously improve the overall DevOps process and capabilities [116].

7 CONCLUSION

This study has brought important contributions to both academia and industry on the DevOps topic. In summary, an MLR was run on DevOps Metrics. To find literature, Google search, Scopus, Web of Science, IEEE, ACM, and EBSCO were utilized, and 139 publications were recognized as relevant to this research area: (1) In this literature review, a definition of DevOps Metric is proposed. (2) Moreover, 22 main DevOps Metrics were identified and categorized. (3) DevOps Metrics were discussed and categorized into Business, Change, Operational, and Cultural KPIs. (4) It is discussed why, how to improve, and expectations for each metric. (5) Benefit characteristics of main DevOps Metrics are exposed. (6) Identified four top metrics MTTR, MLT, DF and CFR. (7) It was found that the community agrees on the top four metrics and is focusing on them. (8) Discussed how organizations could put the main DevOps metrics into practice. (9) Discussed what constitutes a change in the context of DevOps Metrics.

It has been researched that these top four key metrics have expected improvement outcomes from DevOps adoption. MTTR determines the mean of the time required to recover or restore service from a failure in production. MLT indicates how long it takes for a change to go from code committed to code successfully running in production. DF ascertains how often changes are deployed to production. CFR measures how often a change in production fails and must be immediately remedied. According to this MLR, academic studies still demonstrate limited research in this area. However, the industry shows a rising interest in the usage of DevOps Metrics. As a result, the employment of DevOps Metrics should be thoroughly explored due to the possible influence on businesses. In this regard, it would be worthwhile to perform a study not only on DevOps Metrics but also on their relationship to DevOps capabilities, practices, and outcomes.

For some of the most referenced metrics, M02, M03, M04, M06, and M12, there appears to be a relation to the continuous delivery DevOps capability in References [35, 72, 80, 108, 118, 154], which could be of interest to conduct more investigation in future research toward exploring the relations to delivery practices. As research synthesis, the goal was to look into DevOps Metrics, their definition, importance and categorization, without debating in depth how they are implemented. However, as it can be seen, there are obvious indicators that more study may be done from here on than take advantage of eliciting measurements into an organized format.

7.1 Future Work

What was discovered will help further research so that future studies can determine if these metrics remain the most prevalent and may be researched further. An example of this are the following questions: How do DevOps practices and capabilities relate to metrics? When these metrics are put into action, how will the results vary? Which ones will have the most impact in which kind of organizations? Attempts to understand what metrics are associated with each practice, capability, or principles of DevOps were identified in 26 publications seen in Table 6, but this relation is still unclear, and no consensual link was found, leading to an opening of upcoming work in this area. There is space for exploring the practical aspects of implementing the identified metrics and their impact on DevOps capabilities. Also, on the organizational front, we still miss knowing what metrics are already used by which industries? What organizational aspects have more effect on each of the main key metrics? Can we control these aspects? How? Would it be possible to expose the metrics, capabilities, and their influencing factors in information systems to support management decisions [101]? However, there is still debate going on [29, 47, 50, 151] regarding: Should all of these metrics be measured proactively? Which metrics can be measured automatically? What metrics may only be measured using surveys? Which are valuable questions to explore. Finally, improving measuring the software delivery process is relevant and pursued as seen in Section 5. DevOps Metrics should aim to quantify the right elements to understand if DevOps is working.

7.2 Limitations

Regarding Limitations, this study is based on multivocal literature, and the majority of the material has not been subjected to the rigorous peer-review process that academic research is normally subjected to. Instead, the literature has included blogs, white papers, and reports. To mitigate the impact of this danger, it was chosen to design the review procedure using the recommendations given by Garousi et al. [63] and to conduct each step using this method. It is also acknowledged that sources can change, which is why, during the peer-review process, any inaccessible sources were replaced with alternative URLs for the same content. Over time, industry reports influence other sources. But, even in academic literature, multiple voices risk influence. The MLR aims to represent academia, practitioners, and the DevOps community. To mitigate validity threats of sources published in software companies' blogs, metrics with less than 10 references were excluded, and information is cross-verified with independent sources, as shown in Figure 7 and Section 4, as elaborated upon in later sections. Similarly, to address limitations of basic metrics information, a more in-depth analysis, using peer-reviewed sources, was done for a more comprehensive understanding of the metrics. Search keywords and search engines used might lead to an incomplete selection of primary sources. Formal searching utilizing specific keywords was carried out and specific source code was used to reduce the risk of not discovering all relevant studies and increasing the reliability of replicating this research. Last, also a restriction was the inclusion of English-only articles, which may exclude significant studies in other languages.

REFERENCES

[1] Google Accelerate. 2021. *2021 State of DevOps Report.* Technical Report GC2021. Google Cloud. Retrieved from https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

[2] Khalil Ahmad. 2020. DevOps KPIs and "Design for Failure." Retrieved from https://www.linkedin.com/pulse/devops-kpis-design-failure-khalil-ahmad. Accessed on 2022-01-22.

[3] AlertOps. 2018. MTTD vs MTTF vs MTBF vs MTTR - Resolve Major IT Incidents Quickly. (2018). Retrieved from https://alertops.com/mttd-vs-mttf-vs-mtbf-vs-mttr/. Accessed on 2022-01-22.

[4] Altexsoft. 2021. DevOps Metrics: Mean Time to Failure, Server Uptime, Mean Time Between Failures, Mean Time to Recovery, and More. Retrieved from https://www.altexsoft.com/blog/devops-metrics/. Accessed on 2022-01-22.

[5] Ricardo Amaro. 2021. *DevOps Capabilities and Metrics*. Ph.D. Dissertation. IST - Information and Enterprise Systems (MISE). Retrieved from https://fenix.tecnico.ulisboa.pt/cursos/mise/dissertacao/283828618790759

[6] Ricardo Amaro, Ruben Pereira, and Miguel Mira da Silva. 2022. Capabilities and practices in DevOps: A multivocal literature review. *IEEE Trans. Softw. Eng.* 1 (2022), 20. https://doi.org/10.1109/TSE.2022.3166626

[7] Appdynamics. 2020. DevOps Metrics and KPIs: How To Measure DevOps? Retrieved from https://www.appdynamics.com/topics/devops-metrics-and-kpis. Accessed on 2022-01-22.

[8] Emily Arnott. 2021. DevOps Metrics | How to Measure What Matters. Retrieved from https://www.blameless.com/devops/devops-metrics. Accessed on 2022-01-22.

[9] Prashant Arora. 2015. Measuring the Success of DevOps—Prashant Arora's Blog. Retrieved from https://aroraprashant.wordpress.com/2015/04/14/measuring-the-success-of-devops/. Accessed on 2022-01-22.

[10] Rashed Azzam. 2021. 8 Proven DevOps Metrics: Effectively Measure and Optimize Your DevOps Success. Retrieved from https://www.vardot.com/en-us/ideas/blog/8-proven-devops-metrics-effectively-measure-and-optimize-your-devops-success. Accessed on 2022-01-22.

[11] Sher Badshah, Arif Ali Khan, and Bilal Khan. 2020. Towards process improvement in DevOps: A systematic literature review. In *Proceedings of the 24th Evaluation and Assessment in Software Engineering Conference (EASE'20)*. ACM, 427–433. https://doi.org/10.1145/3383219.3383280

[12] Michael Baldani. 2019. DORA Metrics—Getting on the Bandwagon. Retrieved from https://www.cloudbees.com/blog/dora-metrics-getting-bandwagon. Accessed on 2022-01-22.

[13] Len Bass, Ingo Weber, and Liming Zhu. 2015. *DevOps: A Software Architect's Perspective*. Addison-Wesley, New York. Retrieved from http://my.safaribooksonline.com/9780134049847

[14] Mary "Lisa" Williams Bates and Enrique I. Oviedo. 2021. Software reliability in a DevOps continuous integration environment. In *Proceedings of the Annual Reliability and Maintainability Symposium (RAMS'21)*. IEEE, 4. https://doi.org/10.1109/RAMS48097.2021.9605768

[15] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Inc., Google. Retrieved from https://landing.google.com/sre/sre-book/toc/

[16] Marco Bizzantino. 2019. 4 Fundamental Metrics to Measure DevOps Performances. (2019). Retrieved from https://www.kiratech.it/en/blog/4-fundamental-metrics-to-measure-devops-performances. Accessed on 2022-01-22.

[17] May Britt Bjerke and Ralph Renger. 2017. Being smart about writing SMART objectives. *Eval. Program Plan.* 61 (Apr. 2017), 125–127. https://doi.org/10.1016/j.evalprogplan.2016.12.009

[18] David N . Blank-Edelman. 2018. *Seeking SRE: Conversations About Running Production Systems at Scale*. O'Reilly Media, Inc..

[19] Robert Bobbett. 2018. DevOps Value: How to Measure the Success of DevOps. Retrieved from https://www.fpcomplete.com/blog/devops-value-how-to-measure-the-success-of-devops/. Accessed on 2022-01-22.

[20] Kasper de Boer. 2016. 2 Most Important DevOps Metrics Tools. Retrieved from https://labs.sogeti.com/the-two-most-important-metrics-for-devops/. Accessed on 2022-01-22.

[21] Dnyaneshwar Borase. 2021. What You Need to Know About DevOps Metrics in Jira? | Addteq Blog. Retrieved from https://web.archive.org/web/20220401230729/https://addteq.co.in/blog/what-you-need-to-know-about-devops-metrics-in-jira/. Accessed on 2022-01-22.

[22] Charles Border. 2019. Development of a configuration management course for operations students. In *Proceedings of the 20th Annual SIG Conference on Information Technology Education*. ACM, New York, NY, 41–41. https://doi.org/10.1145/3349266.3351360

[23] Andreas Brunnert, Andre van Hoorn, Felix Willnecker, Alexandru Danciu, Wilhelm Hasselbring, Christoph Heger, Nikolas Herbst, Pooyan Jamshidi, Reiner Jung, Joakim von Kistowski, Anne Koziolek, Johannes Kroß, Simon Spinner, Christian Vögele, Jürgen Walter, and Alexander Wert. 2015. Performance-oriented DevOps: A research agenda. Retrieved from https://arxiv.org/abs/1508.04752. https://doi.org/10.48550/ARXIV.1508.04752

[24] Francisco João Lúcio Bruno. 2021. *DevOps Dashboard*. Ph.D. Dissertation. ISCTE-IUL. Retrieved from https://repositorio.iscte-iul.pt/handle/10071/24112

[25] Lianping Chen. 2017. Continuous delivery: Overcoming adoption challenges. *J. Syst. Softw.* 128 (June 2017), 72–86. https://doi.org/10.1016/j.jss.2017.02.013

[26] Cigniti. 2016. Top 6 DevOps Metrics That Enterprise Dashboards Should Capture. (2016). Retrieved from https://www.cigniti.com/blog/6-devops-metrics-for-enterprise-dashboards/. Accessed on 2022-01-22.

[27] Lauma Cīrule. 2019. Analyze DevOps Metrics With eazyBI. Retrieved from https://eazybi.com/blog/analyze-devops-metrics-with-eazybi. Accessed on 2022-01-22.

[28] CloudNative. 2021. DevOps Metrics: How to Monitor Performances Optimally. Retrieved from https://blog.mia-platform.eu/en/devops-metrics-how-to-monitor-performances-optimally. Accessed on 2022-01-22.

[29] Cprime. 2021. DevOps Metrics to Monitor Software Development—Cprime. Retrieved from https://www.cprime.com/resources/blog/devops-metrics-to-monitor-software-development/. Accessed on 2022-01-22.

[30] Royal Cyber. 2019. DevOps KPIs to Measure Success. Retrieved from https://www.royalcyber.com/blog/devops/devops-kpis-to-measure-success/. Accessed on 2022-01-22.

[31] Matthew David. 2021. DevOps Metrics: How to Measure Metrics for Your Devops Team. Retrieved from https://www.simplilearn.com/devops-metrics-used-to-measure-devops-team-article. Accessed on 2022-01-22.

[32] Patrick Debois. 2011. DevOps from a sysadmin perspective. *Login—Usenix Mag.* 36, 6 (2011), 3.

[33] Tiempo Development. 2020. A Guide To Measuring DevOps Success and Proving ROI. Retrieved from https://www.tiempodev.com/blog/measuring-devops/. Accessed on 2022-01-22.

[34] Devopedia. 2019. DevOps Metrics. Retrieved from https://devopedia.org/devops-metrics. Accessed on 2022-01-22.

[35] DevOps Research and Assessment (DORA). 2019. *State of DevOps 2019—DORA*. Technical Report DORA2019. DORA. Retrieved from https://services.google.com/fh/files/misc/state-of-devops-2019.pdf.

[36] DevOpsEnterpriseSummit. 2017. Featured Resource: Metrics for DevOps Initiatives—IT Revolution. Retrieved from https://itrevolution.com/devops-resource-metrics/. Accessed on 2022-01-22.

[37] Jessica Díaz, Rubén Almaraz, Jennifer Pérez, and Juan Garbajosa. 2018. DevOps in practice—An exploratory case study. In *Proceedings of the 19th International Conference on Agile Software Development*. 3. https://doi.org/10.1145/3234152.3234199

[38] Jessica Díaz, Daniel López-Fernández, Jorge Pérez, and Ángel González-Prieto. 2021. Why are many businesses installing a DevOps culture into their organization? *Empir. Softw. Eng.* 26, 2 (2021), 50. https://doi.org/10.1007/s10664-020-09919-3

[39] Digital.ai. 2019. 4 DevOps Metrics to Improve Delivery Performance on Vimeo. Retrieved from https://vimeo.com/331032185. Accessed on 2022-01-22.

[40] digital.ai. 2021. 10 DevOps Metrics You Should Know. Retrieved from https://digital.ai/resources/infographic/10-devops-metrics-you-should-know. Accessed on 2022-01-22.

[41] Damian Dingley. 2019. 4 Key Metrics for DevOps Success Video. Retrieved from https://www.veracitysolutions.com/4-key-metrics-for-devops-success. Accessed on 2022-01-22.

[42] Bojana Dobran. 2019. 15 DevOps Metrics and Key Performance Indicators (KPIs) To Track. Retrieved from https://phoenixnap.com/blog/devops-metrics-kpis. Accessed on 2022-01-22.

[43] Paul Duvall. 2018. Measuring DevOps Success with Four Key Metrics | Stelligent. Retrieved from https://stelligent.com/2018/12/21/measuring-devops-success-with-four-key-metrics/. Accessed on 2022-01-22.

[44] Paul M. Duvall, Steve Matyas, and Andrew Glover. 2007. *Continuous Integration: Improving Software Quality and Reducing Risk* (1st ed.). Addison-Wesley Professional, Upper Saddle River, NJ.

[45] Aliza Earnshaw and Puppet. 2013. 5 KPIs That Make the Case for DevOps. Retrieved from https://puppet.com/blog/5-kpis-make-case-for-devops/. Accessed on 2022-01-22.

[46] Mark Edwards. 2019. Measuring for Success—Change or Hold—DevOps. Retrieved from https://web.archive.org/web/20210415075956/https://www.tesm.com/resources-blog-measuring-for-success-should-you-change-or-should-you-hold-devops-pt-5/. Accessed on 2022-01-22.

[47] Roy Edwards. 2021. Research Highlights Challenges of Salesforce DevOps in 2020. Retrieved from https://www.enterprisetimes.co.uk/2021/02/16/research-highlights-challenges-of-salesforce-devops-in-2020/. Accessed on 2022-01-22.

[48] João Faustino, Daniel Adriano, Ricardo Amaro, Rubén Pereira, and Miguel Mira da Silva. 2022. DevOps Benefits: A systematic literature review. *Softw.: Pract. Exper.* 52, 9 (2022), 1905–1926. https://doi.org/10.1002/spe.3096

[49] Vladimir Fedak. 2020. DevOps Metrics: What to Track, How and Why Do It. Retrieved from https://medium.com/@FedakV/devops-metrics-what-to-track-how-and-why-do-it-e08dc6864eab. Accessed on 2022-01-22.

[50] Flosum. 2021. Keys to Improve Salesforce DevOps Efficiency - Flosum - Continuous Integration, Release Management. Retrieved from https://flosum.com/keys-to-improve-salesforce-devops-efficiency/. Accessed on 2022-01-22.

[51] Nicole Forsgren. 2015. Metrics for DevOps Initiatives. Retrieved from https://itrevolution.com/articles/devops-resource-metrics/. Accessed on 2022-01-22.

[52] Nicole Forsgren. 2017. How to Use Metrics, Measurement to Drive DevOps. Retrieved from https://techbeacon.com/devops/how-use-metrics-measurement-drive-devops. Accessed on 2022-01-22.

[53] Nicole Forsgren, Jez Humble, and Gene Kim. 2018. *Accelerate: The Science of Lean Software and Devops: Building and Scaling High Performing Technology Organizations*. IT Revolution. Retrieved from https://itrevolution.com/accelerate-book/.

[54] Nicole Forsgren, Jez Humble, Gene Kim, A Brown, and N Kersten. 2018. Accelerate state of DevOps 2018 strategies for a new economy. *Report. DevOps Res. Assess. (DORA)* 1 (2018), 78.

[55] Nicole Forsgren and Mik Kersten. 2017. DevOps Metrics: Your Biggest Mistake Might Be Collecting the Wrong Data. *Queue* 15, 6 (Dec. 2017), 19–34. https://doi.org/10.1145/3178368.3182626

[56] Nicole Forsgren and Mik Kersten. 2018. DevOps Metrics. *Commun. ACM* 61, 4 (Dec. 2018), 44–48. https://doi.org/10.1145/3159169

[57] Nicole Forsgren, Marcus Rothenberger, Jez Humble, Jason Thatcher, and Dustin Smith. 2020. A taxonomy of software delivery performance profiles: Investigating the effects of devops practices. In *Proceedings of the 26th Americas Conference on Information Systems (AMCIS'20)*. 5.
[58] Nicole Forsgren, Monica Chiarini Tremblay, Debra VanderMeer, and Jez Humble. 2017. DORA Platform: DevOps assessment and benchmarking. In *Designing the Digital Transformation*, Alexander Maedche, Jan vom Brocke, and Alan Hevner (Eds.). Springer International Publishing, Cham, 436–440. https://doi.org/10.1007/978-3-319-59144-5_27
[59] Breno B Nicolau de França, Helvio Jeronimo, Guilherme Horta Travassos, Breno B. Nicolau de França, Helvio Jeronimo, and Guilherme Horta Travassos. 2016. Characterizing DevOps by hearing multiple voices. In *Proceedings of the 30th Brazilian Symposium on Software Engineering*, E. S. DeAlmeida (Ed.). Unicesumar; Colivre; Espweb; Tasa Eventos, New York, NY, 53–62. https://doi.org/10.1145/2973839.2973845
[60] Ann Marie Fred and Craig Cook. 2021. 6 Proven Metrics for DevOps Success | TechBeacon. Retrieved from https://techbeacon.com/devops/6-proven-metrics-devops-success. Accessed on 2022-01-22.
[61] Philip Gallagher. 2020. Tracking Success in DevOps Pipelines. Retrieved from https://blog.goodelearning.com/subject-areas/devops/how-to-measure-success-in-devops/. Accessed on 2022-01-22.
[62] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2016. The need for multivocal literature reviews in software engineering. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE'16)*. ACM Press, New York, NY, 6. https://doi.org/10.1145/2915970.2916008
[63] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Info. Softw. Technol.* 106 (Feb. 2019), 101–121. https://doi.org/10.1016/j.infsof.2018.09.006
[64] John Gelo. 2020. DevOps Metrics Matter: Why, Which Ones, and How - HCL SW Blogs. Retrieved from https://blog.hcltechsw.com/accelerate/devops-metrics-matter-why-which-ones-and-how-2/. Accessed on 2022-01-22.
[65] Gene Kim and IT Revolution. 2012. The Three Ways: The Principles Underpinning DevOps. Retrieved from https://itrevolution.com/the-three-ways-principles-underpinning-devops/. Accessed on 2022-01-22.
[66] Tom Gilmore. 2018. DevOps Metrics—ADAPT Model Community. Retrieved from http://www.adapttransformation.com/devops-toolchain/monitor/devops-metrics/. Accessed on 2022-01-22.
[67] Gitlab. 2020. Getting Started with Agile/DevOps Metrics | GitLab. Retrieved from https://web.archive.org/web/20211016034045/https://about.gitlab.com/handbook/marketing/strategic-marketing/devops-metrics/. Accessed on 2022-01-22.
[68] Brian Gracely. 2017. The Most Important DevOps Metric to Measure. Retrieved from https://www.openshift.com/blog/important-devops-metric-measure. Accessed on 2022-01-22.
[69] Gary Gruver, Tommy Mouser, and Gene Kim. 2015. *Leading the Transformation: Applying Agile and DevOps Principles at Scale.* IT Revolution Press, Portland, OR.
[70] George Guimarães. 2020. On the Four Key DevOps Metrics, and Why I Measure Them Differently—SourceLevel. Retrieved from https://sourcelevel.io/blog/on-the-four-key-devops-metrics-and-why-i-measure-them-differently. Accessed on 2022-01-22.
[71] Viral Gupta, P. K. Kapur, and Deepak Kumar. 2017. Modeling and measuring attributes influencing devops implementation in an enterprise using structural equation modeling. *Info. Softw. Technol.* 92, 1 (2017), 75–91. https://doi.org/10.1016/j.infsof.2017.07.010
[72] Omed Habib. 2019. DevOps Accelerate Metrics | Harness Platform-as-a-Service. Retrieved from https://www.harness.io/blog/dora-metrics. Accessed on 2022-01-22.
[73] Philipp Haindl and Reinhold Plosch. 2020. Focus areas, themes, and objectives of non-functional requirements in DevOps: A systematic mapping study. In *Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'20)*, Martini A., Wimmer M., and Skavhaug A. (Eds.). Institute of Electrical and Electronics Engineers Inc., Johannes Kepler University Linz, Institute of Business Informatics–Software Engineering, Linz, Austria, 394–403. https://doi.org/10.1109/SEAA51224.2020.00071
[74] Tom Hall. 2016. DevOps Metrics | Atlassian. Retrieved from https://www.atlassian.com/devops/frameworks/devops-metrics. Accessed on 2022-01-22.
[75] Adam Hawkins. 2019. Measuring DevOps Success: What, Where, and How - Cloud Academy. Retrieved from https://cloudacademy.com/blog/measuring-devops-success-what-where-and-how/. Accessed on 2022-01-22.
[76] Aymeric Hemon-Hildgen, Frantz Rowe, and Laetitia Monnier-Senicourt. 2020. Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk and work conditions. *Eur. J. Info. Syst.* 29, 5 (Sept. 2020), 474–499. https://doi.org/10.1080/0960085X.2020.1782276
[77] Dan Holloran. 2019. Top Metrics for Measuring DevOps Delivery Value. Retrieved from https://web.archive.org/web/20210928103412/https://victorops.com/blog/top-metrics-for-measuring-devops-delivery-value. Accessed on 2022-01-22.

[78] Rami Honig. 2020. Bridge the DevOps Development Chasm to Boost DevOps KPIs—Ozcode. Retrieved from https://oz-code.com/blog/devops/bridging-the-devops-development-observability-chasm-to-boost-devops-kpis. Accessed on 2022-01-22.

[79] Maruf Hossain. 2020. What Key Performance Indicators (KPIs) Are Used to Measure DevOps? Retrieved from https://www.quora.com/What-key-performance-indicators-KPIs-are-used-to-measure-DevOps. Accessed on 2022-01-22.

[80] Jez Humble and David Farley. 2010. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Addison-Wesley Professional.

[81] Jez Humble and Joanne Molesky. 2011. Why enterprises must adopt devops to enable continuous delivery. *Cutter IT J.* 24, 8 (2011), 6–12.

[82] Jez Humble and Barry O'Reilly. 2014. *Lean Enterprise: How High Performance Organizations Innovate at Scale.* O'Reilly Media, Inc.

[83] Henn Idan. 2018. The Must Have Metrics Any DevOps and SRE Manager Should Measure. Retrieved from https://www.overops.com/blog/the-must-have-metrics-any-devops-and-sre-manager-should-measure/. Accessed on 2022-01-22.

[84] IEEE. 2021. 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), 91. https://doi.org/10.1109/IEEESTD.2021.9415476

[85] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. 2018. Towards a benefits dependency network for DevOps based on a systematic literature review. *J. Softw.: Evol. Process* 30, 11 (Nov. 2018), 26. https://doi.org/10.1002/smr.1957

[86] Jellyfish. 2021. Jellyfish Adds DevOps Metrics to Its Engineering Management Platform. Retrieved from https://www.prnewswire.com/news-releases/jellyfish-adds-devops-metrics-to-its-engineering-management-platform-301404849.html. Accessed on 2022-01-22.

[87] Stephen Jones, Joost Noppen, and Fiona Lettice. 2016. Management challenges for DevOps adoption within UK SMEs. In *Proceedings of the 2nd International Workshop on Quality-Aware DevOps.* ACM, Saarbrücken Germany, 7–11. https://doi.org/10.1145/2945408.2945410

[88] Vinati Kamani. 2019. 7 Crucial DevOps Metrics That You Need to Track. Retrieved from https://hub.packtpub.com/7-crucial-devops-metrics-that-you-need-to-track/. Accessed on 2022-01-22.

[89] Lea Karam. 2017. DevOps Metrics You Must Take into Account. Retrieved from https://apiumhub.com/tech-blog-barcelona/devops-metrics/. Accessed on 2022-01-22.

[90] Jane Kernel. 2020. DevOps Metrics: 7 KPIs to Evaluate Your Team's Maturity. Retrieved from https://www.xplg.com/devops-metrics-7-kpis/. Accessed on 2022-01-22.

[91] Aditya Khanduri. 2020. DevOps Metrics: Measuring What Matters. Retrieved from https://blog.sonatype.com/devops-metrics-measuring-what-matters. Accessed on 2022-01-22.

[92] Gene Kim, Kevin Behr, Kim Spafford, and George Spafford. 2014. *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win.* IT Revolution. Retrieved from https://books.google.pt/books?id=H6x-DwAAQBA.

[93] Gene Kim, Jez Humble, Patrick Debois, and John Willis. 2016. *The DevOps Handbook : How to Create World-Class Agility, Reliability, and Security in Technology Organizations.* IT Revolution Press. Retrieved from https://www.amazon.com/DevOps-Handbook-World-Class-Reliability-Organizations/dp/1942788002.

[94] Gene Kim, Jez Humble, Patrick Debois, John Willis, and Nicole Forsgren. 2021. *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations*, 2nd ed. IT Revolution.

[95] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering—A systematic literature review. *Info. Softw. Technol.* 51, 1 (2009), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

[96] Barbara A. Kitchenham. 2012. Systematic review in software engineering. In *Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies (EAST'12).* ACM Press, New York, NY, 1. https://doi.org/10.1145/2372233.2372235

[97] KnowledgeHut. 2017. What Are the Metrics and Why Do They Matter for DevOps Success? Retrieved from https://www.knowledgehut.com/blog/agile/metrics-matters-devops-success. Accessed on 2022-01-22.

[98] Dilyana Kodjamanova. 2020. 4 DevOps Metrics To Maximize Success—MentorMate. Retrieved from https://mentormate.com/blog/how-devops-metrics-pave-the-way-to-better-performance/. Accessed on 2022-01-22.

[99] Indika Kumara, Martín Garriga, Angel Urbano Romeu, Dario Di Nucci, Fabio Palomba, Damian Andrew Tamburri, and Willem-Jan van den Heuvel. 2021. The do's and don'ts of infrastructure code: A systematic gray literature review. *Info. Softw. Technol.* 137 (Sept. 2021), 106593. https://doi.org/10.1016/j.infsof.2021.106593

[100] Performance Lab. 2021. Metrics for Successful DevOps? Retrieved from https://performancelabus.com/successful-devops-metrics/. Accessed on 2022-01-22.

[101] Jane P. Laudon & Kenneth C. Laudon. 2017. *Management Information Systems: Managing the Digital Firm, Global Edition*. Pearson Education, USA.

[102] Cate Lawrence. 2020. The Four Key Metrics of DevOps. Retrieved from https://humanitec.com/blog/devops-key-metrics. Accessed on 2022-01-22.

[103] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A survey of DevOps concepts and challenges. *Comput. Surveys* 52, 6 (Nov. 2019), 35. https://doi.org/10.1145/3359981

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

[105] Alex Lichtenberger and Impactmatters. 2019. Blog: Agile: Dead End? Taking the next Step by Applying DevOps Practices Effectively—Impact Matters Blog. Retrieved from https://www.impactmatters.ch/blog/agiledevops-deadend/. Accessed on 2022-01-22.

[106] Elysia Lock. 2020. Measure DevOps Metrics That Matter. Retrieved from https://www.devopsdigest.com/measure-devops-metrics-that-matter. Accessed on 2022-01-22.

[107] Gorilla Logic. 2020. DevOps Success: What to Measure and Why—Gorilla Logic. Retrieved from https://gorillalogic.com/blog/devops-success-what-to-measure-and-why/. Accessed on 2022-01-22.

[108] JAX London. 2017. Measuring DevOps: The Key Metrics That Matter—JAX London. Retrieved from https://jaxlondon.com/blog/devops-continuous-delivery/measuring-devops-key-metrics-matter/. Accessed on 2022-01-22.

[109] Welder Pinheiro Luz, Gustavo Pinto, and Rodrigo Bonifácio. 2019. Adopting DevOps in the real world: A theory, a model, and a case study. *J. Syst. Softw.* 157, July (Nov. 2019), 110384. https://doi.org/10.1016/j.jss.2019.07.083

[110] Lucy Ellen Lwakatare, Terhi Kilamo, Teemu Karvonen, Tanja Sauvola, Ville Heikkilä, Juha Itkonen, Pasi Kuvaja, Tommi Mikkonen, Markku Oivo, and Casper Lassenius. 2019. DevOps in practice: A multiple case study of five companies. *Info. Softw. Technol.* 114 (2019), 217–230. https://doi.org/10.1016/j.infsof.2019.06.010

[111] Lucy Ellen Lwakatare, Pasi Kuvaja, Markku Oivo, C. Lassenius, T. Dingsoyr, and M. Paasivaara. 2015. Dimensions of DevOps. In *Agile Processes in Software Engineering and Extreme Programming*, Vol. 212. Springer International Publishing, Cham, 212–217. https://doi.org/10.1007/978-3-319-18612-2_19

[112] Lucy Ellen Lwakature. 2017. *Devops Adoption and Implementation in Software Development Practice: Concept, Practices, Benefits and Challenges*. University of Oulu, Finland.

[113] Gilad David Maayan. 2021. 6 Great DevOps Metrics—And How to Choose the Right Metrics. Retrieved from https://www.codemotion.com/magazine/dev-hub/devops-engineer/best-devops-metrics/. Accessed on 2022-01-22.

[114] C. Marnewick and J. Langerman. 2020. DevOps and Organizational Performance: The Fallacy of Chasing Maturity. *IEEE Softw.* 38, 5 (2020), 48–55. https://doi.org/10.1109/MS.2020.3023298

[115] Krikor Maroukian and Stephen R. Gulliver. 2021. Synthesis of a leadership model for DevOps adoption. In *Proceedings of the 2nd European Symposium on Software Engineering (ESSE'21)*. ACM, New York, NY, 58–66. https://doi.org/10.1145/3501774.3501783

[116] Lilianny Marrero and Hernán Astudillo. 2021. DevOps-RAF: An assessment framework to measure DevOps readiness in software organizations. In *Proceedings of the 40th International Conference of the Chilean Computer Science Society (SCCC'21)*. IEEE, Chile, 8. https://doi.org/10.1109/SCCC54552.2021.9650363

[117] Mark Michaelis. 2015. DevOps Metrics—IntelliTect. Retrieved from https://intellitect.com/devops-metrics/. Accessed on 2022-01-22.

[118] Alok Mishra and Ziadoon Otaiwi. 2020. Devops and software quality: A systematic mapping. *Comput. Sci. Rev.* 38, 1 (Nov. 2020), 14. https://doi.org/10.1016/j.cosrev.2020.100308

[119] Sara Miteva. 2020. 13 DevOps Metrics for Increased Productivity. Retrieved from https://dev.to/microtica/13-devops-metrics-for-increased-productivity-5084. Accessed on 2022-01-22.

[120] Johann Mitlohner, Sebastian Neumaier, Jurgen Umbrich, and Axel Polleres. 2016. Characteristics of open data CSV files. In *Proceedings of the 2nd International Conference on Open and Big Data (OBD'16)*. IEEE, 72–79. https://doi.org/10.1109/OBD.2016.18

[121] Samer I. Mohamed. 2016. DevOps maturity calculator DOMC -Value Oriented Approach. *Int. J. Eng. Res. Sci.* 2, 2 (2016), 2395–6992.

[122] Luciano de Aguiar Monteiro. 2021. A proposal to systematize introducing devops into the software development process. In *Proceedings of the International Conference on Software Engineering*. IEEE, 269–271. https://doi.org/10.1109/ICSE-Companion52605.2021.00124

[123] Fabio Jose Moraes. 2018. DevOps KPI in Practice — Chapter 1 — Deployment Speed, Frequency and Failure. Retrieved from https://medium.com/@fabiojose/devops-kpi-in-practice-chapter-1-deployment-speed-frequency-and-failure-2fd0a9303249. Accessed on 2022-01-22.

[124] Gabriela Motroc. 2018. Key DevOps Metrics That Matter: How Well Does Your Team Sleep? Retrieved fromhttps://web.archive.org/web/20220119200950/https://jaxenter.com/devops-influencers-interview-series-4-142312.html. Accessed on 2022-01-22.

[125] Mirna Muñoz and Mario Negrete Rodríguez. 2021. A guidance to implement or reinforce a DevOps approach in organizations: A case study. *J. Softw.: Evol. Process* 1 (2021), 21. https://doi.org/10.1002/smr.2342

[126] Håvard Myrbakken and Ricardo Colomo-Palacios. 2017. DevSecOps: A multivocal literature review. *Commun. Comput. Info. Sci.* 770, 1 (2017), 17–29. https://doi.org/10.1007/978-3-319-67383-7_2

[127] Omar Nasser. 2020. What Metrics Should DevOps Teams Be Tracking? Retrieved from https://cto.ai/blog/what-metrics-should-devops-teams-be-tracking/. Accessed on 2022-01-22.

[128] Terence Nero. 2021. DevOps Metrics : 15 KPIs That Boost Results and RoI—Cuelogic Technologies Pvt. Ltd. Retrieved from https://www.cuelogic.com/blog/devops-metrics. Accessed on 2022-01-22.

[129] Rodney T. Ogawa and Betty Malen. 1991. Towards rigor in reviews of multivocal literatures: Applying the exploratory case study method. *Rev. Edu. Res.* 61, 3 (Sept. 1991), 265–286. https://doi.org/10.3102/00346543061003265

[130] Opsgenie. 2021. DevOps Metrics. Retrieved from https://docs.opsgenie.com/docs/devops-metrics-global. Accessed on 2022-01-22.

[131] Roy Osherove. 2018. Ten Devops an Agility Metrics to Check at the Team Level—Pipeline Driven. Retrieved from https://pipelinedriven.org/article/ten-ideas-for-things-you-can-measure-as-a-team-on-your-devops-journey. Accessed on 2022-01-22.

[132] Hewlett Packard. 2016. Measuring DevOps Success. Retrieved from http://www.baldrover.com/wp-content/uploads/Measuring-DevOps-Success.pdf. Accessed on 2022-01-22.

[133] Pagerduty. 2015. The Best Metrics for Driving Cultural Change in DevOps Teams. Retrieved from https://www.pagerduty.com/blog/best-metrics-devops-culture/. Accessed on 2022-01-22.

[134] Tim Palko. 2015. The Missing Metrics of DevOps. Retrieved from https://insights.sei.cmu.edu/devops/2015/05/the-missing-metrics-of-devops.html. Accessed on 2022-01-22.

[135] Pulasthi Perera, Roshali Silva, and Indika Perera. 2017. Improve software quality through practicing DevOps. In *Proceedings of the 17th International Conference on Advances in ICT for Emerging Regions (ICTer'17)*. IEEE, 13–18. https://doi.org/10.1109/ICTER.2017.8257807

[136] Kai Petersen and Claes Wohlin. 2009. Context in industrial software engineering research. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 401–404. https://doi.org/10.1109/ESEM.2009.5316010

[137] Plutora. 2020. DORA DevOps Metrics—Accelerate Your Value Stream—Plutora.Com. Retrieved from https://www.plutora.com/resources/videos/devops-dora-metrics. Accessed on 2022-01-22.

[138] Plutora. 2021. The 10 Essential DevOps Metrics That Really Matter. Retrieved from https://www.plutora.com/blog/10-essential-devops-metrics-that-really-matter. Accessed on 2022-01-22.

[139] Dina Graves Portman. 2020. Using the Four Keys to Measure Your DevOps Performance. Retrieved from https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance. Accessed on 2022-01-22.

[140] Ron Powell. 2020. How to Measure DevOps Success: 4 Key Metrics. Retrieved from https://circleci.com/blog/how-to-measure-devops-success-4-key-metrics/. Accessed on 2022-01-22.

[141] Luís Prates, João Faustino, Miguel Silva, and Rúben Pereira. 2019. DevSecOps metrics. In *Lecture Notes in Business Information Processing*, Maslankowski J. Wrycza S. (Ed.). Vol. 359. ISCTE-IUL, Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal, 77–90. https://doi.org/10.1007/978-3-030-29608-7_7

[142] Alix Pressley. 2021. The Top 10 DevOps Metrics You Should Know About. Retrieved from https://www.intelligentcio.com/eu/2021/04/16/the-top-10-devops-metrics-you-should-know-about/. Accessed on 2022-01-22.

[143] Rebecca Pruess. 2020. DevOps Best Practices: 5 Key Performance Indicators. Retrieved from https://flexagon.com/devops-best-practices-5-key-performance-indicators/. Accessed on 2022-01-22.

[144] Puppet Labs. 2013. *2013 State of DevOps Report.* Technical Report. Retrieved from http://puppetlabs.com/2013-devops-report.

[145] Puppet Labs. 2014. *2014 State of DevOps Report.* Technical Report. Retrieved from http://puppetlabs.com/2014-devops-report.

[146] Puppet Labs. 2015. *2015 State of DevOps Report.* Technical Report. Retrieved from http://puppetlabs.com/2015-devops-report.

[147] Puppet Labs. 2016. *2016 State of DevOps Report.* Technical Report. Retrieved from https://puppetlabs.com/solutions/devops/.

[148] Puppet Labs. 2017. *2017 State of DevOps Report.* Technical Report. Retrieved from https://puppetlabs.com/solutions/devops/.

[149] Puppet Labs. 2018. *2018 State of DevOps Report.* Technical Report. Retrieved from https://media.webteam.puppet.com/uploads/2019/11/Puppet-State-of-DevOps-Report-2018_update.pdf.

[150] Puppet Labs. 2019. *2019 State of DevOps Report.* Technical Report. Retrieved from https://puppet.com/resources/report/2019-state-of-devops-report.

[151] Puppet Labs. 2020. *2020 State of DevOps Report*. Technical Report. Retrieved from https://puppet.com/resources/report/2020-state-of-devops-report/.

[152] Asif Qumer Gill, Abhishek Loumish, Isha Riyat, and Sungyoup Han. 2018. DevOps for information management systems. *VINE J. Info. Knowl. Manage. Syst.* 48, 1 (Jan. 2018), 122–139. https://doi.org/10.1108/VJIKMS-02-2017-0007

[153] S. Rafi, W. Yu, M. A. Akbar, A. Alsanad, and A. Gumaei. 2020. Prioritization-based taxonomy of DevOps security challenges using PROMETHEE. *IEEE Access* 8 (2020), 105426–105446. https://doi.org/10.1109/ACCESS.2020.2998819

[154] Anjana Ramesh. 2020. Ten Key DevOps Metrics to Accelerate Your Continuous Delivery Pipeline. Retrieved from https://web.archive.org/web/20211205080155/https://www.go2group.com/resources/blog/devops-metrics-to-accelerate-ci-cd/. Accessed on 2022-01-22.

[155] Aruna Ravichandran, Kieran Taylor, and Peter Waterhouse. 2016. *DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory*. Springer Nature. https://doi.org/10.1007/978-1-4842-1842-6

[156] ReleaseTEAM. 2021. DevOps Metrics Measure Your DevOps Results. Retrieved from https://www.releaseteam.com/measure-your-devops-results/. Accessed on 2022-01-22.

[157] New Relic. 2018. Measuring DevOps. Retrieved from https://newrelic.com/devops/measuring-devops. Accessed on 2022-01-22.

[158] Jennifer Riggins. 2020. Google's Formula for Elite DevOps Performance—The New Stack. Retrieved from https://thenewstack.io/googles-formula-for-elite-devops-performance/. Accessed on 2022-01-22.

[159] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen, and Tomi Männistö. 2016. DevOps adoption benefits and challenges in practice: A case study. In *Product-Focused Software Process Improvement*, Vol. 10027 LNCS. Springer International Publishing, Department of Computer Science, University of Helsinki, Finland, 590–597. https://doi.org/10.1007/978-3-319-49094-6_44

[160] Stephen Roddewig. 2021. 8 DevOps Metrics to Measure Team Activity & Progress. Retrieved from https://blog.hubspot.com/website/devops-metrics. Accessed on 2022-01-22.

[161] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M. Verner, and Markku Oivo. 2017. Continuous deployment of software intensive products and services: A systematic mapping study. *J. Syst. Softw.* 123 (2017), 263–291. https://doi.org/10.1016/j.jss.2015.12.015

[162] Pilar Rodríguez, Mika Mäntylä, Markku Oivo, Lucy Ellen Lwakatare, Pertti Seppänen, and Pasi Kuvaja. 2019. Advances in using agile and lean processes for software development. In *Advances in Computers*, Memon A. M. (Ed.). Vol. 113. Academic Press Inc., Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland, 135–224. https://doi.org/10.1016/bs.adcom.2018.03.014

[163] Wiebe de Roos. 2021. Dealing with DevOps Metrics and KPIs. Retrieved from https://web.archive.org/web/20210925173812/https://amazicworld.com/dealing-with-devops-metrics-and-kpis/. Accessed on 2022-01-22.

[164] Mike Rother. 2019. *Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results*. MGH, New York.

[165] Martin Rütz. 2019. DEVOPS: A systematic literature review. *Info. Softw. Technol.* 86 (Aug. 2019), 87–100. https://www.researchgate.net/publication/335243102

[166] Isaac Sacolick. 2018. 15 KPIs to Track Devops Transformation. Retrieved from https://www.infoworld.com/article/3297041/15-kpis-to-track-devops-transformation.html. Accessed on 2022-01-22.

[167] Johnny Saldana. 2015. *The Coding Manual for Qualitative Researchers Third Edition* (3rd ed.). SAGE Publications Ltd, Los Angeles, CA.

[168] Marc Sallin, Martin Kropp, Craig Anslow, James W. Quilty, and Andreas Meier. 2021. Measuring software delivery performance using the Four Key Metrics of DevOps. In *Lecture Notes in Business Information Processing*, Peggy Gregory, Casper Lassenius, Xiaofeng Wang, and Philippe Kruchten (Eds.), Vol. 419 LNBIP. Springer International Publishing, Cham, 103–119. https://doi.org/10.1007/978-3-030-78098-2_7

[169] Meshach Samuel. 2019. How to Successfully Scale Agile and DevOps – Part 3: Driving Success with Technology. Retrieved from https://web.archive.org/web/20211204052511/https://www.hcltech.com/blogs/how-successfully-scale-agile-and-devops-part-3-driving-success-technology. Accessed on 2022-01-22.

[170] Mary Sánchez-Gordón, Ricardo Colomo-Palacios, Alex Sánchez, and Sandra Sanchez-Gordon. 2020. Integrating approaches in software development: A case analysis in a small software company. In *Systems, Software and Services Process Improvement (Communications in Computer and Information Science)*, Murat Yilmaz, Jörg Niemann, Paul Clarke, and Richard Messnarz (Eds.). Springer International Publishing, Cham, 95–106. https://doi.org/10.1007/978-3-030-56441-4_7

[171] Amy Schurr. 2019. Mobile App DevOps Metrics That Matter—NowSecure. Retrieved from https://www.nowsecure.com/blog/2019/02/27/mobile-app-devops-metrics-that-matter/. Accessed on 2022-01-22.

[172] Mali Senapathi, Jim Buchan, and Hady Osman. 2018. DevOps capabilities, practices, and challenges: Insights from a case study. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering (EASE'18)*. ACM, New York, NY, 57–67. https://doi.org/10.1145/3210459.3210465

[173] Charlie Shabe. 2017. Understanding DevOps Metrics. Retrieved from https://betanews.com/2017/08/24/devops-metrics/. Accessed on 2022-01-22.

[174] Sanjeev Sharma. 2017. *The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise*. John Wiley & Sons, Inc., Indianapolis, Indiana. https://doi.org/10.1002/9781119310778

[175] Reshma Shinde. 2019. Is Your DevOps Successful? Retrieved from https://web.archive.org/web/20210729003128/https://www.accenture.com/us-en/blogs/software-engineering-blog/reshma-shinde-devops-success-metrics. Accessed on 2022-01-22.

[176] Gursimran Singh. 2019. Measuring DevOps Success with DevOps Metrics. Retrieved from https://www.xenonstack.com/blog/devops-metrics/. Accessed on 2022-01-22.

[177] Jens Smeds, Kristian Nybom, and Ivan Porres. 2015. DevOps: A definition and perceived adoption impediments. In *Lecture Notes in Business Information Processing*. Vol. 212. Springer, 166–177. https://doi.org/10.1007/978-3-319-18612-2_14

[178] Sam Smith. 2020. High Performing DevOps Metrics. Retrieved from https://samlearnsazure.blog/2020/04/30/high-performing-devops-metrics/. Accessed on 2022-01-22.

[179] Barry Snyder and Bill Curtis. 2017. Using analytics to guide improvement during an agile-DevOps transformation. *IEEE Softw.* 35, 1 (Jan. 2017), 78–83. https://doi.org/10.1109/MS.2017.4541032

[180] Indium Software. 2017. 6 Metrics to Measure DevOps Test Automation. Retrieved from https://www.indiumsoftware.com/blog/devops-test-automation-metrics/. Accessed on 2022-01-22.

[181] Damir Solajić and Anamarija Petrović. 2019. Devops and modern software delivery. In *Proceedings of the International Scientific Conference (Sinteza'19)*. Singidunum University, Novi Sad, Serbia, 360–368. https://doi.org/10.15308/Sinteza-2019-360-368

[182] Leandro Sousa, António Trigo, and João Varajão. 2019. Devops—Foundations and perspectives. In *Proceedings of the 19th Portuguese Association of Information Systems Conference (CAPSI'19)*. Associacao Portuguesa de Sistemas de Informacao, Instituto Politécnico de Coimbra, ISCAC, Quinta Agrícola, Bencanta, Coimbra, 3040-316, Portugal, 8. Retrieved from https://aisel.aisnet.org/capsi2019/8/.

[183] Coveros Staff. 2016. Essential Quantitative DevOps Metrics—Coveros. Retrieved from https://www.coveros.com/essential-quantitative-devops-metrics/. Accessed on 2022-01-22.

[184] Jonny Steiner. 2021. These Are the DevOps Metrics That Will Boost Your VSM. Retrieved from https://digital.ai/catalyst-blog/these-are-the-devops-metrics-that-will-boost-your-vsm. Accessed on 2022-01-22.

[185] Sean Sullivan. 2021. Four Key DevOps Metrics for Success. Retrieved from https://www.dragonspears.com/blog/four-key-devops-metrics-for-success. Accessed on 2022-01-22.

[186] Paul Swartout. 2014. *Continuous Delivery and DevOps: A Quickstart Guide*, 2nd ed. Packt Publishing Ltd, UK.

[187] Dave Swersky. 2017. What Key Performance Indicators (KPIs) Are Used to Measure DevOps? Retrieved from https://devops.stackexchange.com/questions/738/what-key-performance-indicators-kpis-are-used-to-measure-devops. Accessed on 2022-01-22.

[188] Lalith Boovaragavan Marketing Manager at Aspire Systems. 2021. 7 Ways to Measure DevOps Success - Aspire Systems. Retrieved from https://blog.aspiresys.com/infrastructure-managed-services/7-ways-to-measure-devops-success/. Accessed on 2022-01-22.

[189] Information Technology and Intelligence Consulting. 2019. *2019 Global Server Hardware, Server OS Reliability Report*. Technical Report March. Information Technology Intelligence Consulting (ITIC) Corp.

[190] Riverbed Technology. 2017. Seven Metrics That Matter When Measuring DevOps Success. Retrieved from https://web.archive.org/web/20210623152132/https://www.aternity.com/blogs/seven-metrics-matter-measuring-devops-success/. Accessed on 2022-01-22.

[191] Daniel Teixeira, Rúben Pereira, Telmo Henriques, Miguel Mira Da Silva, and João Faustino. 2020. A maturity model for DevOps. *Int. J. Agile Syst. Manage.* 13, 4 (2020), 464. https://doi.org/10.1504/IJASM.2020.112343

[192] Daniel Teixeira, Ruben Pereira, and Miguel Mira. 2019. *A Maturity Model to Support DevOps Implementation A Maturity Model to Support DevOps Implementation*. Technical Report. Instituto Universitario de Lisboa (ISCTE-IUL). Retrieved from http://hdl.handle.net/10071/20297.

[193] Bjørnar Tessem and Jon Iden. 2008. Cooperation between developers and operations in software engineering projects. *Proceedings of the International Conference on Software Engineering*. 105–108. https://doi.org/10.1145/1370114.1370141

[194] TestEnvironmentManagement.com. 2019. Top 5 DevOps Metrics – Test Environment Management. Retrieved from https://www.testenvironmentmanagement.com/top-5-devops-metrics/. Accessed on 2022-01-22.

[195] Nora Tomas, Jingyue Li, and Huang Huang. 2019. An empirical study on culture, automation, measurement, and sharing of DevSecOps. In *Proceedings of the 5th International Conference on Cyber Security and Protection of Digital Services (Cyber Security'19)*. Institute of Electrical and Electronics Engineers Inc., Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, 8. https://doi.org/10.1109/CyberSecPODS.2019.8884935

[196] Ubiq. 2020. Top DevOps Metrics and KPIs To Monitor Regularly—Ubiq BI Blog. Retrieved from http://ubiq.co/analytics-blog/top-devops-metrics-kpis-to-monitor-regularly/. Accessed on 2022-01-22.
[197] Sricharan Vadapalli. 2018. *DevOps: Continuous Delivery, Integration, and Deployment with DevOps Dive into the Core DevOps Strategies*. Packt Publishing Ltd, UK.
[198] Alfonso Valdes. 2020. 5 DevOps Metrics and KPIs That CTOs Must Monitor. Retrieved from https://www.clickittech.com/devops/devops-metrics-and-kpis/. Accessed on 2022-01-22.
[199] Valtech. 2015. 4 Metrics for Measuring DevOps Success. Retrieved from https://www.valtech.com/insights/4-metrics-for-measuring-devops-success/. Accessed on 2022-01-22.
[200] Dmytro Vavilkin. 2021. DevOps Metrics and KPIs to Improve Your Team Efficiency. Retrieved from https://u-tor.com/topic/devops-metrics-and-kpis. Accessed on 2022-01-22.
[201] Veritis. 2020. Measuring DevOps: Key 'Metrics' and 'KPIs' That Drive Success! Retrieved from https://www.veritis.com/blog/measuring-devops-key-metrics-and-kpis-that-drive-success/. Accessed on 2022-01-22.
[202] Harshal Vora. 2018. Software Quality Metrics for Agile and DevOps Success—QMetry. Retrieved from https://www.qmetry.com/blog/software-quality-metrics-for-agile-and-devops-success/. Accessed on 2022-01-22.
[203] Kentaro Wakayama. 2020. How to Ensure the Success of DevOps in Your Organization. Retrieved from https://codersociety.com/blog/articles/devops-success-in-organization. Accessed on 2022-01-22.
[204] Peter Waterhouse. 2015. DevOps Practitioner Series—Metrics That Matter. Retrieved from https://docs.broadcom.com/doc/devops-practitioner-series-metrics-that-matter-developing-and-tracking-key-indicators. Accessed on 2022-01-22.
[205] Matt Watson. 2017. 15 Metrics for DevOps Success. Retrieved from https://stackify.com/15-metrics-for-devops-success/. Accessed on 2022-01-22.
[206] Stephen Watts. 2017. DevOps: Metrics and Key Performance Indicators (KPIs). Retrieved from https://itchronicles.com/devops/devops-metrics-kpis/. Accessed on 2022-01-22.
[207] Stephen Watts. 2019. DevOps Metrics and KPIs – BMC Blogs. Retrieved from https://www.bmc.com/blogs/devops-kpi-metrics/. Accessed on 2022-01-22.
[208] Waydev. 2021. DORA Metrics: The 4 Key Metrics For Efficient DevOps Performance Tracking. Retrieved from https://waydev.co/dora-metrics/. Accessed on 2022-01-22.
[209] Jonathan Weinberg. 2021. Four Key DevOps Metrics and How To Measure Them. Retrieved from https://www.wwt.com/article/four-key-devops-metrics-and-how-to-measure-them. Accessed on 2022-01-22.
[210] Anton Weiss. 2016. Measuring DevOps Flow by Otomato. Retrieved from https://devopsflowmetrics.org/. Accessed on 2022-01-22.
[211] R. Westrum. 2004. A typology of organisational cultures. *Qual. Safe. Health Care* 13, 2 (2004), 22–27. https://doi.org/10.1136/qshc.2003.009522
[212] Johannes Wettinger, Uwe Breitenbücher, and Frank Leymann. 2014. DevOpSlang—Bridging the gap between development and operations. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8745 LNCS. IFIP, Stuttgart, 108–122. https://doi.org/10.1007/978-3-662-44879-3_8
[213] Anna Wiedemann, Nicole Forsgren, Manuel Wiesche, Heiko Gewald, and Helmut Krcmar. 2019. Research for practice: The devops phenomenon. *Commun. ACM* 62, 8 (2019), 44–49. https://doi.org/10.1145/3331138
[214] Anna Wiedemann, Manuel Wiesche, Heiko Gewald, and Helmut Krcmar. 2020. Understanding how DevOps aligns development and operations: A tripartite model of intra-IT alignment. *Eur. J. Info. Syst.* 29, 5 (Oct. 2020), 458–473. https://doi.org/10.1080/0960085X.2020.1782277
[215] John Willis and Itrevolution. 2012. DevOps Culture (Part 1) - IT Revolution. Retrieved from https://itrevolution.com/devops-culture-part-1/. Accessed on 2022-01-22.
[216] Claes Wohlin. 2014. 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 (EASE'14)*. ACM, New York, NY, 10. https://doi.org/10.1145/2601248.2601268
[217] Liming Zhu, Len Bass, and George Champlin-Scharff. 2016. DevOps and its practices. *IEEE Softw.* 33, 3 (May 2016), 32–34. https://doi.org/10.1109/MS.2016.81

Received 11 April 2022; revised 31 January 2024; accepted 7 March 2024

CHAPTER 4

Article #3

The article (A3) aims to promote DevOps adoption by identifying and validating 37 DevOps capabilities and 24 metrics through Design Science Research (DSR), resulting in a Capability Evaluation Matrix, presented in Table 4.17, with updated relations in blue [28]. As part of the study objectives, it evaluates and tries to enhance organizational practices by creating an improvement roadmap in the form of this artifact. This is done by finding relationships between DevOps metrics and capabilities and identifying which DevOps capabilities have a positive impact on which main metrics.

The primary contributions and findings comprise the validation of the 37 DevOps capabilities and the inclusion of two additional primary metrics, derived from the interviews, to the 22 metrics identified through previous literature reviews. The findings were validated using semi-structured interviews with practitioners. This led to the development of a **Capability Evaluation Matrix** artifact, which was proposed and validated by DevOps experts, confirming the usefulness to promote DevOps implementation and adoption. It also discusses the significance of empowering teams and establishing a robust organizational culture for the successful implementation of DevOps.

Article details:

– Title: Capabilities and metrics in DevOps: A design science study

– Date: May 2023

– Journal: Information & Management

– Scimago Journal Rank: Q1

– Publisher: Elsevier B.V.

Information & Management 60 (2023) 103809

Contents lists available at ScienceDirect

Information & Management

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

Capabilities and metrics in DevOps: A design science study

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

[a] Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal
[b] Instituto de Telecomunicações(IT), Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal
[c] Instituto Superior Técnico (IST), Lisbon, Portugal

A R T I C L E I N F O

Keywords:
DevOps
Capabilities
Practices
Metrics
Software delivery
IT operations

A B S T R A C T

Customer demands, competition, regulatory environments, and sophisticated external threats have all increased the importance of DevOps in IT organizations. However, DevOps adoption is still uneven, emphasizing the need to provide management with relevant IS data and insights. Regrettably, there is a measurement inefficiency between these capabilities.

To sustain promoting DevOps adoption, Design Science Research (DSR) is done using two multivocal literature reviews and semi-structured interviews to elicit key DevOps metrics and capabilities. Thirty-seven DevOps capabilities and twenty-four metrics were defined, classified, and validated by conducting 31 interviews with practitioners and experts leading to an outcome-based capability evaluation matrix, for promoting DevOps implementation and adoption. Empowering teams and organizational culture have the greatest impact.

1. Introduction

In any Information Technology (IT) enabled organization [111] where Software Engineering (SE) is part of the core business, having a competitive advantage [58] by doing things better than competitors is a requirement for success [25]. However, the need for frequent software delivery, without sustained builds, proper testing and release automation, generates burnout and pain in the engineers doing operations [123], decaying software delivery performance and leading to poor reliability [30].

In response to this faulty process, we see the emergence of Developer (Dev) and Operations(Ops) (DevOps), an organizational approach that emphasizes empathy and encourages greater collaboration between engineering teams involved in software delivery [47], in order to reduce development time, improve deployment rates, increase stability, optimize Mean Time to Recover, and lower deployment and implementation costs [89].

Management needs to have a clear vision of the steps to take ahead based on information and metrics in order to increase efficiency [58], thus the success of applying DevOps capabilities should be measured with existing DevOps metrics. But what are these metrics and capabilities? And what relation exists between them?

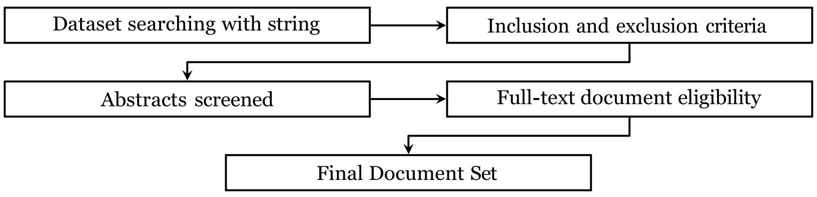
While DevOps adoption [102] success is irregular, as a few impediments exist [27,105] and measuring its maturity can be hard, relating different DevOps capabilities with existing DevOps metrics will support the management decision process towards increasing performance in the Software Development Life Cycle (SDLC) within the organization. The main objective is to evaluate and improve the organization's practices by creating an improvement roadmap [79] and finding the relation between DevOps metrics and DevOps capabilities or practices that will facilitate adopting DevOps successfully.

However, since there exists a lack of systematization of the different DevOps capabilities or practices and existing DevOps metrics, this research intends to explore a relationship between the metrics and the capabilities, in order to elicit the main DevOps metrics for each DevOps capability, align the relations impacting positively each found metric and create an evaluation matrix of the DevOps capabilities. Therefore, the following research questions are proposed: RQ1. What are the main DevOps capabilities or practices? RQ2. Where are capabilities or practices mentioned? RQ3. How do authors distinguish capabilities from practices? RQ4. What are the main DevOps metrics? RQ5. What is the purpose of each metric? RQ6. Why is each metric important? RQ7. How are DevOps capabilities categorized? RQ8. How are the main metrics categorized? RQ9. What DevOps capabilities have a positive impact on which main metrics?

The rationale for this research proposal is based on the lack of previous work that has examined the relationship between capabilities and metrics in DevOps in a single study. While there are studies that address

* 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. da Silva).

https://doi.org/10.1016/j.im.2023.103809
Received 17 February 2022; Received in revised form 15 April 2023; Accepted 25 April 2023
Available online 9 May 2023
0378-7206/© 2023 Elsevier B.V. All rights reserved.



Dataset searching with string
Inclusion and exclusion criteria
Abstracts screened
Full-text document eligibility
Final Document Set

Fig. 1. Review protocol planned and used for the literature review.

Table 1
Databases and steps used in the initial systematic literature review (SLR).

Database	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Scopus	443	31	31	25	14	4
Web Of Science	18	12	12	8	6	4
IEEE	11	7	7	7	7	4
ACM	667	7	7	7	7	5
EBSCO	28	17	2	2	2	0
Total	1167	74	59	49	36	17

*Legend:*Step 1 = Query All fields, All documents; Step 2 = Query Title and Abstract, Peer reviewed publications; Step 3 = Relevant (inclusion/exclusion criteria); Step 4 = After Removing duplicates; Step 5 = After Abstracts Screened; Step 6 = Full-text Document Assess.

both topics separately, as evidenced in Section 2, our unique study takes a more comprehensive approach by integrating insights from both practitioners and academic sources into a single research framework, utilizing the applied methodology of Design Science Research as discussed in Section 3 in order to systematize the relationship between DevOps metrics and DevOps capabilities.

2. Related work

This section offers a summary and critical analysis of previous studies that are relevant to the current research, while establishing the context for the study and highlighting the gaps or limitations in the existing literature that the research aims to address.

The importance and uniqueness of the study are acknowledged through a rigorous research-based approach using the Systematic Literature Review (SLR) protocol. Systematic Literature Review (SLR) is a commonly used method in Software Engineering and other scientific areas for gathering evidence from relevant studies. The SLR process typically consists of three phases: planning the review, conducting the review, and documenting/reporting the review.

The original SLR process was proposed by Kitchenham et al. in their seminal works [54,55] and has been widely adopted by the Software Engineering community [77]. An SLR is designed to systematically expose and analyze relevant literature to answer a set of research questions, providing sustained evidence for the study at hand. In this Systematic Literature Review (SLR) process, the planning phase is undertaken. This begins by providing the motivation for the study, followed by stating the objectives and research question that will guide the research. Additionally, a review protocol is presented to guide the literature review process.

The main motivation for the research is to improve DevOps adoption by providing management, in software development organizations, with relevant information and metrics to measure the success and efficiency of implementing DevOps capabilities internally [58,105]. In order to achieve this objective, the aim was to find a relationship between DevOps metrics and capabilities and provide a systematization from existing literature in this area. To accomplish the initial research purpose, a comprehensive investigation of scientific literature related to DevOps metrics and capabilities was conducted. This led to the formulation of the following research question: ***Which studies relate DevOps metrics to DevOps capabilities?***.

To identify relevant studies related to the proposed research question, a comprehensive search was conducted in January 2021 using carefully chosen keywords. These keywords were used to compose the search string below, which enabled retrieving the maximum number of studies from the selected scientific databases. The search string and datasets are listed in this section for reference.

- **Search String**: `(devops AND (metrics OR measures OR kpi OR indicators) AND (practices OR capabilities))`.
- **Datasets**: The search engines used were two brokers: Scopus[1] and Web of Science,[2] in conjunction with IEEE,[3] ACM.[4] and EBSCO[5]

The complete review protocol is illustrated in Fig. 1. The first set of papers was obtained. In a first phase, after the search was complete, inclusion and exclusion criteria were applied for refining the search results.

The *inclusion criteria* for the Systematic Literature Review (SLR) were that the papers must be peer-reviewed and scientific, explicitly discuss DevOps, and mention DevOps metrics and capabilities. The *exclusion criteria* removed papers not written in English, non peer-reviewed or inaccessible papers, vendor tool advertisements or duplicates, and papers that do not mention DevOps metrics or capabilities. The abstracts were carefully screened to assess their relevance to the research question. Subsequently, the relevant papers were thoroughly read to obtain the final selection of studies that meet the full-text eligibility criteria for the final document set, which will be used for the review.

The complete sequence of steps is summarized in Table 1, giving us a summary of the articles found with the filters used. In *step 1*, the search string was applied to all fields in all documents for each dataset, resulting in 1167 documents. In *step 2*, the same search string was used to filter in peer-reviewed publications with keywords either in titles or abstracts, resulting in a total of 74 papers for screening. The discrepancy from step 1 to step 2 is justified by the fact that initially the keywords could be found anywhere within the returned item and some search engines return more literature than just academic papers, like newspapers or reports.

After applying the inclusion and exclusion criteria in *step 3*, a total of 59 articles remained. This led to *step 4*, which involved removing duplicates from the list of results to obtain a set of documents for abstract screening. Following the screening, a full-text document assessment was conducted, resulting in the identification of 17 eligible documents for further analysis and extraction of relevant information for this research. The distribution of the final document set by database reveals that the majority, 29.4% of results originated from ACM. Scopus, Web of Science, and IEEE each contributed with 23.5% relevant research documents.

To evaluate the suitability of the documents and ascertain their relevance, four lists were created, as summarized in Fig. 2. However, none of the publications found in the search systematically addressed the relationship between DevOps metrics and DevOps capabilities to

[1] https://www.scopus.com
[2] https://apps.webofknowledge.com
[3] https://ieeexplore.ieee.org
[4] https://dl.acm.org
[5] https://search.ebscohost.com

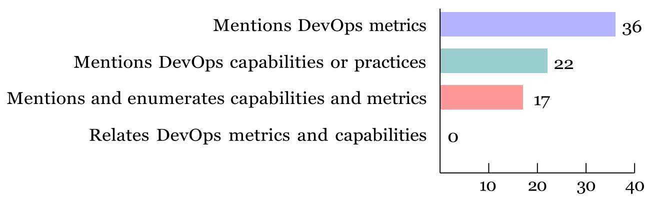


Fig. 2. Document relevance breakdown.

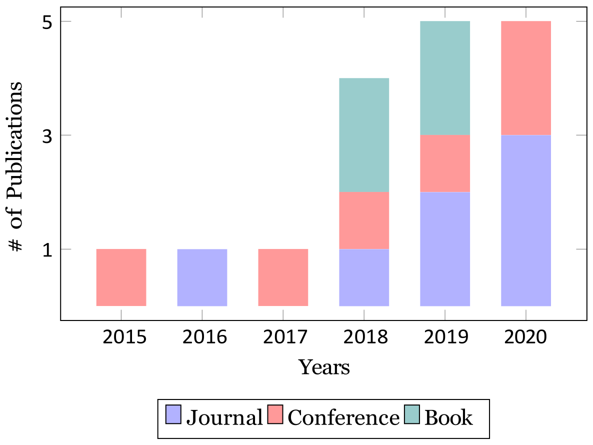


Fig. 3. Publication year of final documents per type.

adequately answer the research question. Out of the 36 publications initially screened in *step 5*, which mentioned DevOps metrics, only 22 mentioned capabilities or practices. However, only 17 publications mentioned and enumerated both capabilities and metrics. Even though the selection did not yield any conclusive results for the research question, it still highlights the significant interest in these topics within the published literature. In summary, out of the 36 publications screened in *step 5*, only 17 were considered the most relevant, as shown in Fig. 2.

The increase in selected papers over the years, as shown in Fig. 3, is also noteworthy. This indicates a growing interest in the previous years with an incrementing volume of research on DevOps metrics and capabilities topic, affirming the relevance and potential impact of the proposed research in the field. The first publication in this set [80] was a conference paper, followed by a journal article and another paper in the subsequent two years. However, the interest in this subject significantly grew in 2018, including books, and in 2020 with 2 conference papers and 3 journal articles, totaling 5 relevant publications in each of the last 2 years.

The given publications, were fully reviewed with a manual inspection for similar research objectives in order to deeply understand if any of the remaining articles were relating DevOps metrics with DevOps capabilities.

Nagarajan et al. [73] focuses on creating artifacts to help financial organizations implement DevOps, guided by a conceptual framework. Forsegren et al. [33] aims to develop a taxonomy of Software Delivery Performance Profiles for DevOps development settings, incorporating attributes for throughput and stability. Abdelkebir et al. [1] presents a holistic and practical strategic framework for improving ITSM service management processes, incorporating Agility management based on DevOps and an agility Process Maturity Framework. Bertolino et al. [7] proposes DevOpRET, an approach for reliability testing as part of the acceptance testing stage in DevOps, supporting continuous software reliability testing. Mishra et al. [69] conducts a systematic mapping of DevOps features and quality attributes, analyzing the implications of DevOps features on software quality. Marijan et al. [63] presents an approach for improving time-efficiency in DevOps, particularly in continuous integration testing, using continuous test optimization. Angara et al. [3] conducts a literature survey on various strategies applied for continuous testing, proposes a continuous testing architecture, and presents the conceptual design of important testing metrics for successful implementation of continuous testing in the context of DevOps. Prates et al. [84] identifies and lists several monitoring metrics for DevSecOps through a Multivocal Literature Review (MLR). Pourmajidi et al. [82] reports the experience of creating a prototype platform for monitoring and analysis of logs emitted by components of IBM Cloud services.

AhmadIbrahim et al. [49] reviews the challenges of quality assurance in DevOps and provides tentative recommendations to address quality issues. Sun et al. [109] proposes an approach that uses log information to select different classifiers trained with monitored metrics data via public clouds during DevOps operations. Tamburri et al. [110] elaborates a conjecture, illustrating a set of metrics to be used in the DevOps scenario and overview challenges and future research directions. Diaz et al. [20] presents the formalization of the activity for supporting feedback from Operations to Development of IoT systems. Wang et al. [119] focuses on the improvement of the test automation process, presenting an experience report on TAPI in a DevOps team at *F*-Secure, a Finnish software company. Perez et al. [80] considers the problem of designing a tool capable of providing feedback to the developer on performance, reliability, and general quality characteristics of the application at runtime. Ding et al. [21] evaluates the performance of readily available tests in the release pipeline and examines whether these tests can be used as performance tests to demonstrate performance improvements from performance issue fixes, and Snyder et al. [106] emphasizes the importance of using analytics during the DevOps transformation improvement.

Notably, none of the publications share the same objective as this study. Therefore, the research proposal addresses this existing gap in the literature by aiming a study that takes a more comprehensive approach to explore the relationship between capabilities and metrics in DevOps. By integrating insights from both practitioners and academic sources within a single research framework and applying Design Science Research methodology, this study contributes to the advancement of understanding and practice in the field of DevOps research and adoption. Through this approach, our study advances the understanding and practice of capabilities and metrics in DevOps, and makes a significant impact on the field.

3. Research methodology

In an initial report detailed in Section 2, a literature review was conducted to identify the problem of this research. In this article, the activities of Design Science Research (DSR) proposed by [45] in Information Systems (IS) research are used as the main investigation methodology. For the build phase [62], two Multivocal Literature Reviews (MLRs) [38] are conducted as mentioned in Section 3.2 to help identify the artifact. After having the initial artifact, it is conducted the evaluation phase [107] using semi-structured interviews mentioned in Section 3.3, in order to support the findings of the initial research and improve the artifact.

3.1. Design science research

This Design Science Research (DSR) is divided into two processes [44,45,62], build (the process of construction of an artifact) supported by two MLR in Section 4, resulting in a proposal in Section 5 and evaluation (to determine how well the artifact behaves) in Section 6.

Baskerville et al. [5] underline the dual mandate of the DSR, which is to use acquired knowledge to solve problems, create changes, or improve existing solutions; and at the same time to generate knowledge, perceptions, and theoretical explanations. Hevner [44] calls attention to the pragmatism of DSR in cycles of relevance and rigor, in the creation of

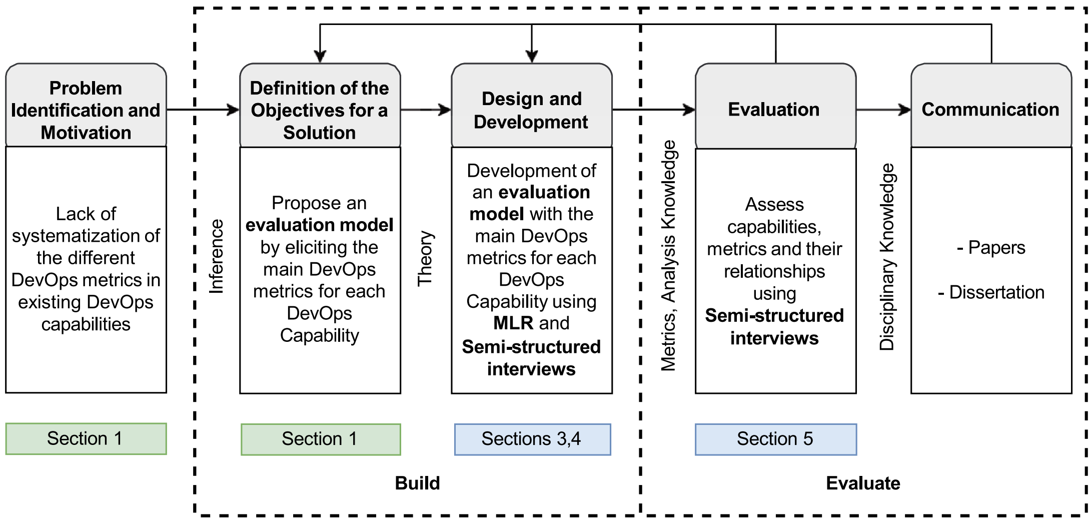

Problem Identification and Motivation
Lack of systematization of the different DevOps metrics in existing DevOps capabilities
Section 1
Inference
Definition of the Objectives for a Solution
Propose an evaluation model by eliciting the main DevOps metrics for each DevOps Capability
Section 1
Theory
Design and Development
Development of an evaluation model with the main DevOps metrics for each DevOps Capability using MLR and Semi-structured interviews
Sections 3,4
Build
Metrics, Analysis Knowledge
Evaluation
Assess capabilities, metrics and their relationships using Semi-structured interviews
Section 5
Disciplinary Knowledge
Communication
- Papers
- Dissertation
Evaluate

Fig. 4. Adapted phases of the DSR process model [45].

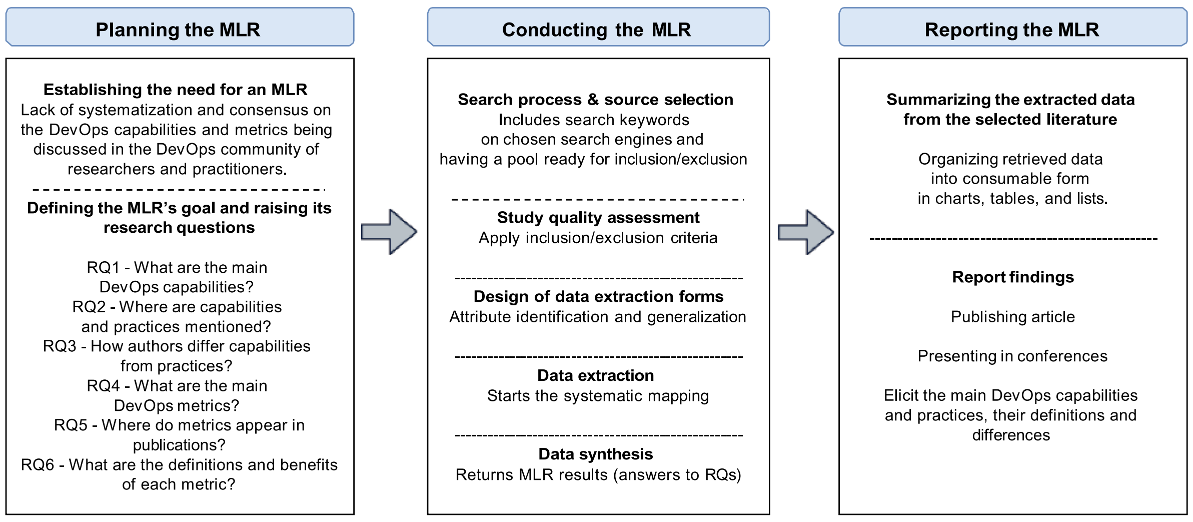

Planning the MLR
Establishing the need for an MLR
Lack of systematization and consensus on the DevOps capabilities and metrics being discussed in the DevOps community of researchers and practitioners.
Defining the MLR's goal and raising its research questions
RQ1 - What are the main DevOps capabilities?
RQ2 - Where are capabilities and practices mentioned?
RQ3 - How authors differ capabilities from practices?
RQ4 - What are the main DevOps metrics?
RQ5 - Where do metrics appear in publications?
RQ6 - What are the definitions and benefits of each metric?
Conducting the MLR
Search process & source selection
Includes search keywords on chosen search engines and having a pool ready for inclusion/exclusion
Study quality assessment
Apply inclusion/exclusion criteria
Design of data extraction forms
Attribute identification and generalization
Data extraction
Starts the systematic mapping
Data synthesis
Returns MLR results (answers to RQs)
Reporting the MLR
Summarizing the extracted data from the selected literature
Organizing retrieved data into consumable form in charts, tables, and lists.
Report findings
Publishing article
Presenting in conferences
Elicit the main DevOps capabilities and practices, their definitions and differences

Fig. 5. DevOps capabilities Multivocal Literature Review (MLR) Steps [38].

artifacts. The DSR methodology process in the field of Information Systems, as shown in Fig. 4, is based on Develop/build and justify/evaluate [45].

1. **Identification of the problem and motivation**, defining the research problem and justifying the value of a solution. The introduction identified the problem scope by being the lack of systematization of different DevOps metrics in existing DevOps capabilities, and a purpose was defined.
2. **Definition of objectives for a solution**, inferring the objectives of a solution from the problem definition and knowledge of the state of the problem and possible solutions. The objectives can be quantitative or qualitative. For this purpose, an evaluation model for DevOps capabilities work is proposed.
3. **Design and development** of the artifact's desired functionality and its architecture followed by its creation. Such artifacts can be, constructs, models, methods, or instantiations [45]. This research targets the development of an evaluation model of DevOps capabilities. For this stage, two MLRs and semi-structured interviews are used, as explained in Sections 3.2 and 3.3. The semi-structured interviews done with experts, individually, to avoid bias, as given in Section 5.1.3, help refine the capabilities and metrics knowledge gathered and are considered enough when data saturation in Section 5.1.4 is reached and confirmed.
4. **Evaluation** of the solution, by comparing the objectives and results, by doing iterative evaluations in Section 6.1.1 with another round of experts, observing and measuring how well the artifact supports a solution to the problem. The cases where a need was felt to consider the interview feedback to improve the effectiveness of the artifact went back to the design and development phase before continuing to communication [78].
5. **Communication** of the problem, the artifact, its utility, novelty, and effectiveness, as well as the rigor of its design to researchers and other relevant audiences. In this case, communicating in papers and a dissertation was done.

3.2. *Multivocal literature review*

An MLR is a type of Systematic Literature Review (SLR), which aims to incorporate gray literature like blogs, videos, web pages, and white papers, which are constantly produced by Software Engineering practitioners outside academic forums, notwithstanding the published (peer-

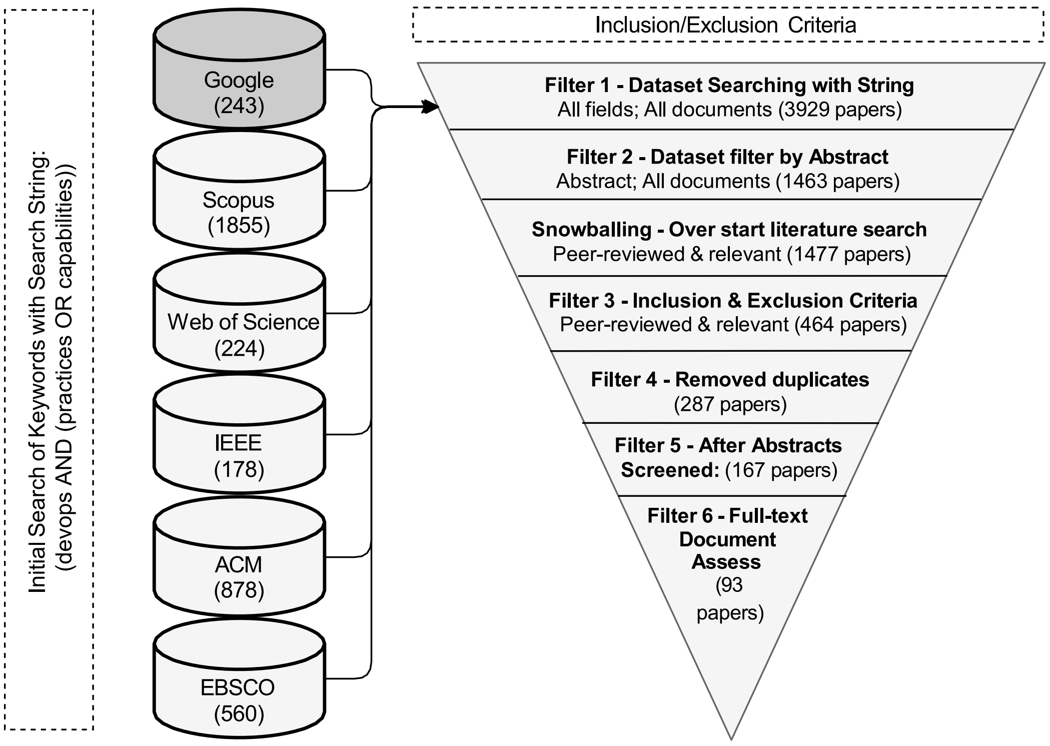

Initial Search of Keywords with Search String:
(devops AND (practices OR capabilities))
Google (243)
Scopus (1855)
Web of Science (224)
IEEE (178)
ACM (878)
EBSCO (560)
Inclusion/Exclusion Criteria
Filter 1 - Dataset Searching with String
All fields; All documents (3929 papers)
Filter 2 - Dataset filter by Abstract
Abstract; All documents (1463 papers)
Snowballing - Over start literature search
Peer-reviewed & relevant (1477 papers)
Filter 3 - Inclusion & Exclusion Criteria
Peer-reviewed & relevant (464 papers)
Filter 4 - Removed duplicates
(287 papers)
Filter 5 - After Abstracts Screened: (167 papers)
Filter 6 - Full-text Document Assess
(93 papers)

Fig. 6. DevOps capabilities multivocal literature review process (adapted) [38].

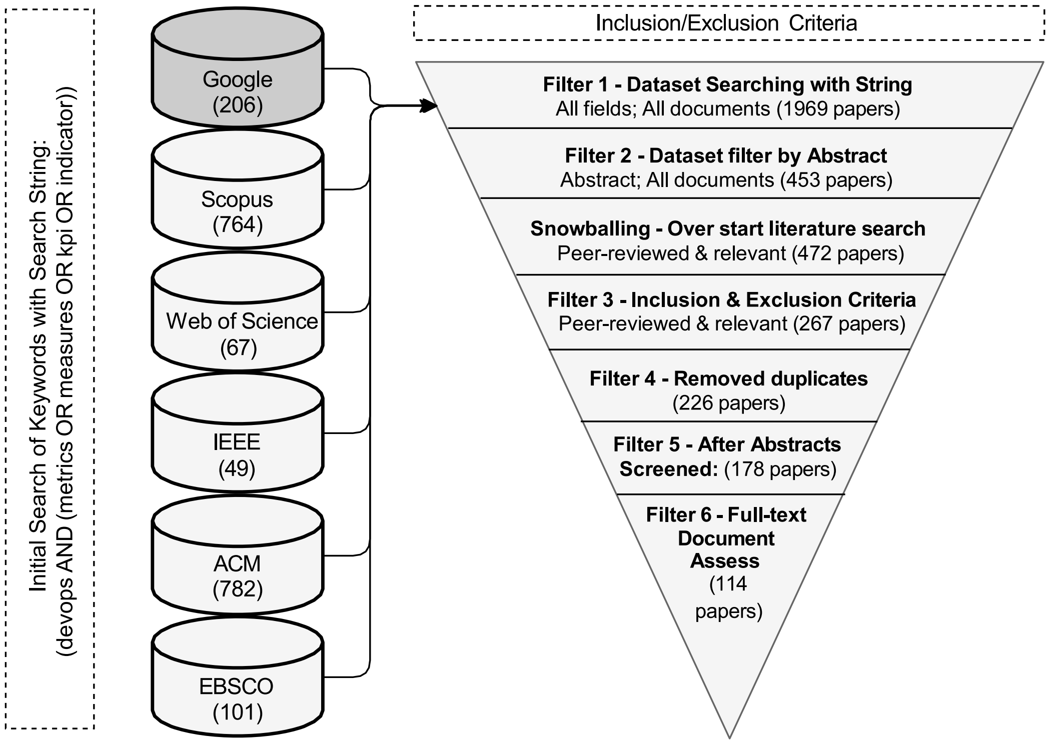

Initial Search of Keywords with Search String:
(devops AND (metrics OR measures OR kpi OR indicator))
Google (206)
Scopus (764)
Web of Science (67)
IEEE (49)
ACM (782)
EBSCO (101)
Inclusion/Exclusion Criteria
Filter 1 - Dataset Searching with String
All fields; All documents (1969 papers)
Filter 2 - Dataset filter by Abstract
Abstract; All documents (453 papers)
Snowballing - Over start literature search
Peer-reviewed & relevant (472 papers)
Filter 3 - Inclusion & Exclusion Criteria
Peer-reviewed & relevant (267 papers)
Filter 4 - Removed duplicates
(226 papers)
Filter 5 - After Abstracts Screened: (178 papers)
Filter 6 - Full-text Document Assess
(114 papers)

Fig. 7. DevOps metrics multivocal literature review process (adapted) [38].

reviewed) writing like journal articles and conference papers. Therefore, MLR is important for the expansion of the research by including literature that normally wouldn't be taken due to its "gray" nature [37, 38]. Some examples of successful DevOps research, in the same area, using MLR already exist [34,84,98], thus corroborating the practical usefulness of this method for the proposed research, expanding the diversity of sources that are available in a variety of forms, reflecting different purposes and perspectives [76].

Table 2
Filters used in the capabilities MLR protocol.

Database	Filter 1	Filter 2	Snowballing	Filter 3	Filter 4	Filter 5	Filter 6
Google	243	243	+14	89	89	77	75
Scopus	1855	342		157	42	40	2
Web Of Science	224	174		91	29	24	1
IEEE	178	146		67	67	14	8
ACM	878	92		22	22	6	4
EBSCO	560	475		38	38	6	3
Total	3929	1463	1477	464	287	167	93

*Legend:*Filter 1 = Query All fields, All documents; Filter 2 = Query Abstracts, All documents; Snowballing = Applied over starting literature search [38] Filter 3 = Relevant (inclusion/exclusion criteria); Filter 4 = Remove duplicates; Filter 5 = After Abstracts Screened; Filter 6 = Full-text Document Assess.

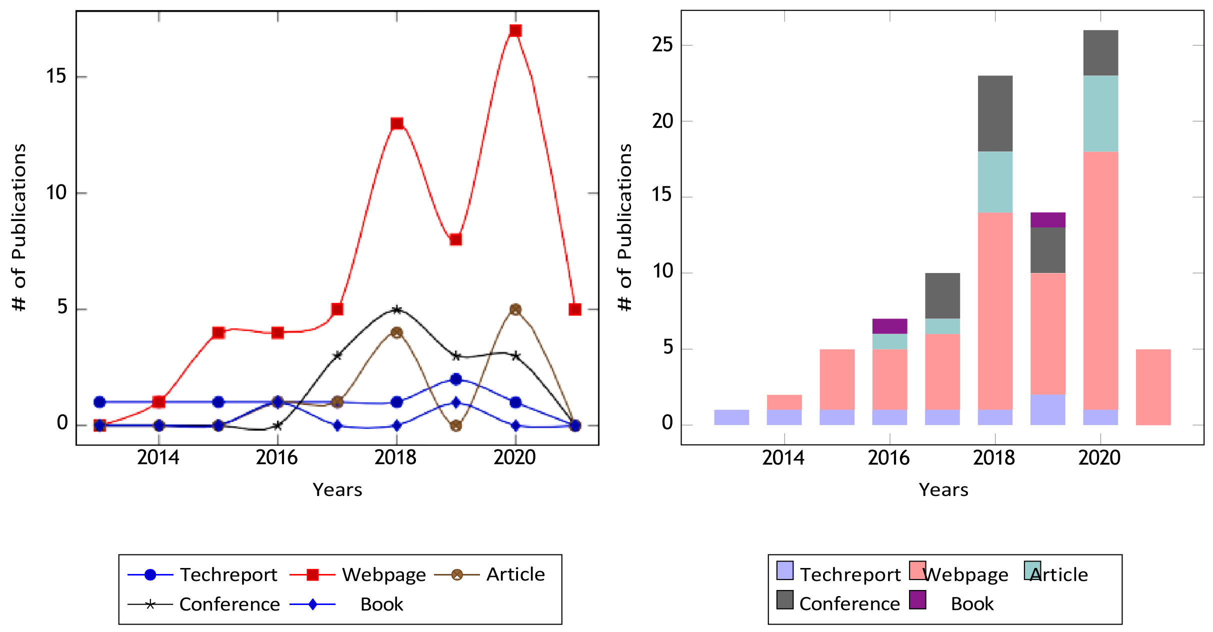


Fig. 8. Capabilities – distribution of publications per type over the years.

3.3. *Semi-structured interviews*

During these semi-structured interviews, an interview protocol is used and participants are informed about the research goals while asking for their consent to be interviewed. Interviews with a semi-structured format are a common approach in development research. It is common for people to give vital information that researcher had not before [92]. These kinds of interviews are a common method for qualitative research [2]. The researchers and participants engage in a formal interview, using a developed “interview guide” with a list of questions and topics. However, the interviewer can also follow other topics in the conversation that can guide when he or she feels this is appropriate [12, 13]. The DevOps capabilities confirmed in the literature, together with the related metrics to be found, are used to form an interview protocol in which the targeted evaluation model acts as a guide to elicit metrics from capabilities.

4. Capabilities and metrics multivocal literature reviews

This research was conducted in April 2021 using various keywords. The search strings used to perform the search to retrieve the maximum number of studies were: *(devops AND (practices OR capabilities))* and *(devops AND (metrics OR measures OR KPI OR indicator))*. Applied to the **datasets** of Google search, Scopus, Web of Science, ACM, and EBSCO. The full process shown in Fig. 5 exposes the planning, conducting, and reporting as proposed by Garousi et al. [38].

In order to facilitate searching and collecting high volumes of gray literature, code was developed as seen in Appendix B, Listing 1 (Python code for consistent fetching of a large number of Google search results) to parse the data into two CSV files [70]. This way, we can ensure obtaining repeatable and clean results that are not specific to the researcher. As part of the review protocol, after the search is completed the first set of papers is obtained, and snowballing is done, while inclusion and exclusion criteria are applied for refining the search results. The accepted criteria were that the publication had to be written in English, published in and after 2013 and 2010 respectively, with full-text accessibility and discussion on the mentioned topics. Videos with fully available spoken content were also taken into account [22,36, 81]. However, publications where the author is not identified; do not have a publication date; or are an advertisement or a job post are excluded. After that step, the abstracts are screened in order to evaluate the relevance they have to the research. Finally, the relevant papers are to obtain the final selection of studies to perform the review.

4.1. *Conducting the MLRs*

For reference, the complete summary of the review process is shown in the diagram in Figs. 6 and 7 with a visual representation of the applied MLR selection process. This reflects all the selection work done through the methodical process of MLR.

In the initial search step *filter 1* (All fields; All documents) was used together with the search string, as part of the MLR protocol to find the

Table 3
Filters used in the metrics MLR protocol.

Database	Filter 1	Filter 2	Snowballing	Filter 3	Filter 4	Filter 5	Filter 6
Google	206	206	+20	170	165	132	127
Scopus	764	140		87	69	11	2
Web of science	67	61		46	38	9	2
IEEE	49	38		35	25	8	3
ACM	782	28		19	17	7	3
EBSCO	101	66		21	19	8	2
Total	1969	539	558	377	333	175	139

*Legend:*Filter 1 = Query All fields, All documents; Filter 2 = Query Abstracts, All documents; Snowballing = Applied over starting literature search [38] Filter 3 = Relevant (inclusion/exclusion criteria); Filter 4 = Remove duplicates; Filter 5 = After Abstracts Screened; Filter 6 = Full-text Document Assess.

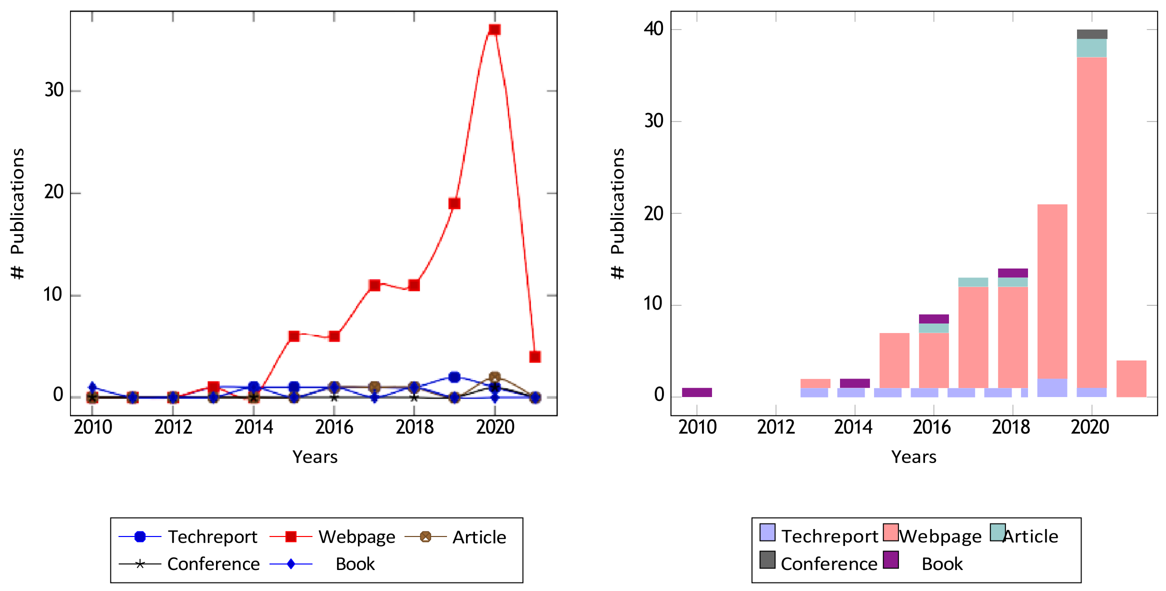


Fig. 9. Metrics - distribution of publications per type over the years.

final set of articles, which gives us a relation of the articles found with the filters used. In the MLR for DevOps capabilities or practices, the search string was (devops AND (practices OR capabilities)) as shown in Fig. 6 where the initial dataset was composed of 3929 publications and by the end of this process, we remain with 93 publications for full-text assessment. In the MLR for DevOps metrics, the search string (devops AND (metrics OR measures OR kpi OR indicator)) is shown in Fig. 7 where the initial dataset was composed of 1969 publications and after all processes, we remain with 114 publications for full-text assessment.

The initial screening of blogs and videos in particular, which do not have abstracts, was done over the full content, since these tend to be small when compared to academic literature, with a second, more in-depth pass on the last full-text assessment.

The relation of Gray and white literature documents set per database is reflected in Table 2 showing that 75 results came from Google search gray literature. For the cases of Scopus, Web Of Science, IEEE, ACM, and EBSCO they all contributed a total sum of 18 relevant research documents for capabilities.

Relevant to note is the distribution and growth of the selected papers shown over the years in relation to the publication shown in Fig. 8. This shows a growing interest in the last three years with an increase in the volume of research work related to researching capabilities, confirming the potential interest and usefulness of this research might have in the area.

First appearing the 2013 State of DevOps Report [87], the 2014 State of DevOps Report [88], and a webpage on “Six Core Capabilities of a DevOps Practice” [75]. In 2015, 2016, and 2017, the number of web pages and conference papers increased. The huge increase in web pages in 2018 indicates that practitioner publications have progressed substantially faster than scientific research. Despite a little decline in 2019, practitioners’ publications continued to expand in 2020, as shown in Fig. 8. Because both searches generated from the databases occurred in March 2021, the values for 2021 are lower.

The filtering protocol in the MLR and the relationship of the documents per database, as represented in Table 3 show that the vast majority of 109 results originated from Google search gray literature.

Only two publications are contributed by Web Of Science (1.75%). ACM, EBSCO, and Scopus each provided one item, leading to a total of three publications (2.63%) of relevant research articles.

This confirms the anticipation that practitioner findings will be more diverse than scientific findings. Despite the modest amount of research done on metrics, there has been an increase in interest in the area in recent years shown in Fig. 9, suggesting the potential attractiveness and utility of this research on the subject. Publications have evolved, with the majority of generated literature appearing on websites by 2020.

The *Techreports* from 2013 and 2014 had special importance in ramping up the interest in DevOps metrics topic and raising awareness of the fact that measurements are visible and actionable [87,88]. Finally, the number of gray literature articles increased considerably in 2020, as demonstrated by the massive increase in web pages related content in that year, showing that practitioner writings grew far faster than scientific research. In this study, 22 key metrics are found out of 58 DevOps metrics discovered. These metrics are reduced to 22 because the MLR found 10 years with relevant publications out of the 12 years, it was accordingly chosen to use the metrics that are cited 10 or more times as

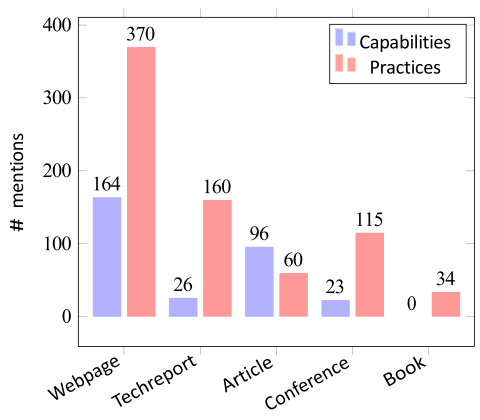


Fig. 10. Number of publications mentioning capabilities or practices among sources.

Table 4
Six publication properties identified from the MLR.

Property	Total publications
Interchangeably mentions capabilities or practices	66
Mentions capabilities directly	19
Presents different or reorganized capabilities compared to [101]	14
Distinguishes practices from capabilities	8
Indicates a definition for capability	6
Indicates a definition for practice	1

the key DevOps measure for this research.

Finally, in our review of the literature, we have identified that no previous studies have examined the relationship between capabilities and metrics in DevOps within a single research endeavor. This highlights a clear opportunity for our research to contribute to the field by addressing this important gap and providing valuable insights into the relationship between capabilities and metrics in the context of DevOps.

4.2. RQ1—What are the main DevOps capabilities or practices?

The MLR provides an answer to the first research question, by revealing a list of 37 capabilities. This list is based on the literature review of the 93 publications, also considering the fact that some practitioners mention them as practices and others as capabilities, as shown in Fig. 10. The capabilities and the relation of own, many publications make a reference to each one are as follows: C01-Cross-team collaboration and communication, mentioned 81 times; C02-Continuous integration, mentioned 80 times; C03-Continuous delivery and deployment automation, mentioned 78 times; C04-Proactive monitoring, observability, and autoscaling, mentioned 75 times; C05-Test automation and environments, mentioned 62 times; C06-Continuous improvement of processes and workflows, mentioned 46 times; C07-Version control system, mentioned 45 times; C08-Support learning culture and experimentation, mentioned 44 times; C09-Trust/empower teams to make decisions and changes, mentioned 42 times; C10-Focus on people, process, and technology, mentioned 32 times; C11-Configuration management, mentioned 30 times; C12-Cloud infrastructure and cloud-native, mentioned 30 times; C13-Artifacts versioning and registry, mentioned 28 times; C14-Loosely coupled architecture/ microservices, mentioned 27 times; C15-Database change management/ release alignment, mentioned 25 times; C16-Infrastructure as code, mentioned 25 times; C17-Emergency response/ proactive failure notification, mentioned 24 times; C18-Containerization, mentioned 24 times; C19-Open source software adoption, mentioned 22 times; C20-Shift left on security, mentioned 20 times; C21-Transformational leadership, mentioned 20 times; C22-Trunk-based development, mentioned 19 times; C23-Monitor systems to inform business decisions, mentioned 19 times; C24-Performance/Westrum organizational culture, mentioned 19 times; C25-Working in small batches, mentioned 16 times; C26-Centralized log management, mentioned 15 times; C27-Lightweight/streamlining change approval, mentioned 15 times; C28-Visibility of work in the value stream, mentioned 14 times; C29-Work in progress limits, mentioned 14 times; C30-Customer/user feedback, mentioned 13 times; C31-Blameless postmortems/reduced fear of failure, mentioned 12 times; C32-Data-driven approach for improvements, mentioned 12 times; C33-Job satisfaction, mentioned 11 times; C34-Test data

Table 5
Number of publications mentioning capabilities or practices.

ID	Capability	Mentioned as practice	Mentioned as capability	Total
C01	Cross-team collaboration and communication	66	15	81
C02	Continuous integration	63	17	80
C03	Continuous delivery and deployment automation	60	18	78
C04	Proactive monitoring, observability and autoscaling	57	17	74
C05	Test automation and environments	49	13	62
C06	Continuous improvement of processes and workflows	35	11	46
C07	Version control system	33	12	45
C08	Support learning culture and experimentation	32	12	44
C09	Trust/empower teams to make decisions and changes	31	11	42
C10	Focus on people, process and technology	25	7	32
C11	Configuration management	26	4	30
C12	Cloud infrastructure and cloud-native	21	9	30
C13	Artifacts versioning and registry	24	4	28
C14	Loosely coupled architecture/ microservices	18	9	27
C15	Database change management/ release alignment	15	10	25
C16	Infrastructure as code	20	5	25
C17	Emergency response/ proactive failure notification	13	11	24
C18	Containerization	17	7	24
C19	Open source software adoption	21	1	22
C20	Shift left on security	10	10	20
C21	Transformational leadership	10	10	20
C22	Trunk-based development	9	10	19
C23	Monitor systems to inform business decisions	14	5	19
C24	Performance/Westrum organizational culture	11	8	19
C25	Working in small batches	8	8	16
C26	Centralized log management	12	3	15
C27	Lightweight/streamlining change approval	7	8	15
C28	Visibility of work in the value stream	7	7	14
C29	Work in progress limits	6	8	14
C30	Customer/user feedback	5	8	13
C31	Blameless postmortems/ reduced fear of failure	8	4	12
C32	Data-driven approach for improvements	11	1	12
C33	Job satisfaction	5	6	11
C34	Test data management	2	8	10
C35	Chaos engineering	8	1	9
C36	Code maintainability	5	4	9
C37	Visual management capabilities	2	7	9

Table 6
Definition of DevOps capability.

A **DevOps capability**, is here defined as the ability to do something [10] in DevOps or by the quality, or state of being capable [66]. It consists of the combined skills [96] accumulated and developed by its members over time. As an example, DevOps technological capabilities are the information and skills - technical, managerial, and institutional - that enable productive enterprises [48,97] to utilize equipment and technology efficiently.

Table 7
Definition of DevOps practice.

A **DevOps practice**, is here defined as the action taken rather than thought or ideas [11] or by the act of carrying out [67] a DevOps activity. It can also be seen as an enabler [69,101,105] of a mentioned capability by an individual or a group such as the engineering team. As a result, authors will discuss capabilities from the perspective of evaluation, like an assessment, but refer to practices from the perspective of a hands-on approach.

management, mentioned 10 times; C35-Chaos engineering, mentioned 9 times; C36-Code maintainability, mentioned 9 times; C37-Visual management capabilities, mentioned 9 times.

4.3. RQ2—Where are capabilities or practices mentioned?

Based on the extended research enabled by this MLR, it is observed that capabilities have been mentioned interchangeably as practices in 66 publications in Table 4 and eight publications even distinguish practices from capabilities [4,61,68,101,112]. The two terms are described across several types of white and gray literature, as seen in Fig. 10.

It can be observed in the figure, that the webpages, overwhelmingly created by practitioners, have the most mentions as practices, but also include a substantial number of mentions as capabilities. The same happens in Techreport, Conference and Book, which are closer to the gray literature. Table 4 shows that one publication indicates a DevOps practice definition [112] to be a subset implementation of a capability.

On the other hand, the scientific articles make more mention of capabilities, which becomes a very interesting finding of this research, as it reveals that practitioners are focused more on DevOps practices, while the scientific community tries to organize capabilities in a way that abstracts more generic concepts applicable to building skills and enablers. Nevertheless, the concepts are the same, only at different stages of the process. A different example of this same interchangeability can be observed in a seminal book, "Continuous Delivery" [47] from 2010, which starts by mentioning the term capabilities. Despite not explicitly mentioning the word "DevOps" it describes, however, in detail the deployment pipeline pattern, which is usually central to DevOps capabilities. On page 109 of the book, the capability of deployment and production release is described in detail, explaining how the process is automated, with speed, repeatability, and reliability in mind. The same author mentions that when the "capability" of automating the process as normal events are available, releases become practically without risk.

4.4. RQ3—How do authors distinguish capabilities from practices?

It is largely observed that the word "capability" is used when the observation is external or at a high-level overview. It is then a matter of perspective. When talking about a capability, we see a third-party assessment of something that is being looked at from the outside, while observing a group to see what they are capable of doing. Whereas, a practice is seen from the standpoint of the internal team or group, realizing "I am doing these things". That ability converted to action is then mentioned with the term "practice". Therefore, the authors will speak about capabilities from an evaluation standpoint, and practices from a hands-on approach perspective. The number of publications mentioning capabilities or practices is organized in Table 5.

The capability definition points to an organization's "ability" to perform or achieve a certain process, whereas practice is referred to more at the level of DevOps practitioners and thus more observed in the gray literature publications. Clear examples of this more formal research concept were presented earlier by [101,105] and more recently in the book Accelerate [31], in DORA (DevOps Research and Assessment) [19, 24,40] and in several journal articles or conferences [4,61,68,69,96, 112,113].

A capability is also mentioned as a "construct" [31,32] and there are "capabilities we are building" [53] in order to enable the organization for a certain practice. Organizations should target developing capabilities and habits in their people [97] as an enabler for continuous improvement and functional skills. It is now clear that, despite some confusion still existing while interchanging these two words, as seen in Fig. 10 and Table 5, we are, in reality, talking about the same fundamental concepts. The usage of capabilities or practices is not consensual, and because this study is not about achieving that consensus, it is chosen to use the term **capabilities**; nevertheless, others might also use practices. A consensus between the two should be achieved, therefore a definition of DevOps capabilities is purposed in Table 6 and DevOps practices in Table 7.

The uses of DevOps practice or capability as labels are contextual, and differ depending on the perspective, as can be seen from the definition of capability defined in Table 7.

As a result, the authors will discuss capabilities from the perspective of evaluation but refer to practices from the perspective of a hands-on approach. Therefore, this research will use the term "capability" now, since that is applicable to a more scientific evaluation of DevOps adoption. Moreover, it is seen that the capabilities are, in fact, dynamic and have been changing over the years. The list of the most studied and approached capabilities was collected as it was proposed. Of which, it is highlighted *cross-team collaboration and communication*. In this collaboration, developers have access to a self-service platform that provides a foundation for automation, standardization, and team autonomy to enable the other three most mentioned capabilities: *continuous integration*; *continuous delivery and deployment automation*; *proactive monitoring, observability, and autoscaling*.

These capabilities bring several types of advantages which can be grouped into technical, cultural, and business benefits, Kim et al. [53]. The technical outcomes are mostly in the practices of Continuous Integration, Delivery, and Reliability [9].

4.5. RQ4—What are the main DevOps metrics?

There are 22 main DevOps metrics found in the MLR, gathered from all the publications, selected for review. The metrics are the following: M01 - Mean Time To Recover/Restore (MTTR), with 96 mentions; M02 - Mean Lead-time for Changes (MLT), with 91 mentions; M03 - Deployment Frequency (DF), with 88 mentions; M04 - Change Failure Rate (CFR), with 72 mentions; M05 - Service Availability and Uptime, with 31 mentions; M06 - Deployment duration time, with 30 mentions; M07 - Mean Time To Detection (MTTD), with 26 mentions; M08 - Application response time, with 23 mentions; M09 - Defect escape rate, with 21 mentions; M10 - Cycle Time Value (CTV), with 21 mentions; M11 - Service Level Agreements (SLAs) and Service Level Objectives (SLOs), with 21 mentions; M12 - Deployment size, with 20 mentions; M13 - Production Error and Incident rate, with 19 mentions; M14 - Customer tickets Volume and Feedback, with 19 mentions; M15 - Mean time to failure (MTTF), with 17 mentions; M16 - Customer Usage and traffic, with 16 mentions; M17 - Pipeline automated tests success/fail rate, with 14 men - ons; M18 - Westrum organizational culture measures, with 14 men - ons; M19 - Automated Test Code Coverage, with 13 mentions; M20 - Work in Progress (WIP) /Load, with 12 mentions; M21 - Unplanned Work Rate (UWR), with 11 mentions; M22 - Wait Time, with 10 mentions;

Table 8
Purpose and total references for each main DevOps metric.

ID	Metric	Purpose	Total References
M01	Mean Time To Recover/ Restore (MTTR)	Measures the mean of the time required to recover or restore service from a failure in production.	96
M02	Mean Lead-time for Changes (MLT)	Indicates how long it takes for a change to go from code committed to code successfully running in production.	91
M03	Deployment Frequency (DF)	Checks how often changes are deployed to production.	88
M04	Change Failure Rate (CFR)	Informs how often a change in production fails and must be immediately remedied.	72
M05	Service Availability and Uptime	Shows the percentage of service available during a period of time.	31
M06	Deployment duration time	Informs on how long it takes to deploy a set of changes.	30
M07	Mean Time To Detection (MTTD)	Measures the mean of the time required to detect a failure in production.	26
M08	Application response time	How an application responds to increases or decreases in user traffic and activity.	23
M09	Defect escape rate	Indicates the number of defects discovered in production versus the number of defects found during development.	21
M10	Cycle Time Value (CTV)	Provides information on the full Cycle Time Value, beginning with the idea and finishing with user feedback..	21
M11	Service Level Agreements (SLAs) and Objectives (SLOs)	Sets customer expectations for service availability with SLA and internal teams with SLO.	21
M12	Deployment size	Shows the number of changes incorporated in each production release.	20
M13	Production Error and Incident rate	Measures the frequency of faults and incidents in production following a deployment.	19
M14	Customer tickets Volume and Feedback	Indicates the level of satisfaction of customers using their feedback.	19
M15	Mean time to failure (MTTF)	Exposes the average time a flawed deployment into a system will manage to run until it fails.	17
M16	Customer Usage and traffic	Measures usage and traffic of customer-facing applications when there are defined business goals to increase.	16
M17	Pipeline automated tests success/fail rate	Shows the rate of success/ failure of Pipeline automated test jobs.	14
M18	Westrum organizational culture measures	Results of the Westrum cultural assessment [122]	14
M19	Automated Test Code Coverage	Measures how many lines, statements, or blocks of code are tested using the suite of automated tests.	13
M20	Work in Progress (WIP) /Load	Presents the number of open issues of each type (story, defect, task).	12
M21	Unplanned Work Rate (UWR)	Indicates the amount of time spent on tasks that weren't in the initial plan. Shouldn't be over 25%.	11
M22	Wait Time	Measures the amount of time spent waiting for the next step to add value.	10

4.6. RQ5—What is the purpose of each metric?

From the investigation done in this MLR, Table 8 shows the list of the main DevOps metrics including their ID, metric name, purpose description, the references that mention each specific metric, and the total of references discovered. The ID is used across the paper as a reference.

The 22 key metrics presented here are among the initial 58 identified in the MLR. They were referenced ten times or more and are relevant to organizational and mission goals like operational efficiency, customer happiness, profitability, productivity, and product or service quality [95].

4.7. RQ6—Why is each metric important?

We now know what each metric is, but do not know why is it necessary to measure each one? What is the significance or importance of each metric? Here, the MLR goes over *why* these DevOps metrics are important.

M01. *Mean Time To Recover/Restore (MTTR)* handling unexpected outages should be as quick as possible and a focus for DevOps KPI monitoring, as it contributes to greater customer satisfaction, faster application delivery, and better cost control [17,29,33,69].

M02. *Mean Lead-time for Changes (MLT)* is the hardest to measure of the top four key DevOps metrics. It is important to know how long it takes to deploy a change, and understand delaying problems like technical debt [43,87,91,95].

M03. *Deployment Frequency (DF)* happens many times per day for elite industry performers [19,59], where there is continuous development, testing, and integration of small changes and continuously improving applications. Important to respond quickly to business requests for new features and to critical issues [108,120].

M04. *Change Failure Rate (CFR)* should be as low as possible in DevOps. It is a critical metric that businesses must monitor since unsuccessful deployments result in revenue losses and dissatisfied consumers [15,23].

M05. *Service Availability and Uptime* should be more than 99.999% of the time for users of the system, with a ratio based on availability, reliability, and uptime. Achieving 100% availability is unrealistic, once planned downtime for maintenance is accounted for. Therefore, it is important to track and distinguish from unplanned downtime [17,74].

M06. *Deployment duration time* allows tracking the progress of the deployment. It can help identify potential problems and allow a dramatic increase in revenue by using that extra time to develop more value-added services [72,120,121].

M07. *Mean Time To Detection (MTTD)* demonstrates the effectiveness of monitoring technologies and intelligent alerting techniques, assessing whether current response efforts are appropriate. High detection times may result in bottlenecks [23,28].

M08. *Application response time* that is long may indicate bottlenecks that require attention, since it degrades user experience and satisfaction. The cause might be code, data access, protocol problems, or a variety of other factors [17,115].

M09. *Defect escape rate* is an important DevOps metric to track how often defects make it to production. Abnormally high defect rates could be the first sign of problems in testing, qualification, or in team performance [17,93,104].

M10. *Cycle Time Value (CTV)* is the time it takes to go from idea to production, spanning all the steps of build, test, stage, and push to production. Slowing down the cycle with manual testing or assessments creates friction for developers and the business [23,100].

M11. *Service Level Agreements (SLAs) and Objectives (SLOs)* serve to define external and internal availability goals as commitments between providers, clients, and internal teams, defining how fast releasing is possible while measuring that performance with Service Level Indicators (SLIs) [8,9,19,23,71,118].

Table 9
First batch of interviews with practitioners' details.

ID	Interview date	Current position	Company	Years of DevOps practice	Age (years)	Country
I01	2021-05-07	Principal Architect	Google	More than 3 years	45–54	United States
I02	2021-05-14	Indvidual contributor	Noesis	Between 1 year and 3 years	25–34	Portugal
I03	2021-05-14	Team Lead	Freelance	Between 1 year and 3 years	35–44	Portugal
I04	2021-05-17	Indvidual contributor	Newdecision	More than 3 years	45–54	Portugal
I05	2021-05-18	Team Lead	Siemens	Between 1 year and 3 years	25–34	Portugal
I06	2021-05-18	Indvidual contributor	Azores Government	Between 1 year and 3 years	45–54	Portugal
I07	2021-05-19	Principal	Inuits.eu	More than 3 years	45–54	Belgium
I08	2021-05-19	Team Lead	IBM	More than 3 years	45–54	United States
I09	2021-05-21	Team Lead	amazee.io	Between 1 year and 3 years	25–34	Switzerland
I10	2021-05-22	Indvidual contributor	Acquia	Less than 1 year	45–54	UK
I11	2021-05-24	Software Infrastructure Engineer	Freelance	Between 1 year and 3 years	45–54	Spain
I12	2021-05-26	Team Lead	Siemens	More than 3 years	35–44	Portugal
I13	2021-05-27	Indvidual contributor	Drupal Association	More than 3 years	35–44	United States
I14	2021-05-27	Manager	Manifold	More than 3 years	25–34	United States
I15	2021-05-31	Director	Bloomidea	More than 3 years	35–44	Portugal
I16	2021-06-01	Director	Deeper Insights	More than 3 years	35–44	Portugal
I17	2021-06-07	Team Lead	Facebook	More than 3 years	35–44	United States
I18	2021-06-09	VP	Acquia	More than 3 years	> 55	United States
I19	2021-06-10	Senior Security Engineer	Acquia	Between 1 year and 3 years	25–34	India
I20	2021-06-15	Director	Dropsolid	More than 3 years	35–44	Belgium
I21	2021-06-18	Indvidual contributor	Acquia	Between 1 year and 3 years	25–34	India

Table 10
Total contributions from participants during the build phase.

	Relations			Updated or added		
Interview	Contributed	Total	Percentage	Capabilities	Metrics	Categories
I01	18	18	2.03%	0	2	1
I02	14	32	1.58%	1	0	0
I03	26	58	2.93%	3	0	0
I04	22	80	2.48%	0	0	0
I05	17	97	1.91%	0	0	0
I06	23	120	2.59%	0	0	0
I07	35	155	3.94%	1	0	0
I08	28	183	3.15%	2	2	0
I09	24	207	2.70%	0	0	0
I10	20	227	2.25%	0	0	0
I11	15	242	1.69%	2	0	0
I12	18	260	2.03%	0	0	0
I13	13	273	1.46%	0	1	0
I14	9	282	1.01%	0	0	0
I15	12	294	1.35%	0	0	0
I16	11	305	1.24%	0	0	0
I17	8	313	0.90%	0	0	0
I18	6	319	0.68%	0	0	0
I19	7	326	0.79%	0	0	0
I20	4	330	0.45%	0	0	0
I21	2	332	0.23%	0	0	0

🟨 Data saturation, showing residual gains $< 1\%$.

M12. *Deployment size* is important to keep an eye on the deployment artifacts that are shipped to production with each release and track the number of bug fixes and feature requests delivered [23,28,104].

M13. *Production Error and Incident rate* tells the DevOps team how often new bugs appear in running applications. It is important to capture spikes in the error rate because these can indicate that something is not right. Not all errors are equally impactful on customers' trust [17,52, 114].

M14. *Customer tickets Volume and Feedback* is a good assessment of a successful DevOps adoption. Reduce customer tickets by preventing bugs from reaching production, and repairing them as fast as possible if they do, improving quality. Customer satisfaction leads to a competitive advantage [17,28,46,101].

M15. *Mean time to failure (MTTF)* is an indication of how long on average the system or a component can run before failing after deployment. It can indicate problems with the deployment or quality of the software. For example, there may not be enough tests covering different scenarios that might contain bugs [18,50,52].

M16. *Customer Usage and traffic* metrics allow tracking user engagement with application features. Increased engagement after an update may indicate users are pleased with the updates. If traffic reports indicate too much or no activity, there might be an issue, suggesting a malfunctioning component is generating the anomaly [28,104,120].

M17. *Pipeline automated tests success/fail rate* is another contributor to the speed of the DevOps process. Automated tests are faster than humans, but they should deliver the right results. To increase velocity, the team needs to make extensive usage of unit and functional testing. It is important to know how often changes are causing tests to break [17,104,120].

M18. *Westrum organizational culture measures* results are key to fostering a performance-oriented organizational environment, stating [122]. DevOps teams must be supported by transformational leadership [16,57] to enable this culture, thus empowering strategic alignment and reducing conflict [32,51,90,121].

M19. *Automated Test Code Coverage* is a DevOps best practice for measuring the percentage of the unit or integrity test coverage [28,103].

M20. *Work in Progress (WIP) /Load* is a lean manufacturing principle shown in the Toyota Production System [97] that enhances teams overall throughput by limiting work in progress (partially completed work). This increases the total velocity [14,35,39].

M21. *Unplanned Work Rate (UWR)* tracks the amount of time spent on unplanned work. A high Unplanned Work Rate (UWR) indicates that efforts are wasted on unexpected errors that were not identified early in the workflow. The difference between acting on warning signs or having an unexpected outage [6] is defined as unplanned work [23,89].

M22. *Wait Time* (queuing or waste) is an estimate of the time that the work item spends idle in a non-productive state during its processing by the value stream. Wait time is in opposition to touch time when value is created [39,53,72].

5. Research proposal

In research development, semi-structured interviews are a popular method. Unlike formal interviews, which follow a strict set of predefined questions, they are more conversational and focused on specific themes [92].

5.1. Interviews with practitioners

For the construction of the proposed artifact, this research used the results from the two MLR done in Section 4 and the first set of 21 semi-structured interviews out of the total of 31 Interviews done to DevOps practitioners seen in Tables 9 and 15. Semi-structured interviews were held online using Zoom (https://zoom.us) or Jitsi videoconferences (https://jitsi.org) to broaden the possibility of reaching out to more practitioners worldwide. Invites were sent to major private and public companies in the IT sector, specialized conference attendees, and LinkedIn professionals. But we only agreed to interview people with a minimum experience in the sector of at least one year.

Slots of 45 min were made available via a webpage created for practitioners to signup for interviews at their preferred time and date, between May and July 2021. The page informed participants of the research interview confidentiality and aimed to increase our understanding of how DevOps Capabilities can be further measured with their valuable opinion. The page also mentioned the research questions involved. A link to a spreadsheet was shared with participants before the interview showing the resulting lists of DevOps capabilities in Section 4.2 and metrics in Section 4.5 from the MLR. In this process, a form was used to collect the characterization presented in Table 9. The first set with 21 interviews aimed to respond to the research questions present in this section, leading to a proposed artifact seen in Section 5.2. The second set aimed to evaluate the artifact in Section 6 by using 10 iterations of interviews.

5.1.1. Preparation

Preparation was done considering each practitioner's specific session to expedite and facilitate the interview process. After the participants gave their consent to start the interview, a spreadsheet with capabilities and metrics resulting from the MLR, which was empty at first, was screen-shared, and subsequently got populated and refined with answers, and the interviews followed a semi-structured framework (shown in Appendix A, Table A.19), aimed to focus on the research problem present in Section 1. Before conducting each following semi-structured interview, close observation was done of the previous interview results to have a thorough grasp of the issue and to reformulate appropriate semi-structured questions if needed [12]. As a result, better open-ended questions relating to the problem were included, to give the possibility for discovering new approaches to challenge and comprehend the artifact's evolution.

5.1.2. Practitioners characterization

A total of 21 interviews were conducted in this section to build the research proposal. The practitioners interviewed and shown in Table 9 are: three individual contributors, six team leads and two principal engineers. The companies they work for are very diversified: Google, Noesis, Newdecision, Siemens, Azores Government, Inuits.eu, IBM, amazee.io, Acquia, Drupal Association, Manifold, Bloomidea, Deeper Insights, Facebook, Dropsolid and some freelancers. Twelve practitioners have more than 3 years of experience in DevOps, while eight practitioners have between 1 year and 3 years of experience and only one has less than 1-year of experience. Of all the 21 practitioners, eight are individual contributors with diverse roles, six are Team Leads, two are Principals and the other five are at the management level or above.

The majority of the seven practitionersares between 35and 44 years of age, six are between 25 and 34 years of age, seven are between 45 and 54 years and one has more than 55 years of age. Their locations are Belgium, India, Portugal, Spain, Switzerland, the USA, and the United Kingdom.

5.1.3. Conducting

The same research questions, including "What are the main DevOps Capabilities or practices?" from Section 4.2 and "What are the main DevOps Metrics?" from Section 4.5, were asked to practitioners during the semi-structured interviews to confirm the findings in the two MLRs, with the addition of Section 5.1.5 "What are the main metrics for each DevOps capability?", present in this section. Interviewees validated the suggested set of capabilities and metrics, with one deeper validation occurring through a follow-up interview from I12. For each participant, a digestible classification of capabilities and measurements emerged, as well as the basic relationships of the capabilities that have an influence on which metrics. In other cases, validation resulted in the capabilities and metrics being adjusted or increased to align with the interviewee's ideas.

The first interviewed practitioner (I01) accorded substantial importance to the cultural aspects, starting with team collaboration capability (C01) first and then other capabilities, always focusing more on the culture measures, which are reflected in Table 13 and also reinforcing that it was his personal opinion that organizations need to come up with a process that makes sure that people are being included despite their diversity, culture, religion, or the fact of being remote. All that can be reflected in two new added cultural KPIs, suggested being **team happiness** and **talent retention**, which are part of the Westrum organizational culture (C24). Therefore, with the evolution of the research with other interviews, the previously defined metric *team happiness* (M42) was pulled in from the MLR and a new outcome, *talent retention* (M59) was added as a valuable metric. When talent is leaving the organization in large numbers, it should raise a flag to management to review the related capabilities' status. This was also confirmed by all the practitioners in this first phase of interviews when asked. *Talent retention* should be measured monthly and can be simply expressed as seen in Eq. (1).

$$Talent\ retention = \frac{Employees\ number - Employees\ departed}{Employees\ number} \tag{1}$$

The goal must be to decrease manual work and the delivery time as

Table 11
Categorization of DevOps capabilities.

Category	ID	DevOps capability
Cultural	C01	Cross-team collaboration and communication
Technical	C02	Continuous Integration
Technical	C03	Continuous Delivery and Deployment automation
Measurement	C04	Proactive Monitoring, Observability and autoscaling
Technical	C05	Test Automation and environments (Continuous testing)
Process	C06	Continuous Improvement of processes and workflows
Technical	C07	Version Control System
Cultural	C08	Support learning culture and experimentation
Technical	C09	Trust/empower teams to make decisions and changes
Process	C10	Focus on people, process and technology
Technical	C11	Configuration Management
Technical	C12	Cloud infrastructure and cloud-native
Technical	C13	Artifacts versioning and registry
Technical	C14	Loosely coupled architecture/ microservices
Technical	C15	Database change management/ release alignment
Technical	C16	Infrastructure as Code
Measurement	C17	Emergency response/ proactive failure notification
Technical	C18	Containerization
Cultural	C19	Open source software adoption
Technical	C20	Shift left on security
Cultural	C21	Transformational leadership
Technical	C22	Trunk-based development
Measurement	C23	Monitor systems to inform business decisions
Cultural	C24	Performance/Westrum organizational culture
Process	C25	Working in small batches
Technical	C26	Centralized log management
Process	C27	Lightweight/streamlining change approval
Process	C28	Visibility of work in the value stream
Measurement	C29	Working in progress limits
Process	C30	Customer/user feedback
Cultural	C31	Blameless Postmortems/reduced fear of failure
Process	C32	Data-driven approach for improvements
Cultural	C33	Job satisfaction
Technical	C34	Test data management
Technical	C35	Chaos Engineering
Technical	C36	Code maintainability
Measurement	C37	Visual management capabilities

Table 12
Categorization of main DevOps metrics.

Proposed category	ID	Main metric
Operating KPI	M01	Mean Time To Recover/Restore (MTTR)
Change KPI	M02	Mean Lead-time for Changes (MLT)
Change KPI	M03	Deployment Frequency (DF)
Change KPI	M04	Change Failure Rate (CFR)
Operating KPI	M05	Service Availability and Uptime
Change KPI	M06	Deployment duration time
Operating KPI	M07	Mean Time To Detection (MTTD)
Operating KPI	M08	Application response time
Change KPI	M09	Defect escape rate
Change KPI	M10	Cycle Time Value (CTV)
Operating KPI	M11	SLAs and SLOs
Change KPI	M12	Deployment size
Operating KPI	M13	Production Error and Incident rate
Business KPI	M14	Customer tickets Volume and Feedback
Change KPI	M15	Mean time to failure (MTTF)
Business KPI	M16	Customer Usage and traffic
Change KPI	M17	Pipeline automated tests success/fail rate
Cultural KPI	M18	Westrum organizational culture measures
Change KPI	M19	Automated Test Code Coverage
Operating KPI	M20	Work in Progress (WIP) /Load
Operating KPI	M21	Unplanned Work Rate (UWR)
Operating KPI	M22	Wait Time
Cultural KPI	M42	Team Happiness
Cultural KPI	M59	Talent retention

much as possible, while still explaining why we do what we do. Another important point brought up is that automation is not only in the code pipeline, it is end-to-end; automation will be part of almost everything in DevOps.

The second interview (I02) brought new knowledge about the definitions and categories of capabilities, giving a good basis for the result worked on throughout all the interviews, as seen in Table 11. In this interview, there was more focus on the top four metrics, giving examples that suggest high relevance of these outcomes for organizations.

In interview three (I03) practitioners emphasized the importance of team style and the various approaches used in each organization, as well as a considerable number of metrics, reflected in Table 12. Good automated security systems are rare, expensive, and complex, with security ending up being handled to the left of the process but usually in a manual form. It should also be common for the security team to examine a release as an all rather than a single code review. There Shift left on security (C20), which could positively interfere with (C23) monitor systems to inform the business decision, when fully automated. In turn, job satisfaction (C33) is essential for a well-functioning company to run a business. Finally, organizations, where the measures to be reviewed are well-defined, perform best since there are fewer inconsistencies between data providers and data consumers.

The fourth interview (I04) focused on reconciling the relationships between the capabilities that influence each metric and cementing the categorization of each one. It was mentioned that by experience, the transformation driven by DevOps capabilities does have an impact on several of the metrics leading to a better work-life balance and positive business outcomes, like in the case of Mean Lead-time for Changes (MLT) (M01), Mean Time To Recover/Restore (MTTR) (M02), Change Failure Rate (CFR) (M04) and several others with a special focus on the Emergency response (C17) in order automate response to failures. Cross-team collaboration (C01), *Continuous Integration* (CI) (C02) and *Continuous Delivery or Deployment* (CD) (C03) were again mentioned to have sustained relevance in the practitioner's company, giving several examples where the metrics impact was witnessed internally.

The concept of DevOps capability defined in Section 4 was also confirmed in the fifth interview (I05) practitioner. There's also value in pushing the capability Shift left on security (C20) earlier in the process to automate and optimize delivery while reducing the number of major security issues in production. Due to their engagement in operational operations, the Work in Progress (WIP)/Load (M20) metric was highlighted as having a much higher weight.

In the sixth interview (I06), the practitioner expressed an interest in implementing the study's findings in the current role, even though many of the capabilities had yet to be implemented in the organization. Many questions were placed relative to the connection of the version control system (C07) to continuous integration (C02), the centralization of logs (C26), Trunk-based development (C22), and user feedback (C30) because it passes through a formal internal hierarchical structure. Other valuable experiences were given, focusing on the visibility of the work in the value stream (C28), Code maintainability (C36) and the importance of feedback in determining whether the implementation is effective in production.

For interview seven (I07), the practitioner showed a high level of experience in giving refreshed directions to the capabilities relating to metrics. Capabilities needed to be grown from the team level up and supported by Transformational leadership (C21). Team happiness (M42) and talent retention (M59) were pointed out as critical, but it is really hard to measure the path to get there. Most failures in DevOps adoption reside in the team happiness factor. Capability (C10) stated that new tooling impacts the process and ultimately the people's culture. But changing individuals will only work for some, while others will remain apathetic. Many businesses can improve their configuration management (C11) and don't need all the cloud or cloud-native benefits (C12) because their demands are minimal, and their load is predictable. Artifacts (C13) are a must, and (C18) Containerization is great when done well. Early DevOps conferences advocated for an open-source attitude (C19), claiming ownership in-house, and avoiding vendor lock-in. More recently, in March 2021, when a large data center (OVH) burned down [56], his customers were able to withstand the outage due to the

Table 13
DevOps capabilities influencing main metrics.

ID	M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	M42	M59
C01	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓				✓					✓	✓
C02	✓	✓	✓	✓		✓			✓	✓		✓			✓		✓		✓					
C03	✓	✓	✓	✓		✓				✓					✓			✓				✓		
C04	✓				✓		✓	✓			✓		✓			✓					✓			
C05		✓	✓	✓						✓			✓	✓		✓	✓		✓				✓	
C06	✓	✓	✓							✓								✓		✓	✓	✓		
C07	✓	✓	✓										✓						✓			✓	✓	✓
C08			✓					✓		✓			✓				✓	✓					✓	✓
C09	✓	✓	✓	✓	✓	✓				✓			✓	✓	✓		✓	✓		✓		✓	✓	✓
C10		✓	✓											✓				✓					✓	✓
C11		✓		✓															✓				✓	✓
C12	✓	✓	✓		✓	✓		✓		✓	✓										✓	✓	✓	
C13	✓		✓				✓					✓									✓	✓		
C14	✓	✓	✓	✓				✓			✓	✓		✓	✓					✓		✓		
C15	✓		✓	✓	✓				✓				✓	✓	✓						✓		✓	
C16	✓	✓			✓	✓	✓			✓		✓								✓	✓	✓	✓	
C17	✓				✓		✓	✓			✓			✓						✓	✓	✓		
C18	✓		✓	✓	✓	✓		✓			✓	✓					✓		✓		✓		✓	
C19		✓	✓							✓									✓			✓	✓	✓
C20	✓	✓	✓	✓	✓		✓	✓	✓					✓						✓	✓		✓	✓
C21	✓	✓	✓	✓	✓					✓				✓				✓					✓	✓
C22	✓	✓	✓	✓					✓	✓		✓	✓					✓			✓		✓	
C23	✓			✓	✓			✓					✓				✓							
C24	✓	✓	✓	✓			✓								✓			✓				✓	✓	✓
C25		✓	✓			✓					✓	✓	✓	✓				✓		✓	✓	✓		
C26	✓			✓	✓		✓						✓	✓							✓	✓	✓	
C27	✓	✓	✓							✓								✓					✓	
C28	✓		✓	✓									✓	✓							✓		✓	
C29	✓	✓	✓															✓		✓			✓	✓
C30	✓			✓					✓				✓	✓		✓								
C31			✓	✓	✓	✓					✓		✓	✓		✓	✓			✓	✓	✓	✓	✓
C32	✓	✓	✓	✓							✓												✓	
C33	✓		✓	✓														✓					✓	✓
C34			✓	✓													✓					✓		
C35	✓			✓	✓		✓		✓		✓		✓	✓	✓		✓	✓			✓			✓
C36		✓																			✓	✓	✓	
C37	✓	✓	✓	✓		✓				✓	✓									✓	✓		✓	

*Legend:*M01 - Mean Time To Recover; M02 - Mean Lead-time for Changes; M03 - Deployment Frequency; M04 - Change Failure Rate; M05 - Service Availability and Uptime; M06 - Deployment duration time; M07 - Mean Time To Detection; M08 - Application response time; M09 - Defect escape rate; M10 - Cycle Time Value; M11 - SLAs and SLOs; M12 - Deployment size; M13 - Production Error and Incident rate; M14 - Customer tickets Volume and Feedback; M15 - Mean time to failure; M16 - Customer Usage and traffic; M17 - Pipeline automated tests success/fail rate; M18 - Westrum organizational culture measures; M19 - Automated Test Code Coverage; M20 - Work in Progress/Load; M21 - Unplanned Work Rate; M22 - Wait Time; M42 - Team Happiness; M59 - Talent retention. C01 - Cross-team collaboration and communication; C02 - Continuous Integration; C03 - Continuous Delivery and Deployment automation; C04 - Proactive Monitoring; Observability and autoscaling; C05 - Test Automation and environments; C06 - Continuous Improvement of processes and workflows; C07 - Version Control System; C08 - Support learning culture and experimentation; C09 - Empower teams to make decisions and changes; C10 - Focus on people; process and technology; C11 - Configuration Management; C12 - Cloud infrastructure and cloud-native; C13 - Artifacts versioning and registry; C14 - Loosely coupled architecture; C15 - Database change management; C16 - Infrastructure as Code; C17 - Emergency response; C18 - Containerization; C19 - Open source software adoption; C20 - Shift left on security; C21 - Transformational leadership; C22 - Trunk-based development; C23 - Monitor systems to inform business decisions; C24 - Westrum organizational culture; C25 - Working in small batches; C26 - Centralized log management; C27 - Lightweight change approval; C28 - Visibility of work in the value stream; C29 - Working in progress limits; C30 - Customer feedback; C31 - Blameless Postmortems; C32 - Data-driven approach for improvements; C33 - Job satisfaction; C34 - Test data management; C35 - Chaos Engineering; C36 - Code maintainability; C37 - Visual management capabilities.

Table 14
Proposed artifact showing categorized DevOps capabilities influencing main metrics.

		Change KPI										Operating KPI									Cultural KPI			Business KPI	
		M02	M03	M04	M06	M09	M10	M12	M15	M17	M19	M01	M05	M07	M08	M11	M13	M20	M21	M22	M18	M42	M59	M14	M16
Cultural	C01	✓	✓	✓	✓	✓	✓					✓	✓	✓		✓					✓	✓	✓	✓	
	C08		✓				✓			✓					✓		✓				✓	✓	✓		
	C19	✓	✓				✓				✓									✓		✓	✓		
	C21	✓	✓	✓			✓					✓	✓								✓	✓	✓	✓	
	C24	✓	✓	✓					✓			✓		✓						✓	✓	✓	✓		
	C31		✓	✓	✓					✓			✓			✓	✓	✓	✓	✓		✓	✓	✓	✓
	C33		✓	✓								✓									✓	✓	✓		
Technical	C02	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓													
	C03	✓	✓	✓	✓		✓		✓			✓								✓	✓				
	C05	✓	✓	✓			✓			✓	✓						✓					✓		✓	✓
	C07	✓	✓								✓	✓					✓			✓		✓	✓		
	C09	✓	✓	✓	✓		✓		✓	✓		✓	✓				✓	✓		✓	✓	✓	✓	✓	
	C11	✓		✓							✓											✓	✓		
	C12	✓	✓		✓		✓					✓	✓		✓	✓			✓	✓		✓			
	C13		✓					✓				✓		✓					✓	✓					
	C14	✓	✓	✓				✓	✓			✓			✓	✓		✓		✓				✓	
	C15		✓	✓		✓			✓			✓	✓				✓		✓			✓		✓	
	C16	✓			✓		✓	✓				✓	✓	✓				✓	✓	✓		✓			
	C18		✓	✓	✓			✓		✓	✓	✓	✓		✓	✓			✓			✓			
	C20	✓	✓	✓		✓						✓	✓	✓	✓			✓	✓			✓	✓	✓	
	C22	✓	✓	✓		✓	✓	✓				✓					✓		✓		✓	✓			
	C26			✓								✓	✓	✓			✓		✓	✓		✓		✓	
	C34		✓	✓						✓										✓					
	C35			✓		✓			✓	✓		✓	✓	✓		✓	✓		✓		✓		✓	✓	
	C36	✓																	✓	✓		✓			
Measurement	C04											✓	✓	✓	✓	✓	✓		✓						✓
	C17											✓	✓	✓	✓	✓		✓	✓	✓				✓	
	C23			✓						✓		✓	✓		✓		✓								
	C29	✓	✓									✓						✓			✓	✓	✓		
	C37	✓	✓	✓	✓		✓					✓				✓		✓	✓			✓			
Process	C06	✓	✓				✓					✓						✓	✓	✓	✓				
	C10	✓	✓																		✓	✓	✓	✓	
	C25	✓	✓		✓			✓								✓	✓	✓	✓	✓	✓			✓	
	C27	✓	✓				✓					✓									✓	✓			
	C28		✓	✓								✓					✓		✓			✓		✓	
	C30			✓		✓						✓					✓							✓	✓
	C32	✓	✓	✓								✓				✓						✓			

*Legend:*M01-M59, C01-C37: Same Capabilities and Metrics already listed in Table 13.

resilience gained from DevOps culture, open-source tooling, and automation.

In interview eight (I08), the practitioner stated that if a team collaborates well (C01), it indicates that they are providing high-quality service, and a team that knows their flow is probably highly performant. In production, this would imply that it would have extremely low numbers for the time it takes to recover (M01) and very low numbers for lead time (M02) or wait time (M22). The interviewee mentioned that his favorite capability was (C31) Blameless Postmortems, a learning capability that should affect positively several metrics that were taken note of. It raised a concern that Mean time to failure (MTTF) (M15) is a metric that is hard to track and should be more related to how long is the build broken for in the cases of deployment. Of the other capabilities that were discussed in this interview, the one that also had more relevancy was (C09) Trust/empower teams to make decisions and changes, which correlated to having a lightweight/streamlining change approval (C27).

It was noted in the ninth Interview (I09) that some organizations are presently focusing on embracing all the new technology, and that seems like a knowledge reset. Cross-team collaboration (C01) and encouraging a learning culture while experimenting (C08) are essential parts of DevOps adoption. Continuous improvement processes and workflows (C06) were also mentioned as essential to answer the need of looking at the work that interrupts from the outside, unplanned work rate (M21) also called toil. Lastly, the open-source mindset (C19) was also mentioned to be vitally connected to important metrics like team happiness and talent retention and that it is vital to communicate changes properly and that is where lightweight change approval (C31) should fit most.

Practician in the tenth videoconference (I10) is a seasoned Ops person who tends to give importance to learning more by experimenting (C08), knowing that experimenting is learning, so when someone else wants to learn something, a server is set up to do this. Other Operational people will also know how to install this and that is a kind of training, which is also experimenting. It is good if the organization can say - “Okay, here is some time for learning, and part of the learning is you’re going to exercise and do something.” - for instance, improving Proactive Monitoring, Observability, and autoscaling (C04). Today, cultural attitude is crucial for team cooperation and communication (C01), and the interviewee mentions achieving exception results in OPS, but that would be something to consider.

In interview eleven (I11) is a Drupal (https://drupal.org) developer that started by mentioning that he had looked at the shared spreadsheet and that he found this to be interesting research since in his experience there are organizations that still struggle with version control systems (C07) and that should evolve. Trusting the team to make decisions and adjustments (C09) is essential. There is a significant difference between trying to avoid performing tasks like Database change management (C15) or recognizing the hazard and attempting to perform them themselves. Continuous process and workflow improvement (C06) have led to, properly planned things, and we are now aiming for a release every three to four weeks. However, we used to have no schedule and would just keep adding items until we thought they were ready, like a security update on Thursday. This impacted Mean Lead-time for Changes (MLT).

The twelfth practitioner started by mentioning that regarding team happiness (M42) and talent retention (M59), his organization gives importance to the level of inclusion and diversity, so much so that the current job offers all mention this fact. On the operational side, the team must focus on production issues as well as upstream issues, as technical debt can dramatically increase the team’s unplanned work (M21). Incentives are given to contributors to improve the process to reduce production errors rate and incidents (M13). On the other hand, having visual management capabilities (C37) helps to reduce Work in Progress (WIP)/Load. Finally, the practitioner also mentioned that the team performance is measured through a survey every 6 months, in which all team members and leadership are invited to participate. This leads to the fact that transformational leadership (C21) has a great impact on the outcome of the Westrum model assessment (M18).

Practitioner (I13) mentioned that in the current organization, there was a focus on key performance indicators (KPIs) and on measuring many things that were brainstormed during meetings, where the team came up with ways to measure it, but having the research artifact in Table 14 would make things easier. It was noted that open-source (C19) is sometimes used for things that are not just software adoption, such as open-source data could be used in Test data management (C34), and so we are looking at how this open-source adoption may be evaluated by these KPIs, as well as how and which ones it influences. Consider working in small batches (C25) so that engineers are in a position where there is time to contribute back to an open-source project. Centralized Log Management (C26) was done on their systems recently, which had a very visible and positive impact.

Interviewee (I14) focused for the most part on Containerization (C18), stating that it helps the testing cycle to go faster and also impacts positively release deployment frequency. In his last organization, when they switched to containers, the deployment went about five times more frequently. This was not only related to switching to containers, but also because of switching to a continuous deployment strategy. It is unknown what capability had more influence there, but the failure rate, i.e. the difficulty or possibility of a deployment succeeding.

The following participant (I15) made an important contribution to this research by focusing on capabilities such as Test Automation and environments (Continuous testing) (C05), Version Control System (C07), Customer/User Feedback (C30), and Data-driven Approach for Improvements (C32), highlighting their relationships to specific metrics and solidly improving the artifact. Manual testing wastes time, slows release frequency and decreases team happiness. So, a practitioner’s company’s challenge is to fully automate those tests and make them part of releasing.

The sixteenth practitioner (I16) has shown a lot of interest in this research, diving first into the Focus on people, processes, and technologies (C10) capability, mentioning evaluation in their company is based on the number of changes proposed by individuals and determining which ones were implemented. In this well-known business strategy (C10), people employ processes and technology to perform tasks for the companies they work for, but organizations might become immersed in new technology at times. Finally, lightweight/streamlining change approval (C27) was mentioned to be associated with the peer review approval process to optimize and expedite the time for commits and releases to be accepted.

The practitioner in interview seventeen (I17) was very proactive in going through the capabilities and metrics. It was mentioned that there is a perspective shift when it comes to practices versus capabilities. In talking about a capability, it is a third-party assessment of something you are looking at from the outside. Where a practice is more internal, we say about doing practice of these things. It was approached in detail that the Artifacts versioning and registry (C13) mentioned having a high impact on several of the metrics pointed out in Table 13. On the other hand, for transformational leadership (C21) to happen there needs to be a coalition of leaders, and they are deliberately cross-functional (C01), to bridge the gap so that actions are aligned with the central goal or premise. Identifying your challenge, creating a strategy around that challenge, communicating that to people, and supporting that culture (C08).

Cross-team collaboration and communication (C01) is a historical challenge in the engineering of many organizations. It is siloed from the rest of the organization and even within the engineering organization. When you have multiple teams working on the same software, sometimes things don’t line up. This is because one team is reliant on another team, and the other team is unaware that there is a deadline for them to release their software. One of the ways we could help solve this is by monitoring systems to inform business decisions (C23), but that will still need to have the cross-collaboration necessary for it to work.

Table 15
Semi-structured interview iterations for evaluation with practitioners' details.

ID	Interview date	Current position	Company	Years of DevOps practice	Age (years)	Country
IT01	2021-06-18	Individual contributor	Acquia	More than 3 years	35–44	Romania
IT02	2021-06-18	Associate DevOps Engineer	Acquia	Less than 1 year	24 =<	India
IT03	2021-06-21	Associate DevOps Engineer	Acquia	Less than 1 year	24 =<	India
IT04	2021-06-21	Associate Engineer	Acquia	Between 1 year and 3 years	24 =<	India
IT05	2021-06-21	Individual contributor	Boxboat	More than 3 years	25–34	United States
IT06	2021-06-22	Architect	Acquia	More than 3 years	25–34	United States
IT07	2021-06-24	Individual contributor	Acquia India Pvt. Ltd.	More than 3 years	25–34	India
IT08	2021-06-24	Individual contributor	Acquia	Between 1 year and 3 years	35–44	United States
IT09	2021-06-29	VP	Acquia	Between 1 year and 3 years	35–44	United States
IT10	2021-07-02	Product Owner, Drupalista	Liip	More than 3 years	35–44	Switzerland

Chaos engineering (C35), for the next practitioner (I19) began when designing a risk management framework for a DevOps pipeline, with the question: how do we reduce infrastructure and security incidents? It involves automation and machine learning so that anomalies are automatically detected, tries to correct them, and brings the system back to normal, thus reducing Mean Time To Recover/Restore (MTTR) to a minimum if done right. It is an approach to keeping the system, stable and consistently normal. Shift left on security (C20) entails including security checks and activities in the pipeline in the development stage. The objective is to guarantee that the codebase is intended to be secure from the start using automation.

During the twentieth interview (I20), the practitioner mostly focused on Configuration Management (C11) and when comparing it to infrastructure as code (C16) the first is acceptable for the application on its own while infrastructure as code is valid for controlling a full platform. Configuration Management has a direct positive impact on Mean Leadtime for Changes (MLT), Change Failure Rate (CFR) and several other metrics seen in Table 13.

The last interview done for the research proposal (I21) touched on several capabilities as well but focused more on loosely coupled architecture (C14), which means that any modification to one service will have no impact on the others. It also allows you to test and verify each service against a standalone environment without requiring an integrated environment. On the other hand, Infrastructure as Code (C16) maintains services available at all times. Defined systems and processes enable continuous testing and publication of minor changes rather than batch updates.

5.1.4. *Data saturation*

In this study, data saturation is observed in order to determine the state of the artifact and prepare it for the evaluation phase, as seen in Table 10. In qualitative research, saturation has gained wide support as a methodological concept, with the purpose of understanding when more data gathering and/or analysis is unnecessary based on the data that has already been gathered and analyzed [42,64,99]. When saturation was reached, five more interviews were still conducted, since there is always some probability that one of the subsequent interviewees or more will disagree, and so there is more certainty of the results obtained.

This is observed at the point that new data tends to be residual when compared to that already collected. In interviews, for instance, when the researcher starts hearing the same remarks again and over, data saturation is achieved [83, p. 372]. Then it is time to stop gathering data and start evaluating what has been gathered [41, p. 26]. Although the percentage values are not zero, there is a clear pattern where the last five contributions are below 1% of the total accessible relations in the matrix. A decrease in the percentage of participant-suggested adjustments relative to the total number of feasible relations is the cause of saturation. As a result, we chose to stop after 21 interviews in order to move on to the next step of the Design Science Research (DSR), which is the evaluation process in Section 6. Table 10 shows the number of contributed relations, the total growth since interview I01 and the percentage of relations added, relative to the total possible relations, where from 37 capabilities times 24 metrics it is possible to have 888, relations. For instance, in the case of I07, where $3.94\% = 100 \times 35/888$. As described in Section 5.1.3, there were cases where there were suggestions for improvements in capabilities, metrics, and categories, and those numbers are also captured in Table 10. For instance, in I01 it was suggested and later confirmed by other practitioners to add a new metric called *talent retention* and to update the list with the existing metric of **team happiness**.

5.1.5. *RQ7—How are DevOps capabilities categorized?*

During the interviews, the alignment of capabilities into categories found in the literature reviews in Section 4 [24,31,40,94,101] as well as their relevance and classification leading to a more refined list was discussed in interviews as displayed in Table 11.

With this resulting table, a matrix for DevOps capabilities and metrics was obtained that includes Cultural, Measurement, Process, and Technical categories to be used in Section 5.2.

5.1.6. *RQ8—How are the main metrics categorized?*

With the objective of categorizing the main metrics Table 12 was elicited. The interview discussion was based on Section 4 (literature review findings). They show Business, Change, Cultural, and Operating types of Key Performance Indicators (KPIs).

Interviewees also mentioned that it is difficult for organizations to track these metrics and capabilities, it requires a lot of effort and expertise to get this data (I03,I09). Thus, if a company switches from traditional systems with already figured out monitoring and metrics, they lose a lot of what was implemented, therefore if the migrations of these metrics are not timely considered, there is a loss of what was experimented with and learned over time, which could have a negative impact on the DevOps adoption.

5.1.7. *RQ9—What DevOps capabilities have a positive impact on which main metrics?*

Given the proposed objective in Section 1 of eliciting metrics that are impacted positively by capabilities. Table 13 shows a summary of these relations. Each column represents a metric and each row represents a capability. The intersections with a check mark symbolize a positive impact in a specific metric by a specific capability. The empty intersections denote no positive relation was found during the research.

This set of relations has significantly contributed to our proposed capability evaluation matrix, which is shown in the next section, accounting also for the categorization of metrics done in Sections 5.1.5 and 5.1.6.

5.2. *Proposed capability evaluation matrix*

The proposed DevOps capability evaluation matrix is an artifact that aims to be used for future strategic planning, keeping in mind the importance to focus first on essential metrics to the business.

Table 16
Relations updated in the artifact during each evaluation iteration.

Evaluation iterations	Change KPI										Operating KPI									Cultural KPI|			Business KPI		Updated		
	M02	M03	M04	M06	M09	M10	M12	M15	M17	M19	M01	M05	M07	M08	M11	M13	M20	M21	M22	M18	M42	M59	M14	M16	Relations	Total	Percentage
IT01				C06				C22	C22	C22		C06			C06				C22				C06		8	340	0.90%
IT02				C24					C24			C24			C24						C06	C06	C24		7	347	0.79%
IT03							C24							C24		C24		C24							4	351	0.45%
IT04								C23				C37	C23		C23				C23				C23		6	357	0.68%
IT05					C09	C28	C09								C09			C09	C28				C02,C03		8	365	0.90%
IT06				C14				C18				C14	C14								C14	C14		C14	7	372	0.79%
IT07						C20	C20	C12								C12	C12		C20	C20				C12	8	380	0.90%
IT08	C08			C08													C08		C27					C21	5	385	0.56%
IT09															C15				C15			C15			3	388	0.34%
IT10									C20				C15			C20									3	391	0.34%

*Legend:*M01-M59, C01-C37: Same Capabilities and Metrics already listed in Table 13.

6. Evaluation

The evaluation of the artifact follows the evaluation process in the design cycle mentioned in Section 3.1, based on the framework for evaluation in design science (FEDS) [117] driven by the work of Pries-Heje, Baskerville and Venable who published several related articles [85,86,116]. Per the authors, the evaluation of an Information System's artifact can be conducted before the production of the proposal, known as the "ex-ante" viewpoint, or after the artifact's construction, known as the "ex-post" perspective. Semi-structured interview iterations are conducted, results are summarized, and a validated artifact is then presented.

6.1. *Semi-structured interviews iterations*

This evaluation involved conducting semi-structured interviews, which is relatively cheaper to evaluate with real users in their real context, and also maintains the goal of having a rigorous evaluation, establishing that utility/benefit will continue in real and long-term situations. This phase was targeted at people with various skills to collect different visions and ensure greater completeness of the validated artifact.

Ten semi-structured interviews were carried out in order to perform several assessment iterations with practitioners listed and are characterized in Table 15, following the outline in Table A.20, with the objective of refining and validating the artifact proposed in Section 5.2. The preparation is similar to what is described in Section 5.1.1

A total of 10 interviews were conducted in this section to evaluate the research proposal. The practitioners interviewed and seen in Table 15 were: seven individual contributors, one architect, one VP and one product owner. The companies they work for are Acquia, Boxboat, and Liip. Five practitioners have more than 3 years of experience in DevOps, while three practitioners have between 1 year and 3 years of experience, and two have less than one year of experience. Of the 21 contributors, 19 are individual contributors with diverse roles, six are Team Leads and the other six are at the management level or above. The majority of the 4 practitioners are between 25 and 34 years of age, three are between 35 and 44 and three have less than 24 years. Their locations are India, Romania, Switzerland and USA.

6.1.1. *Evaluation iterations*

In this section, every practitioner was requested to go through the suggested research artifact while the interview followed the outline in Table A.20. The interviewees were incentivized to suggest recommendations for changing or adding the relationships that they would see fit, as well as double-check the metrics and capabilities categorization.

In interview IT01, it was asked to analyze the artifact and make suggestions for improvements of any missing relations. The practitioner mentioned and justified some extra metrics that are impacted positively by Trunk-based development (C22) like Mean time to failure (MTTF), automated tests pass, Automation Code Coverage, and wait time (Tables 16 and 17). It was also suggested that continuous improvement of processes and workflows (C06) improves customer feedback, keeping Service Level Agreements (SLAs) and Service Level Objectives (SLOs), reliability, and deployment speed. Furthermore, in terms of expenses and environments for *Continuous Integration* (CI), an organization needs the infrastructure to accomplish tests and builds, but the team can probably start by doing it on the desktop or in a virtual environment to save costs. For that, it was stated, that an open-source operating system like GNU/Linux (https://gnu.org) is much better suited. Regarding releases, if things don't improve, by the end of the process an engineer is always caught up in something else and doesn't even have time to enjoy the fact that his work is in production because deployment takes too long. The practitioner concluded that the proposed categorization was sound.

For interview IT02 the same was proposed to the practitioner and

Table 17
Validated artifact with categorized DevOps capabilities influencing main metrics.

		Change KPI										Operating KPI									Cultural KPI			Business KPI	
		M02	M03	M04	M06	M09	M10	M12	M15	M17	M19	M01	M05	M07	M08	M11	M13	M20	M21	M22	M18	M42	M59	M14	M16
Cultural	C01	✓	✓	✓	✓	✓	✓					✓	✓	✓		✓					✓	✓	✓	✓	
	C08	✓	✓		✓		✓			✓					✓		✓	✓			✓	✓	✓		
	C19	✓	✓				✓				✓									✓		✓	✓		
	C21	✓	✓	✓			✓					✓	✓								✓	✓	✓	✓	✓
	C24	✓	✓	✓	✓			✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	
	C31		✓	✓	✓					✓			✓			✓	✓	✓	✓	✓		✓	✓	✓	✓
	C33		✓	✓								✓									✓	✓	✓		
Technical	C02	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓												✓	
	C03	✓	✓	✓	✓		✓		✓			✓								✓	✓			✓	
	C07	✓	✓								✓	✓					✓			✓		✓	✓		
	C09	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	
	C11	✓		✓							✓											✓	✓		
	C12	✓	✓		✓		✓		✓			✓	✓		✓	✓	✓	✓	✓	✓		✓			✓
	C13		✓					✓				✓		✓					✓	✓					
	C14	✓	✓	✓	✓			✓	✓			✓	✓	✓	✓	✓		✓		✓		✓	✓	✓	✓
	C15		✓	✓		✓			✓			✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	
	C16	✓			✓		✓	✓				✓	✓	✓				✓	✓	✓		✓			
	C18		✓	✓	✓			✓	✓	✓	✓	✓	✓		✓	✓			✓			✓			
	C20	✓	✓	✓		✓	✓	✓		✓		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	
	C22	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓					✓		✓	✓	✓	✓			
	C26			✓								✓	✓	✓			✓		✓	✓		✓		✓	
	C34		✓	✓						✓										✓					
	C35			✓		✓			✓	✓		✓	✓	✓		✓	✓		✓		✓		✓	✓	
	C36	✓																	✓	✓		✓			
Measurement	C04											✓	✓	✓	✓	✓	✓		✓						✓
	C17											✓	✓	✓	✓	✓		✓	✓	✓				✓	
	C23			✓					✓	✓		✓	✓	✓	✓	✓	✓			✓				✓	
	C29	✓	✓									✓						✓			✓	✓	✓		
	C37	✓	✓	✓	✓		✓					✓	✓			✓		✓	✓			✓			
Process	C06	✓	✓		✓		✓					✓	✓			✓		✓	✓	✓	✓	✓	✓	✓	
	C10	✓	✓																		✓	✓	✓	✓	
	C25	✓	✓		✓			✓								✓	✓	✓	✓	✓	✓			✓	
	C27	✓	✓				✓					✓								✓	✓	✓			
	C28		✓	✓			✓					✓					✓		✓	✓		✓		✓	
	C30			✓		✓						✓					✓							✓	✓
	C32	✓	✓	✓								✓				✓						✓			

Table 18
Capability categories, weight on KPI categories.

	Change KPIs	Operating KPIs	Cultural KPIs	Business KPIs	Total AVG
Cultural capabilities	45.71%	39.68%	90.48%	42.86%	54.68%
Technical capabilities	49.44%	61.90%	46.30%	33.33%	47.74%
Measurement capabilities	20.00%	64.44%	26.67%	30.00%	35.28%
Process capabilities	30.00%	33.33%	52.38%	42.86%	39.64%

Performance Oriented Culture (C24) was identified as highly important since there will be a lot of confidence between the team and the leadership, having a positive impact on deployment speed, automated tests pass, pipeline success rate, reliability, Service Level Objectives (SLOs) and customer feedback. Also mentioned as related to Continuous Improvement (C06), influencing cultural KPIs like team Happiness and talent retention. Businesses with a high-performance culture may consistently achieve high levels of performance and outcomes. Therefore, many companies place an effort on cultivating a high-performance culture, since it may be the difference between mediocrity and growth, efficiency and falling behind.

During interview IT03 the DevOps engineer focused on examining performance organizational culture (C24) capability, impacting change volume, application performance, production errors rate, and Unplanned Work Rate (UWR) and at the same time mentioning that the most creative companies and high-performing organizations are continuously seeking to improve and never consider themselves done with their development or transformational journey. Several other interesting aspects were mentioned like when there is technical depth closed that means, whatever the technical issues there are closed, organizational culture is the enabler for that change. An important lesson learned is that growth requires a careful mix of challenge and nurture. Too much challenge, repeated too frequently, it's overwhelming and can lead to burnout. Too little challenge prevents us from growing and does not improve the needed outcomes.

The interview IT04 Centralized Dashboards for Monitoring systems to inform business decisions (C23) associated with mostly operating KPIs and customer feedback, which is influencing visual management capabilities (C37) was the main focus of discussion and settled on how this affects operational mindset, especially on reliability. The characteristics that we consider a metric to be an Operational KPI and the factors that we consider a metric to be a Change KPI were also discussed. Dashboards improve decision-making and may streamline the entire decision-making process, saving time and decreasing paperwork. It improves participation and cooperation by giving everyone first-hand access to data and allowing them to witness the real-time outcomes of their efforts.

The practitioner in interview IT05 examined how empowering the team to make decisions and changes (C09) is affecting a few more change and operating KPIs and definitely improves all the cultural KPIs and customer feedback. Employees will only strengthen their decision-making skills via experience and learning, especially if the firm is experiencing rapid development. Things may not go as planned the first time, or even the second, which is where leading and mentoring are fundamental. Regardless of industry, business size, or location, high-performance cultures share several common traits. Transformational leadership and empowered teams are common examples. Continuous

Table A.19
Questions used in the research proposal, adapted from Mäkinen et al. [60].

ID	Related RQs	Type	Question
1			**Background Information**
1.1	–	Closed-ended	How large is your organization?
1.2	–	Open-ended	How do you see your current DevOps adoption?
1.3	–	Closed-ended	What is your team size currently?
2			**Methods**
2.1	–	Open-ended	What software engineering methods or capabilities are you using?
2.2	RQ1	Open-ended	What capabilities of DevOps are you using and why?
2.3	RQ4	Open-ended	What metrics should be tracked in the DevOps process and why?
3			**Categorization**
3.1	RQ7	Open-ended	How are DevOps capabilities categorized?
3.2	RQ8	Open-ended	How are the main metrics categorized?
4			**Software Life Cycle Impact**
4.1	–	Open-ended	What aspects most impact your day-to-day work?
4.2	RQ9	Open-ended	What DevOps capabilities have a positive impact on which main metrics?
4.3	RQ9	Closed-ended	Do you see any relations that are missing or incorrect in the shown table?
5			**DevOps Capabilities Challenges and Benefits**
5.1	–	Open-ended	Any other DevOps capabilities challenges or benefits you would like to mention?
6			**DevOps Metrics Challenges and Benefits**
6.1	–	Open-ended	Any other DevOps metrics challenges or benefits you would like to mention?

Table A.20
Interview iterations topics used in evaluating the proposed artifact, adapted from Mäkinen et al. [60].

ID	Related RQs	Type	Question
1			**Background Information**
1.1	–	Closed-ended	How large is your organization?
1.2	–	Open-ended	How do you see your current DevOps adoption?
1.3	–	Closed-ended	What is your team size currently?
2			**Categorization evaluation**
2.1	RQ7	Closed-ended	Do you agree with the DevOps capabilities categorization shown?
2.2	RQ7	Open-ended	If not how would you categorize capabilities?
2.3	RQ8	Closed-ended	Do you agree with the DevOps metrics categorization shown?
2.4	RQ8	Open-ended	If not how would you categorize capabilities?
3			**Capabilities and Metrics evaluation**
3.1	–	Open-ended	Do you see value in using this evaluation matrix and why?
3.2	RQ1	Open-ended	Do you agree or disagree with any of these DevOps capabilities and why?
3.3	RQ4	Open-ended	Do you agree or disagree with any of these DevOps metrics and why?
4			**Impact based evaluation**
4.1	RQ9	Closed-ended	Do you disagree with any of the relations shown in the table and why?
4.2	RQ9	Open-ended	What more DevOps capabilities have a positive impact in which main metrics?
4.3	–	Open-ended	What would be your expected results from applying these capabilities and metrics in your organization?

learning and experimentation, as well as an openness to change, are essential. Another approach was the visibility of work in the value stream (C28), a process capability that also improves Cycle Time Value (CTV) and "wait time" metrics. Finally, customer feedback has a positive impact on C02 *Continuous Integration* (CI) and C03 *Continuous Delivery or Deployment* (CD).

The engineering architect in interview IT06 is an expert on microservices and distributed systems and so contributed more to examining Loosely Coupled Architecture and Microservices (C14), where he argued the importance of adding deployment duration, reliability, and a few other operating and cultural KPIs like team happiness and talent retention as well as customer traffic. Loose coupling is a design pattern that hides implementation details inside each microservice in this capacity. When components are loosely connected, systems have several advantages for the organization. Containerization (C18) was also briefly discussed, with the result of adding deployment speed to Table 17 and confirming others in the list of metrics impacted by this capability. Containerization is also a big benefit in continuous integration *Continuous Integration* (CI) and continuous delivery *Continuous Delivery or Deployment* (CD): containers greatly simplify integration testing and standardize CI/CD through Docker images.

Examination in interview IT07 focused on Cloud Infrastructure and Cloud Native (C12) validating already existing or new change KPIs relations like Mean time to failure (MTTF) and also operating KPIs production errors rate and incidents, Work in Progress (WIP), and customer usage. Moreover, traditional business apps, which are generally run in an on-premises data center, require an entirely different design than cloud-native applications. Security should be shifted to the left and reduce risks to a minimum. Taking this strategy lowers the chances of suffering a data breach or security issue. Shift Left on security (C20) impacts all cultural KPIs and some change KPIs like defect escape rate, Cycle Time Value (CTV), and deployment size, also impacting operating KPIs wait time and organizational culture metrics. Finally, the team happiness cultural KPI was also confirmed to be impacted by the existing relations in the matrix. Some indications of employee dissatisfaction, like low talent retention, are easy to identify.

In interview IT08, three capabilities were assessed in depth. The first was support learning culture and experimentation (C08) where change KPIs Mean Lead-time for Changes (MLT), deployment speed, and work in progress were explained to have a strong impact while practicing this capability. The second, Transformational leadership (C21) was also confirmed to have an influence on customer feedback due to the attention that these leaders are putting into all the life cycle processes to improve it. Lastly, Lightweight change approval (C27) was mentioned to be missing an obvious impact relationship in wait time operating KPI. It also stated the importance of visionary inspirational communication, personal recognition, and intellectual stimulation by supportive leadership while adopting DevOps capabilities.

In interview IT09, the most evaluated capability was Database Change management (C15). For operating KPIs there were missing relations on Service Level Agreements (SLAs) / Service Level Objectives (SLOs), wait time, and cultural KPI talent retention. Database change management is one of a set of capabilities that helps organizations produce better software and perform better. It should be possible to track database changes and those should be implemented and reconciled as quickly as feasible, while still protecting data privacy inside the database context. In the end, talent retention cultural KPI was also stated that by personal experience, when C15 improves there is not only team happiness rising but also fewer people leaving.

In the last evaluation interview IT10 the previous Database Change management (C15) relationships were confirmed also adding Mean Time To Detection (MTTD) and in Shift Left on security (C20) adding related KPIs pipeline success rate and production error rate. Managing database changes was stated to be a difficult process. It is much more challenging when just getting started on a project and the data model is continuously changing. There was also a valuable discussion about

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# This code fetches google search results in a systematic form for MLR
# Date: February 2021
# Requirements: `pip install requests bs4 pandas`

from requests import get
from bs4 import BeautifulSoup
import pandas
import time

def csv_dump(results, name):
    print(results, name)
    df = None
    df = pandas.DataFrame(results)
    df.index += 1
    df.to_csv(name + '.csv')

def parse_results(raw_html):
    soup = BeautifulSoup(raw_html, 'html.parser')
    result_block = soup.find_all('div', attrs={'class': 'g'})
    for result in result_block:
        link = result.find('a', href=True)
        title = result.find('h3')
        if link and title:
            yield { 'URL': link['href'], 'Title': link.text.strip() }

def google_search(query,max_results=500,num_results=100,lang="en"):
    usr_agent ={'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:10.0) \
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36'}
    escaped_query = query.replace(' ', '+')
    results = []
    for start in range(0,max_results,num_results):
        google_url = 'https://www.google.com/search?q={}&num={}&start={}&hl={}'.format(
            escaped_query, num_results+1, start, lang)
        print(google_url)
        time.sleep(3)
        response = get(google_url, headers=usr_agent)
        response.raise_for_status()
        results += parse_results(response.text)
    return results

def get_results(query, name):
    print(query, name)
    results = google_search(query)
    print(results)
    csv_dump(results, name)

if __name__ == '__main__':
    get_results(
        'devops AND (practices OR capabilities)',
        'capabilities')
    get_results(
        'devops AND (metrics OR measures OR kpi OR indicator)',
        'metrics')
```

Listing 1. Python code for consistent fetching of a large number of Google search results.

capabilities being multifaceted and dynamic, allowing different areas of the company to tailor their approach to improvement and focus on the skills that would bring them the most value.

6.1.2. Evaluation results

In this section, the key results of the evaluation done with ten interview iterations are summarized. Primarily, Table 16 shows that the additions or updates to the matrix have become residual, which confirms the data saturation already observed in Section 5.1.4. A summary of all updated relations in each iteration can be found in Table 16, together with the total relations and the relative percentage of each change. There were residual relations suggested between capabilities and metrics, but there were no suggestions for changing capabilities, metrics, or categories, therefore not expressed in the table.

As can be seen, the changes to the artifact are now modest in contrast to the previous phase, indicating that the evaluation was successful and supports the proposal's design [99].

6.2. Validated artifact

Using the validated here artifact presented, business executives and organization leaders will be able to evaluate the various DevOps capabilities in relation to the outcomes, identifying the capabilities that require the most significant long-term enhancements and highlighting the capabilities that should become the primary focus of future IT investments as seen in Table 17.

As a few participants mentioned, this will also allow the organization to stimulate discussions about how each capability impacts DevOps adoption, allowing them to figure out where they can get the most value from. Like what capabilities, if included from the start, would have the most influence on a variety of KPIs. This capability matrix is outcome-based, focusing on important outcomes (KPIs) and how capabilities promote change in those outcomes. This gives clear guidance and strategy on high-level goals (with an emphasis on capabilities to enhance key outcomes) to technical leadership. It also allows team leaders and individual contributors to define progress targets for the current period based on the capabilities their team is working on.

7. Conclusion

In this section, we conclude the research done with communication, general conclusions, limitations, and future work.

7.1. Communication

Out of the research already done, the two MLRs have been submitted for approval and publication in two different journals with Q1 rank. The MLR on DevOps capabilities was already published. The MLR on DevOps metrics was also submitted and awaits publisher reviews.

7.2. Research conclusions

This study has made important contributions to both academia and industry on the DevOps topic. In summary, Design Science research was done based on two MLRs on DevOps capabilities, in DevOps metrics, and 21 semi-structured interviews in the build phase, where 207 papers were identified as relevant to these study topics. In order to evaluate the research proposal, 10 semi-structured interviews were done, resulting in a validated Capability Evaluation Matrix.

- This study proposes a consensus definition distinguishing capabilities from practices, based on academic and industry literature review.
- A thorough investigation was conducted in order to find a consensus definition of DevOps metrics across academics and practitioners.
- It was investigated and exposed where the capabilities or practices are mentioned in the literature (RQ2) and what are their differences seen (RQ3).
- The purpose of each metric is identified in detail (RQ5) and the reason why each metric is important is analyzed (RQ6).
- From all the literature review done in this research, 37 DevOps capabilities (RQ1) and 24 validated DevOps metrics (RQ4) were identified.
- DevOps capabilities have been researched, explained, and after a careful evaluation categorized (RQ7). The same rigorous work was conducted for DevOps metrics (RQ8).
- In order to develop a core study proposal, this investigation drafted the DevOps capabilities that have a beneficial influence on each of the key metrics (RQ9).
- The major outcome of this research is a Capability Evaluation Matrix presented in Section 6.2, which has been proposed, debated, and validated by DevOps practitioners and experts.

A set of interesting conclusions arise when identifying the analysis vector that connects the capability categories to the KPI categories based on the validated artifact in Table 17. Table 18 shows mainly the most evident conclusion is that cultural capabilities have a strong impact on improving the cultural KPIs and the highest overall impact.

Nevertheless, operating KPIs are mostly impacted by technical and measurement capabilities. However, process capabilities are the second most impactful on cultural KPIs, and when looking at what are the main drivers of change, those are technical, cultural, and process capabilities. Therefore, organizations should prioritize not only technology, but also cultural and process improvements. A DevOps capability is defined as the ability to perform a DevOps practice or by the quality, or state, of being capable. These capabilities are, dynamic and have been growing and changing over the years, are defined by the ability of an organization to perform DevOps practices, or by the quality, or state of being capable. The two capabilities with the maximum relation to metrics are, in order, C09-Empower teams to make decisions and changes and C24-Performance organizational culture. A DevOps metric is defined as a quantifiable, business-relevant, trustworthy, actionable, and traceable indicator that aids organizations in making data-driven decisions to continuously improve their software delivery process. The five top metrics with most of the relations in the model, that an organization should start by measuring are, in order, M03-Deployment Frequency (DF), M01-Mean Time To Recover/Restore (MTTR), M42-Team happiness, M02-Mean Lead-time for Changes (MLT) and M04-Change Failure Rate (CFR). Metrics have been extended to 24, divided into four KPI categories: change, operating, cultural, and business. Finally, it was also perceived that releasing software with both speed and stability is achievable if the company is continuously monitoring the appropriate metrics and improving the right capabilities by focusing on the outcomes, rather than just following a prescribed path for each team.

7.3. Limitations

Identified limitations of this study include the fact that it is based on MLR. Therefore, a part of the material has not gone through the critical peer-review process that academic research is typically exposed to. To mitigate the impact of this danger, it was chosen to design the review procedure using the recommendations given by Garousi et al. [38] and to conduct each step using this method. Semi-structured interviews used in this research usually require a large enough variety sample to yield precision and a variety of opinions. To address this problem, 31 interviews were conducted in total, considered a good number [64,65] for this kind of qualitative research. The use of search terms and search engines may result in an inadequate selection of primary materials. Formal searches were conducted using particular keywords, and specific source code was used to decrease the chance of missing all relevant studies and increase the dependability of repeating this study. For the year 2021, this research was limited to only three months. Although the year 2021 is present, it is irrelevant for the extraction of the data because the three-month analysis is insignificant. Lastly, the inclusion of English-only publications, which may exclude relevant research in other languages, was a limitation.

7.4. Future work

The results and findings revealed in this study will assist to feed new research so that future studies can evaluate if certain metrics are still common and should be explored further researching the possible ways to put these capabilities, metrics, and relations into practice. The relationship matrix used in the artifact may also be expanded to indicate whether the influence or impact on the metric can be verified as positive or negative. Based on this, it should be possible to produce a heat map for a capability model that would be easier to use in management decisions. At the organizational level, we still don't know which measures are already being utilized by which industries. Section 5.1.6 states that interviewees acknowledge difficulties for organizations to track metrics and capabilities, which might be an interesting opportunity for future work since the problem is outside the scope of this study. What other organizational factors have the most influence on each of the main, important metrics, and if we can influence these factors, or how? Is it possible to expose the metrics, capabilities, and influencing factors in information systems in order to support management decisions [58]? In contrast, there is still a debate going on [14,26,29,91] regarding: Should all of these indicators be tracked regularly? Which metrics are capable of being monitored automatically? Which metrics can only be monitored using surveys? All are interesting questions to investigate. Lastly, as demonstrated in Section 4, better monitoring of the software delivery process is extremely significant and sought. DevOps metrics should try to measure efficiently the right aspects in order to determine whether DevOps is effective.

Author biographical note

The author was born in Lisbon, Portugal in 1974. He has a Masters Degree in Management Information Systems, a Computer Science Engineering Degree at UAb in Lisbon, and in 1998 he received a B.S. Degree in Arts at Institute Polytechnic of Lisbon.

In 2018, he co-authored the book Seeking SRE, Publisher: O'Reilly Media. Chpt 18 09/2018 ISBN: 9781491978856.

In May 2011 the author founded the Portuguese Drupal Association, dedicated to manage and support the open source software community and from 2015 until 2021 he has been the DevOps and Infrastructure track chair for Drupalcon Europe. Also, he has been a frequently invited speaker at Agile, Devops and Software conferences in the US and Europe.

Since 2011 he has been working for Acquia Inc. www.acquia.com where he manages the DevOps and SRE teams, implementing industry best practices and information systems targeted to support decisions. Before that, from 2006 to 2011 he was Internet Director at Ocasião Lda, a media newspaper in Lisbon and from 2001 to 2005 he was I.T. systems manager at EGEAC and Lisbon City hall public company.

CRediT authorship contribution statement

Ricardo Amaro: Conceptualization, Methodology, Writing – original draft. **Rúben Pereira:** Data curation, Investigation, Supervision, Writing – review & editing. **Miguel Mira da Silva:** Supervision, Validation, Writing – review & editing.

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.

Appendix A. Interview outline

Appendix B. Source code

References

[1] S. Abdelkebir, Y. Maleh, M. Belaissaoui, An agile framework for ITS management in organizations. Proceedings of the 2nd International Conference on Computing and Wireless Communication Systems - ICCWCS'17, ACM Press, New York, New York, USA, 2017, p. 8, https://doi.org/10.1145/3167486.3167556.

[2] W.C. Adams, Conducting semi-structured interviews. Handbook of Practical Program Evaluation: Fourth Edition, John Wiley and Sons, Inc., Hoboken, NJ, USA, 2015, pp. 492–505, https://doi.org/10.1002/9781119171386.ch19.

[3] J. Angara, S. Gutta, S. Prasad, Devops with continuous testing architecture and its metrics model. Advances in Intelligent Systems and Computing vol. 709, Springer Verlag, K.L. University, Vijayawada, AP, India, 2018, pp. 271–281, https://doi.org/10.1007/978-981-10-8633-5_28.

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

[5] R.L. Baskerville, M. Kaul, V.C. Storey, Genres of inquiry in design-science research: justification and evaluation of knowledge production, MIS Q. 39 (3) (2015) 541–564, https://doi.org/10.25300/MISQ/2015/39.3.02.

[6] K. Behr, G. Kim, G. Spafford, The Visible Ops Handbook: Implementing ITIL in Four Practical and Auditable Steps, IT Process Institute (ITPI), Eugene, OR, 2008.

[7] A. Bertolino, G.D. Angelis, A. Guerriero, B. Miranda, R. Pietrantuono, S. Russo, DevOpRET: continuous reliability testing in DevOps, J. Softw. n/a (n/a) (2020) e2298, https://doi.org/10.1002/smr.2298.

[8] B. Beyer, C. Jones, J. Petoff, N.R. Murphy, Site Reliability Engineering: How Google Runs Production Systems, O'Reilly Media, Inc., 2016.Google, https://landing.google.com/sre/sre-book/toc/

[9] D.N. Blank-Edelman, Seeking SRE: Conversations About Running Production Systems at Scale, O'Reilly Media, Inc., USA, 2018.

[10] Cambridge, CAPABILITY | meaning in the Cambridge English Dictionary, 2021a, https://dictionary.cambridge.org/dictionary/english/capability.

[11] Cambridge, PRACTICE | meaning in the Cambridge English Dictionary, 2021b, https://dictionary.cambridge.org/dictionary/english/practice.

[12] D. Cohen, B. Crabtree, Semi-structured Interviews, 2006, http://www.qualres.org/HomeSemi-3629.html.

[13] C.M. Conway, K. Roulston, Conducting and analyzing individual interviews. The Oxford Handbook of Qualitative Research in American Music Education, 2014, https://doi.org/10.1093/oxfordhb/9780199844272.013.014.

[14] Cprime, DevOps Metrics to Monitor Software Development - Cprime, 2021, https://www.cprime.com/resources/blog/devops-metrics-to-monitor-software-development/.

[15] R. Cyber, DevOps KPIs to Measure Success, 2019, https://www.royalcyber.com/blog/devops/devops-kpis-to-measure-success/.

[16] E. DeBoer, Sonatype, Accelerate: A Principle-based DevOps Framework, 2019, https://blog.sonatype.com/principle-based-devops-frameworks-accelerate.

[17] T. Development, A Guide To Measuring DevOps Success and Proving ROI, 2020, https://www.tiempodev.com/blog/measuring-devops/.

[18] Devopedia, DevOps Metrics, 2019, https://devopedia.org/devops-metrics.

[19] DevOps Research and Assessment (DORA), State of DevOps 2019 - DORA. Technical Report, DORA, 2019.https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

[20] J. Díaz, J.E. Pérez, M.A. Lopez-Peña, G.A. Mena, A. Yagüe, Self-service cybersecurity monitoring as enabler for DevSecops, IEEE Access 7 (1) (2019) 100283–100295, https://doi.org/10.1109/ACCESS.2019.2930000.

[21] Z. Ding, J. Chen, W. Shang, Towards the use of the readily available tests from the release pipeline as performance tests: are we there yet. 42nd ACM/IEEE International Conference on Software Engineering, ICSE 2020, IEEE Computer Society, New York, NY, USA, 2020, pp. 1435–1446, https://doi.org/10.1145/3377811.3380351.

[22] D. Dingley, 4 Key Metrics for DevOps Success Video, 2019, https://www.veracitysolutions.com/4-key-metrics-for-devops-success.

[23] B. Dobran, 15 DevOps Metrics and Key Performance Indicators (KPIs) To Track, 2019, https://phoenixnap.com/blog/devops-metrics-kpis.

[24] DORA, DORA research program, 2020, https://www.devops-research.com/research.html.

[25] A. Dyck, R. Penners, H. Lichter, Towards definitions for release engineering and DevOps, in: B. Adams, S. Bellomo, C. Bird, F. Khomh, K. Moir (Eds.), RELENG@ICSE, IEEE Computer Society, 2015, p. 3.http://dblp.uni-trier.de/db/conf/icse/releng2015.html#DyckPL15

[26] R. Edwards, Research highlights challenges of Salesforce DevOps in 2020, 2021, https://www.enterprisetimes.co.uk/2021/02/16/research-highlights-challenges-of-salesforce-devops-in-2020/.

[27] F.M.A. Erich, C. Amrit, M. Daneva, A qualitative study of devops usage in practice, J. Softw. 29 (6) (2017) e1885, https://doi.org/10.1002/smr.1885.

[28] V. Fedak, DevOps metrics: what to track, how and why do it, 2020, https://medium.com/@FedakV/devops-metrics-what-to-track-how-and-why-do-it-e08dc6864eab.

[29] Flosum, Keys to Improve Salesforce DevOps Efficiency - Flosum - Continuous Integration, release management, 2021, https://flosum.com/keys-to-improve-salesforce-devops-efficiency/.

[30] N. Forsgren, J. Humble, The role of continuous delivery in it and organizational performance. SSRN Electronic Journal, 2015, p. 15, https://doi.org/10.2139/ssrn.2681909.

[31] 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.https://itrevolution.com/accelerate-book/

[32] N. Forsgren, J. Humble, G. Kim, A. Brown, N. Kersten, Accelerate state of DevOps 2018 strategies for a new economy, Rep. DevOps Res. Assess. (DORA) 1 (2018) 78.

[33] N. Forsgren, M. Rothenberger, J. Humble, J. Thatcher, D. Smith, A taxonomy of software delivery performance profiles: investigating the effects of devops practices. 26th Americas Conference on Information Systems, AMCIS 2020 vol. 1, 2020, p. 5.

[34] B.B.N. de França, H. Jeronimo, G.H. Travassos, B.B. Nicolau de França, H. Jeronimo, G.H. Travassos, Characterizing DevOps by hearing multiple voices, in: E, S. DeAlmeida (Ed.), Proceedings of the 30th Brazilian Symposium on Software Engineering, Unicesumar; Colivre; Espweb; Tasa Eventos, New York, NY, USA, 2016, pp. 53–62, https://doi.org/10.1145/2973839.2973845.

[35] A.M. Fred, C. Cook, 6 proven metrics for DevOps success | TechBeacon, 2021, https://techbeacon.com/devops/6-proven-metrics-devops-success.

[36] P. Gallagher, Tracking Success in DevOps Pipelines, 2020, https://blog.goodelearning.com/subject-areas/devops/how-to-measure-success-in-devops/.
[37] V. Garousi, M. Felderer, M.V. Mäntylä, The need for multivocal literature reviews in software engineering. Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering - EASE '16, ACM Press, New York, New York, USA, 2016, p. 6, https://doi.org/10.1145/2915970.2916008.
[38] V. Garousi, M. Felderer, M.V. Mäntylä, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Inf. Softw. Technol. 106 (September 2018) (2019) 101–121, https://doi.org/10.1016/j.infsof.2018.09.006.
[39] J. Gelo, DevOps Metrics Matter: Why, Which Ones, and How - HCL SW Blogs, 2020, https://blog.hcltechsw.com/accelerate/devops-metrics-matter-why-which-ones-and-how-2/.
[40] Google, DevOps capabilities | DORA - Google Cloud, 2020, https://cloud.google.com/solutions/devops/capabilities.
[41] M.P. Grady, Qualitative and Action Research: A Practitioner Handbook, Phi Delta Kappa International, 1998.
[42] G. Guest, E. Namey, M. Chen, A simple method to assess and report thematic saturation in qualitative research, PLoS One 15 (5) (2020) e0232076, https://doi.org/10.1371/journal.pone.0232076.
[43] T. Hall, DevOps metrics | Atlassian, 2016, https://www.atlassian.com/devops/frameworks/devops-metrics.
[44] A.R. Hevner, A three cycle view of design science research, Scand. J. Inf. Syst. 19 (2) (2007) 87–92.http://aisel.aisnet.org/sjis/vol19/iss2/4
[45] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (1) (2004) 75–105, https://doi.org/10.2307/25148625.
[46] R. Honig, Bridge the DevOps Development Chasm to Boost DevOps KPIs - Ozcode, 2020, https://oz-code.com/blog/devops/bridging-the-devops-development-observability-chasm-to-boost-devops-kpis.
[47] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Addison-Wesley Professional, USA, 2010.
[48] J. Humble, B. O'Reilly, Lean Enterprise: How High Performance Organizations Innovate at Scale, O'Reilly Media, Inc., USA, 2014.
[49] M.M.A. Ibrahim, S.M. Syed-Mohamad, M.H. Husin, M.M. Ahmad Ibrahim, Managing quality assurance challenges of DevOps through analytics. Proceedings of the 2019 8th International Conference on Software and Computer Applications vol. Part F1479, Association for Computing Machinery, New York, NY, USA, 2019, pp. 194–198, https://doi.org/10.1145/3316615.3316670.
[50] V. Kamani, 7 crucial DevOps metrics that you need to track, 2019, https://hub.packtpub.com/7-crucial-devops-metrics-that-you-need-to-track/.
[51] A. Kankanhalli, B.C. Tan, K.K. Wei, Contributing knowledge to electronic knowledge repositories: an empirical investigation, MIS Q. 29 (1) (2005) 113–143, https://doi.org/10.2307/25148670.
[52] J. Kernel, DevOps Metrics: 7 KPIs to Evaluate Your Team's Maturity, 2020, https://www.xplg.com/devops-metrics-7-kpis/.
[53] 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
[54] B. Kitchenham, Procedures for Performing Systematic Reviews vol. 33, Keele University, Keele, UK, 2004, p. 26.
[55] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report, Ver. 2.3 EBSE Technical Report. EBSE, 2007.
[56] Kris Buytaert, Help, My Datacenter is on fire, 2021, https://www.slideshare.net/KrisBuytaert/help-my-datacenter-is-on-fire.
[57] C.J. Kuiper. Relationship of Transformational Leadership and Organizational Change During Enterprise Agile and DevOps Initiatives In Financial Service Firms, Liberty University, School of Business, 2019. Ph.D. thesis.
[58] J.P.L..K.C. Laudon, Management Information Systems: Managing the Digital Firm, Global Edition, Pearson Education, USA, 2017.
[59] E. Lock, Measure DevOps Metrics That Matter, 2020, https://www.devopsdigest.com/measure-devops-metrics-that-matter.
[60] S. Mäkinen, M. Leppänen, T. Kilamo, A.-L. Mattila, E. Laukkanen, M. Pagels, T. Männistö, Improving the delivery cycle: a multiple-case study of the toolchains in Finnish software intensive enterprises, Inf. Softw. Technol. 80 (2016) 175–194, https://doi.org/10.1016/j.infsof.2016.09.001.
[61] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, H. Shen, L. Chen, K. Lu, Preliminary findings about DevSecOps from grey literature. 20th IEEE International Conference on Software Quality, Reliability, and Security, QRS 2020, Institute of Electrical and Electronics Engineers Inc., Nanjing University, State Key Laboratory for Novel Software Technology, Nanjing, China, 2020, pp. 450–457, https://doi.org/10.1109/QRS51102.2020.00064.
[62] S.T. March, G.F. Smith, Design and natural science research on information technology, Decis. Support Syst. 15 (4) (1995) 251–266, https://doi.org/10.1016/0167-9236(94)00041-2.
[63] D. Marijan, S. Sen, Devops enhancement with continuous test optimization. 30th International Conference on Software Engineering and Knowledge Engineering, SEKE 2018 vol. 2018-July, Knowledge Systems Institute Graduate School, Simula, Norway, 2018, pp. 536–541, https://doi.org/10.18293/SEKE2018-168.
[64] B. Marshall, P. Cardon, A. Poddar, R. Fontenot, Does sample size matter in qualitative research?: a review of qualitative interviews in is research, J. Comput. Inf. Syst. 54 (1) (2013) 11–22, https://doi.org/10.1080/08874417.2013.11645667.
[65] M.J. McIntosh, J.M. Morse, Situating and constructing diversity in semi-structured interviews, Glob. Qual. Nurs. Res. 2 (2015), https://doi.org/10.1177/2333393615597674.
[66] Merriam-Webster, Capability | Definition of Capability by Merriam-Webster, 2021a, https://www.merriam-webster.com/dictionary/capability.
[67] Merriam-Webster, Practice | Definition of Practice by Merriam-Webster, 2021b, https://www.merriam-webster.com/dictionary/practice.
[68] K. Mikko, DevOps capability assessment in a software development team, Science 135 (2018) 408–415.
[69] A. Mishra, Z. Otaiwi, Devops and software quality: a systematic mapping, Comput. Sci. Rev. 38 (1) (2020) 14, https://doi.org/10.1016/j.cosrev.2020.100308.
[70] J. Mitlohner, S. Neumaier, J. Umbrich, A. Polleres, Characteristics of open data CSV files. 2016 2nd International Conference on Open and Big Data (OBD), IEEE, Vienna, 2016, pp. 72–79, https://doi.org/10.1109/OBD.2016.18.
[71] S.I. Mohamed, DevOps maturity calculator DOMC -value oriented approach, Int. J. Eng. Res. Sci. 2 (2) (2016) 2395–6992.
[72] F.J. Moraes, DevOps KPI in Practice — Chapter 1 — Deployment Speed, Frequency and Failure, 2018, https://medium.com/@fabiojose/devops-kpi-in-practice-chapter-1-deployment-speed-frequency-and-failure-2fd0a9303249.
[73] A.D. Nagarajan, S.J. Overbeek, A DevOps implementation framework for large agile-based financial organizations, in: H.A. Proper, R. Meersman, C.A. Ardagna, H. Panetto, C. Debruyne, D. Roman (Eds.), Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, vol. 11229, Springer Verlag, Department of Information and Computing Sciences, Utrecht University, Utrecht, Netherlands, 2018, pp. 172–188, https://doi.org/10.1007/978-3-030-02610-3_10.
[74] T. Nero, DevOps Metrics : 15 KPIs that Boost Results and RoI - Cuelogic Technologies Pvt. Ltd, 2021, https://www.cuelogic.com/blog/devops-metrics.
[75] A. Novak, Six Core Capabilities of a DevOps Practice – The New Stack, 2014, https://thenewstack.io/six-core-capabilities-of-a-devops-practice/.
[76] R.T. Ogawa, B. Malen, Towards rigor in reviews of multivocal literatures: applying the exploratory case study method, Rev. Educ. Res. 61 (3) (1991) 265–286, https://doi.org/10.3102/00346543061003265.
[77] L. Olsina, P. Becker, D. Peppino, G. Tebes, Specifying the process model for systematic reviews: an augmented proposal, J. Softw. Eng. Res. Dev. 7 (February 2020) (2019) 7, https://doi.org/10.5753/jserd.2019.460.
[78] K. Peffers, T. Tuunanen, C.E. Gengler, M. Tuunanen, W. Hui, V. Virtanen, J. Bragge, The design science research process: a model for producing and presenting information systems research. First International Conference on Design Science Research in Information Systems and Technology, 2006.83–16
[79] R. Pereira, J. Serrano, A review of methods used on IT maturity models development: a systematic literature review and a critical analysis, J. Inf. Technol. 35 (2) (2020) 161–178, https://doi.org/10.1177/0268396219886874.
[80] J.F. Pérez, W. Wang, G. Casale, Towards a DevOps approach for software quality engineering. WOSP-C 2015 - Proceedings of the 2015 ACM/SPEC Workshop on Challenges in Performance Methods for Software Development, in Conjunction with ICPE 2015, ACM Press, New York, New York, USA, 2015, pp. 5–10, https://doi.org/10.1145/2693561.2693564.
[81] Plutora, DORA DevOps Metrics - Accelerate your Value Stream - Plutora.com, 2020, https://www.plutora.com/resources/videos/devops-dora-metrics.
[82] W. Pourmajidi, A. Miranskyy, J. Steinbacher, T. Erwin, D. Godwin, Dogfooding: using IBM cloud services to monitor IBM cloud infrastructure. CASCON 2019 Proceedings - Conference of the Centre for Advanced Studies on Collaborative Research - Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering, 2020, pp. 344–353.http://arxiv.org/abs/1907.06094
[83] T. Power, D. Jackson, B. Carter, R. Weaver, Misunderstood as mothers: women's stories of being hospitalized for illness in the postpartum period, J. Adv. Nurs. 71 (2) (2015) 370–380, https://doi.org/10.1111/jan.12515.
[84] L. Prates, J. Faustino, M. Silva, R. Pereira, DevSecOps metrics, in: S. Wrycza, J. Maślankowski (Eds.), Lecture Notes in Business Information Processing vol. 359, ISCTE-IUL, Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal, 2019, pp. 77–90, https://doi.org/10.1007/978-3-030-29608-7_7.
[85] J. Pries-Heje, R. Baskerville, The design theory nexus, MIS Q. 32 (4) (2008) 731–755, https://doi.org/10.2307/25148870.
[86] J. Pries-Heje, R. Baskerville, J. Venable, Strategies for design science research evaluation. 16th European Conference on Information Systems, ECIS 2008, 2008.
[87] Puppet Labs, 2013 State of DevOps Report. Technical Report, Puppet Labs, 2013. http://puppetlabs.com/2013-devops-report
[88] Puppet Labs, 2014 State of DevOps Report. Technical Report, Puppet Labs, 2014. http://puppetlabs.com/2014-devops-report
[89] Puppet Labs, 2015 State of DevOps Report. Technical Report, Puppet Labs, 2015. http://puppetlabs.com/2015-devops-report
[90] Puppet Labs, 2016 State of DevOps Report. Technical Report, Puppet Labs, 2016. https://puppetlabs.com/solutions/devops/
[91] Puppet Labs, 2020 State of DevOps Report. Technical Report, Puppet Labs, 2020. https://puppet.com/resources/report/2020-state-of-devops-report/
[92] K. Raworth, C. Sweetman, S. Narayan, J. Rowlands, A. Hopkins, Conducting Semi-Structured Interviews, Oxfam, 2012.
[93] ReleaseTEAM, DevOps Metrics Measure Your DevOps Results, 2021, https://www.releaseteam.com/measure-your-devops-results/.
[94] I.T. Revolution, Itrevolution, 24 Key Capabilities to Drive Improvement in Software Delivery - IT Revolution, 2020, https://itrevolution.com/24-key-capabilities-to-drive-improvement-in-software-delivery/.

[95] J. Riggins, Google's Formula for Elite DevOps Performance – The New Stack, 2020, https://thenewstack.io/googles-formula-for-elite-devops-performance/.
[96] 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. M. 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, https://doi.org/10.1016/bs.adcom.2018.03.014.
[97] M. Rother, Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results, MGH, New York, USA, 2019.
[98] M. Sánchez-Gordón, R. Colomo-Palacios, A multivocal literature review on the use of DevOps for e-learning systems. Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality, TEEM'18, ACM, Association for Computing Machinery, New York, NY, USA, 2018, pp. 883–888, https://doi.org/10.1145/3284179.3284328.
[99] B. Saunders, J. Sim, T. Kingstone, S. Baker, J. Waterfield, B. Bartlam, H. Burroughs, C. Jinks, Saturation in qualitative research: exploring its conceptualization and operationalization, Qual. Quant. 52 (4) (2018) 1893–1907, https://doi.org/10.1007/s11135-017-0574-8.
[100] A. Schurr, Mobile App DevOps Metrics that Matter - NowSecure, 2019, https://www.nowsecure.com/blog/2019/02/27/mobile-app-devops-metrics-that-matter/.
[101] M. Senapathi, J. Buchan, H. Osman, DevOps capabilities, practices, and challenges: insights from a case study. Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 - EASE'18, number June in EASE'18, ACM, Association for Computing Machinery, New York, USA, 2018, pp. 57–67, https://doi.org/10.1145/3210459.3210465.
[102] S. Sharma, The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise, John Wiley & Sons, Inc., Indianapolis, Indiana, 2017, https://doi.org/10.1002/9781119310778.
[103] R. Shinde, Is your DevOps successful?, 2019, https://www.accenture.com/us-en/blogs/software-engineering-blog/reshma-shinde-devops-success-metrics.
[104] G. Singh, Measuring DevOps Success with DevOps Metrics, 2019, https://www.xenonstack.com/blog/devops-metrics/.
[105] J. Smeds, K. Nybom, I. Porres, DevOps: a definition and perceived adoption impediments. Lecture Notes in Business Information Processing vol. 212, Springer, USA, 2015, pp. 166–177, https://doi.org/10.1007/978-3-319-18612-2_14.
[106] B. Snyder, B. Curtis, Using analytics to guide improvement during an agile-DevOps transformation, IEEE Softw. 35 (1) (2017) 78–83, https://doi.org/10.1109/MS.2017.4541032.
[107] C. Sonnenberg, J. vom Brocke, Evaluation patterns for design science research artefacts, in: M. Helfert, B. Donnellan (Eds.), Practical Aspects of Design Science Chapter 7, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 71–83, https://doi.org/10.1007/978-3-642-33681-2_7.
[108] C. Staff, Essential Quantitative DevOps Metrics - Coveros, 2016, https://www.coveros.com/essential-quantitative-devops-metrics/.
[109] D. Sun, M. Fu, L. Zhu, G. Li, Q. Lu, Non-Intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: a case study in AWS, IEEE Trans. Emerg. Top. Comput. 4 (2) (2016) 278–289, https://doi.org/10.1109/TETC.2016.2520883.
[110] D.A. Tamburri, D. Di Nucci, L. Di Giacomo, F. Palomba, Omniscient DevOps analytics, in: J.M. Bruel, M. Mazzara, B. Meyer (Eds.), Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, DevOps 2018 vol. 11350, Springer International Publishing, 2019, pp. 48–59, https://doi.org/10.1007/978-3-030-06019-0_4.
[111] F.T.C. Tan, S.L. Pan, M. Zuo, Realising platform operational agility through information technology–enabled capabilities: aresource-interdependence perspective, Inf. Syst. J. 29 (3) (2019) 582–608, https://doi.org/10.1111/isj.12221.
[112] D. Teixeira, R. Pereira, T. Henriques, M.M.D. Silva, J. Faustino, A maturity model for DevOps, Int. J. Agile Syst. Manag. 13 (4) (2020) 464, https://doi.org/10.1504/IJASM.2020.112343.
[113] D. Teixeira, R. Pereira, T.A. Henriques, M. Silva, J. Faustino, A systematic literature review on DevOps capabilities and areas, Int. J. Hum. Cap. Inf. Technol. Prof. 11 (2) (2020) 22, https://doi.org/10.4018/IJHCITP.2020040101.
[114] TestEnvironmentManagement.com, Top 5 DevOps Metrics – Test Environment Management, 2019, https://www.testenvironmentmanagement.com/top-5-devops-metrics/.
[115] Ubiq, Top DevOps Metrics and KPIs To Monitor Regularly - Ubiq BI Blog, 2020, http://ubiq.co/analytics-blog/top-devops-metrics-kpis-to-monitor-regularly/.
[116] J. Venable, J. Pries-Heje, R. Baskerville, A comprehensive framework for evaluation in design science research. Proceedings of the 7th International Conference on Design Science Research in Information Systems: Advances in Theory and Practice, 2012, pp. 423–438, https://doi.org/10.1007/978-3-642-29863-9_31.
[117] J. Venable, J. Pries-Heje, R. Baskerville, FEDS: a framework for evaluation in design science research, Eur. J. Inf. Syst. 25 (1) (2016) 77–89, https://doi.org/10.1057/ejis.2014.36.
[118] Veritis, Measuring DevOps: Key 'Metrics' and 'KPIs' That Drive Success!, 2020, https://www.veritis.com/blog/measuring-devops-key-metrics-and-kpis-that-drive-success/.
[119] Y. Wang, M. Mäntylä, S. Demeyer, K. Wiklund, S. Eldh, T. Kairi, Software test automation maturity: a survey of the state of the practice, in: M. Vansinderen, H. G. Fill, L. Maciaszek (Eds.), Proceedings of the 15th International Conference on Software Technologies, No. 15th International Conference on Software Technologies (ICSOFT), SCITEPRESS - Science and Technology Publications, 2020, pp. 27–38, https://doi.org/10.5220/0009766800270038.
[120] M. Watson, 15 Metrics for DevOps Success, 2017, https://stackify.com/15-metrics-for-devops-success/.
[121] S. Watts, DevOps: Metrics and Key Performance Indicators (KPIs), 2017, https://itchronicles.com/devops/devops-metrics-kpis/.
[122] R. Westrum, A typology of organisational cultures, Qual. Saf. Health Care 13 (Suppl 2) (2004) 22–27, https://doi.org/10.1136/qshc.2003.009522.
[123] J. Wettinger, U. Breitenbücher, F. Leymann, DevOpSlang – bridging the gap between development and operations. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNCS, vol. 8745, IFIP, Stuttgart, 2014, pp. 108–122, https://doi.org/10.1007/978-3-662-44879-3_8.

CHAPTER 5

Article #4

This paper (A4) researches the DevOps adoption benefits and synthesizes them, following two SLRs conducted over 98 publications on the topic, together with 36 case studies. In this process, key benefits are identified, such as faster time to market, improved collaboration, and increased automation, and using empirical evidence from case studies to confirm the found benefits [27]. While the first SLR elicits all of the benefits found in the literature, the second SLR compares these benefits to practical data from the case studies' DevOps deployment. The most commonly claimed benefits include faster time to market, enhanced collaboration, and increased automation. The study confirms that empirical evidence exists for all identified benefits. The study contributes to the theoretical body of knowledge by consolidating the benefits of DevOps implementations.

The main findings are consolidating the benefits of DevOps implementation1 through two systematic literature reviews. The most reported benefits include better collaboration between developers and operators, faster time to market, and improvements in synergy and automation. The findings also show that the most reported benefits are better collaboration between developers and operators, faster time to market, and improvements in synergy and automation. Finally, it provides a comprehensive synthesis of DevOps benefits and a discussion on the challenges, helping new practitioners understand what to expect when adopting DevOps.

Article details:

- Title: DevOps benefits: A systematic literature review
- Date: May 2022
- Journal: Software - Practice and Experience
- Scimago Journal Rank: Q2
- Publisher: John Wiley and Sons Ltd

Check for updates

Received: 13 October 2021 | Revised: 9 February 2022 | Accepted: 8 April 2022

DOI: 10.1002/spe.3096

RESEARCH ARTICLE

WILEY

DevOps benefits: A systematic literature review

João Faustino[1] | Daniel Adriano[1] | Ricardo Amaro[1] | Rubén Pereira[1] | Miguel Mira da Silva[2]

[1]Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal

[2]Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

Correspondence
João Faustino, Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal.
Email: jpcfo11@iscte-iul.pt

Abstract

Among current IT work cultures, DevOps stands out as one of the most adopted worldwide. The focus of this culture is on bridging the gap between development and operations teams, enabling collaborative effort toward quickly producing software, without sacrificing its quality and support. DevOps is used to tackle a variety of issues; as such, there are differing benefits reported by authors when performing their analysis. For this research, we aim to reach consensus on the DevOps benefits reported in existing literature. To accomplish this objective, two systematic literature reviews. The first intends to find all benefits reported in the literature, while the second review will be used to map the benefits found in the first one with DevOps implementation case studies, providing empirical evidences of each benefit. To strengthen the results, the concept-centric approach is used. During this research it was possible to observe that the most reported benefits are aligned with the DevOps premises of better collaboration between developers and operators, delivering software and products quicker. Based on DevOps implementation case studies, most reported benefits include a faster time to market as well as improvements in synergy and automation. Less reported benefits include a reduction in failed changes and security issues.

KEYWORDS

agile, benefits, case study, DevOps, systematic literature review

1 | INTRODUCTION

With recent technological advances, information technology (IT) departments have taken an increasingly strategic role in organizations[1] given the importance of IT in helping the accomplishment of business objectives.[2] Several disciplines like IT governance and IT service management (ITSM) have built mechanisms and processes so that both IT and business can be aligned in terms of aims and expectations, helping organizations satisfy their objectives.[3–5]

Just as organizations' strategic view has evolved, software development lifecycles (SDLC) have also matured to satisfy current demands. To face the great changes observed in the modern-day markets, businesses need to have greater speed and flexibility. This translates to challenges for IT departments worldwide.[6] As stated, the SDLC has been evolving, no longer strictly focusing on the performance of its own processes, as seen on traditional software development methodologies like waterfall,[7,8] but on the iterations and relationships between the intervenient on the SDLC process and the value that the software can bring to the business.[9] These kinds of software development methodologies are considered agile methodologies and follow the "Agile Manifesto."[10] Even though Business and IT development are brought closer, a

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

gap is still observed within the IT department's development and operations teams.[11] The major issue between these two teams are the different objectives for each team, IT development team is focused on delivering new features or products, while the IT operators are focused on the stability.[12] It's believed by introducing changes on systems would lead to instability.[13] however due to the constant market changes, IT of the organizations need to evolve into new functionalities.[14] A DevOps culture has emerged to address this gap. The DevOps word itself comes from the junction of two other words: development and operations.[15]

DevOps has been adopted across the globe and new research articles flourish. Several studies have reported practices, benefits, and challenges however not always in a structured, clear, and concise way.[16–18] In literature, one can find studies that synthesize DevOps practices, as for example, deployment automation[19–21]; however, it presently lacks research that specifically synthesize its benefits and challenges, guiding professionals in what they may expect during and after DevOps practice implementation.[22] Lack of willingness to share can be a challenge to DevOps implementation.[23]

Being a contemporary topic, with both theoretical and empirical studies found in literature, this research aims to synthesize the benefits organizations may expect with DevOps implementation and how to achieve them. Being said, by synthetizing the DevOps benefits, this research also provides which problems organizations faced before the DevOps adoption and what was the benefit achieved after that. This will help organizations to know what problems could be fixed by the DevOps adoption. The adopted research methodology will include two systematic literature reviews (SLR).

2 | RESEARCH BACKGROUND

The term DevOps started to be researched after Patrick Debois introduced it at a conference entitled "Agile Infrastructure and Operations" in 2008.[24] A DevOps culture aims to bridge the gap between IT development and IT operations, who support applications after they are delivered to production.[25] The focus of DevOps is on improving communication, collaboration, and synergy of IT teams,[26,27] enabling the continuous development and enhancement of applications to meet both market changes and the dynamic needs of the business.[28,29]

In order to achieve said objectives, DevOps builds a foundation in the following areas: culture, automation, lean, measurement and sharing.[6] By looking at Wiedemann et al.'s[6] work, one may note that of the perspectives presented above, *people* play an important role for culture and sharing. Willingness to share is needed, allowing for colleagues and team members to learn and improve their knowledge and experiences. On automation and measurement, one can state that technological tools are the main factor; tools that are used to improve performance, automating what is being done manually, removing the element of human error, and be used to measure tasks and find improvements.[30] Process optimization is a focal point for lean methodologies. They are used in DevOps to identify opportunities for process enhancement, leveraging feedback loops between a its main actors. In later studies,[31] people, technology and processes are considered the cornerstones of DevOps.

Since 2001, organizations have adopted agile methodologies for its SDLC[32] where the most implemented methodologies are XP and SCRUM.[33] These methodologies are the foundation of DevOps and DevOps can be seen as its extension, since they are based on the same principles of introducing short release cycles and to develop forward the customer or user feedback.[34] However, DevOps includes the operations team on early stages of the software development, being able to develop the software already with the operations team input, thus developing software more stable including the business feedback.[35] Also, DevOps stands out due to the collection of techniques and tool to enable software continuous delivery, clearing the path of the software to production.[36,37]

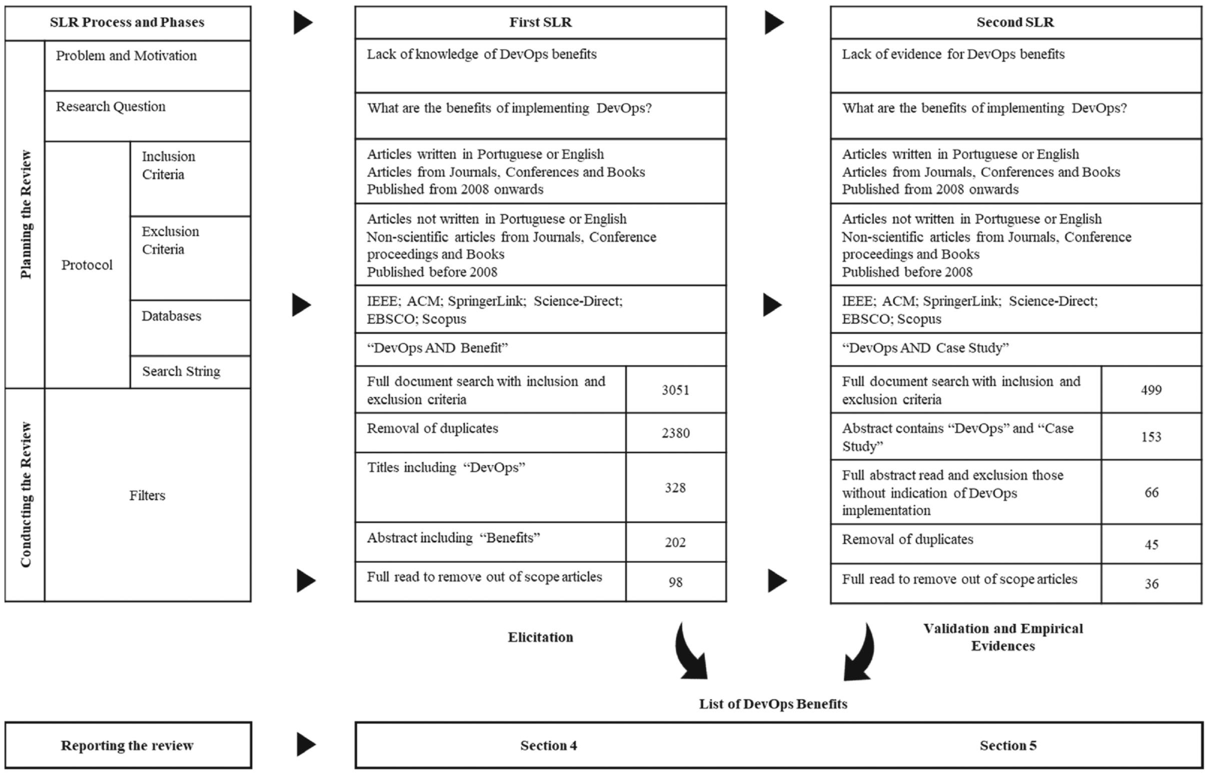
In conclusion, a DevOps culture seems to be very attractive to organizations worldwide, being based in a "*The faster you fail, the faster you recover*" philosophy[14(p1)], enabling a culture of experimentation to release new products, services and software, allowing the organization to grow and to satisfy their customers.[38]

3 | RESEARCH METHODOLOGY

To achieve this research goal, the authors have chosen the systematic literature review (SLR) methodology. It is seen as one of the most widely used research methods to collect and synthesize evidence.[39] SLR's are meant to have a well-defined process to identify, evaluate, and interpret all the evidence collected during research.[40,41] Thus, for this investigation, the authors have followed the framework proposed by Kitchenham[40] where the SLR is split into three stages (Figure 1). Moreover, to add rigour to this research, the authors have chosen to perform two SLR's: the first to find all the benefits

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License



SLR Process and Phases			First SLR		Second SLR	
Planning the Review	Problem and Motivation		Lack of knowledge of DevOps benefits		Lack of evidence for DevOps benefits	
	Research Question		What are the benefits of implementing DevOps?		What are the benefits of implementing DevOps?	
	Protocol	Inclusion Criteria	Articles written in Portuguese or English Articles from Journals, Conferences and Books Published from 2008 onwards		Articles written in Portuguese or English Articles from Journals, Conferences and Books Published from 2008 onwards	
		Exclusion Criteria	Articles not written in Portuguese or English Non-scientific articles from Journals, Conference proceedings and Books Published before 2008		Articles not written in Portuguese or English Non-scientific articles from Journals, Conference proceedings and Books Published before 2008	
		Databases	IEEE; ACM; SpringerLink; Science-Direct; EBSCO; Scopus		IEEE; ACM; SpringerLink; Science-Direct; EBSCO; Scopus	
		Search String	"DevOps AND Benefit"		"DevOps AND Case Study"	
Conducting the Review	Filters		Full document search with inclusion and exclusion criteria	3051	Full document search with inclusion and exclusion criteria	499
			Removal of duplicates	2380	Abstract contains "DevOps" and "Case Study"	153
			Titles including "DevOps"	328	Full abstract read and exclusion those without indication of DevOps implementation	66
			Abstract including "Benefits"	202	Removal of duplicates	45
			Full read to remove out of scope articles	98	Full read to remove out of scope articles	36
Reporting the review			Section 4		Section 5	

FIGURE 1 Steps performed in each of the performed SLRs

described on existing literature, while the second one being used to find instances of those benefits being reported on case studies from DevOps implementation. This second SLR will confirm and evaluate the findings from the first SLR, where all the DevOps benefits were gathered from literature. The authors believe that searching for case studies is a reliable method of evaluation given that these are a research methodology known by providing evidence of a certain phenomenon.[42] The first SLR was conducted between May and September 2020, while the second SLR was carried out between August and October 2020.

The process designed by Kitchenham[40] was followed by both SLR's. The authors have started by defining the Problem and Motivation for the review. For the first SLR where the expected result was to identify the reported benefits on the literature, the motivation was to acknowledge the DevOps benefits on the literature, while for the second SLR the motivation was to find evidence of the DevOps benefits. The next step of the process was to define the research question (RQ) for the review. In this case, the same RQ was identified for both SLRs "What are the benefits of implementing DevOps?"

After the RQ definition, the next step is to define a protocol where inclusion and exclusion criteria was defined, along with the search databases and the search string of each SLR. The inclusion and exclusion criteria were based on the language of the publications, scientific publications, and publication date. Regarding inclusion and exclusion criteria (IEC) a minimum date was set, considering that the DevOps concept was born after the aforementioned "Agile Infrastructure and Operations" conference, in 2008.[24] For the databases, the authors have used some of the most known and used databases on the scientific community. All these criteria were the same for both SLRs except the search string. For the first SLR the search string was focused on DevOps benefits while for the second SLR the search string was focused on DevOps case studies.

After applying these criteria, some filters were added to the review to exclude publications that wouldn't provide the necessary information for this research. One example of the used filters was the removal of duplicated. All this process definition can be seen with more detail in Figure 1 for both SLRs.

To evaluate the research subject's trend, the researchers have analyzed the date of publication of each relevant piece of literature from a chronological point of view. This is extremely helpful to prove that the research topic has a corresponding trend and is largely demanded by the market. The researchers used the concept-centric approach[43] to better synthesize and analyze the concepts elicited from the literature. This helps to understand the focus of the review, for a better understanding of the readers. Also, it helps the researchers to structure the review. The usage of the concept-centric approach can be seen in Section 4 where the concepts identified are the benefits found per reference, while in Section 5 one can see the case studies identified per reference.

4 | FIRST SLR: LIST OF BENEFITS

After performing the first SLR and analyzing the articles, the list of DevOps benefits was elicited and can be seen in Table 1. The concept-centric approach taken by the researchers can be found in Appendix A, where it is possible to see the match between the concepts and the authors that have identified those concepts in literature.

In the next section, one can see a discussion and some conclusions about the benefits found on the literature, regarding Table 1. After the full read of the publications, the authors have identified the benefits described on the publications and grouped those benefits by the concepts, also seen on Table 1.

Several authors among literature claim to see an improvement on the communication and collaboration (as seen in Appendix A) between developers and operators,[22,28,44,45] creating a synergetic environment where both teams desire to work together toward accomplishing overall objectives.[11,46]

Before DevOps, operators and developers may have had different mindsets when facing change. With the disappearing of the waterfall software development methodology and the emergence of the “Agile Manifesto”,[10,47] the developer’s mindset shifted to deliver more features as fast as possible to production, while the operator’s continued to have the mindset of guaranteeing the stability of the systems it was solely responsible for.[12] These divergent views on change typically lead to finger-pointing, with operators blaming developers for the production impact of deployments when they might have been involved earlier in the development process to try to anticipate possible problems before they reached production.[48]

TABLE 1 List of benefits identified in literature

Concept ID	Benefits	# References
B01	Cross team collaboration and communication	49
B02	Faster time to market	41
B03	Faster and better feedback loops	38
B04	Increase of code quality	32
B05	Increase of value	26
B06	Improvement of system reliability	22
B07	Less mean time to recover	17
B08	Increase of team performance	17
B09	Costs reduction	13
B10	Processes and tools standardization	13
B11	Maximization of competences	13
B12	Decrease of manual work	11
B13	Increase of customer satisfaction	11
B14	Less failed changes	11
B15	Increase of employees motivation	9
B16	More innovation	8
B17	Better deployment management	5
B18	Less security issues	5
B19	Organizational cultural changes	41

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Because of the resultant DevOps synergy, both operators and developers are more driven to collaborate across teams. They will feel that they are working toward a common and greater goal for everyone.[49] However, this can also be extended to the business. Just as Agile practices and principles brought business and developers together,[10] DevOps introduces operators into the mix, emphasizing the significance of operations management in the organization.[50]

As seen in Appendix A, faster time to market, related to continuous integration and continuous delivery capabilities, is one of the most reported benefits from DevOps. Organizations can design new, better features for their products as a result of faster feedback.[51] Through DevOps enabled automation they are then able to put said features into the market at a quicker rate than competition.[52]

It is noteworthy to mention that various authors were able to identify the different sources that contribute for a better and faster feedback under DevOps: customers and end users (business users)[14] as well as between the DevOps team itself.[49] Customers and end users are those who use the application; they are best to identify issues and potential improvements.[53,54] DevOps has a practice to shorten the feedback loops between operators and developers, which also leads to faster feedback when something is going wrong and requires further work.[55]

Improved feedback does not only contribute for better development and application stability, but also leads to opportunities for DevOps teams to learn about its components (for example, operators can learn about the development process, and developers learn about processes which guide operators work) as well as the business itself.[56]

Code quality can be increased as a result of implementing improved delivery pipelines under which code is built into packages and introduced to the respective repository after each check.[57] During the packaging of a new build, code can be submitted through quality gates, ensuring that best practices defined for that application are being adhered to.[20] Due to the continuous integration capability encouraged by DevOps culture, developers from several teams will be working in collaboration with other developers. There will be opportunities to find issues or needed improvements to other developers' code, improving the overall code quality of applications.[58]

There is great consensus in literature about the increase of value when using DevOps. DevOps is a culture that uses lean and agile practices. DevOps phenomenon arose as an extension of agile software development inspired by lean concepts.[59] The first Agile Manifesto principle is about value: "*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*"[10] Due to the continuous delivery capability that DevOps employs, shorter development and release cycles[60] can be achieved, where business and customers will notice the ongoing improvement of software, realizing the continuous increase in value of their applications.[29]

Automation brings an additional benefit in the ability to perform defined actions after an event is triggered by automated monitoring.[61] By automating infrastructure using infrastructure as code, the availability and reliability of applications will be improved. Such infrastructure can scale up and scale down according to its reported usage and demand.[14]

Related to the faster time to market benefit, feedback and automation is not only used to deliver new or improved products.[62] Due to the premises of DevOps in having a solid IT team, both developers and operations work together to guarantee that fixes are deployed in production instantly.[63] This contributes to the stability and availability of applications, so that defects do not cause greater impact.[23]

The objective of implementing any framework, practice or methodology is to improve performance. However, operators and developers commonly have different objectives and use different metrics to measure their performance, as explained in Section 1. Thus, to improve a team's performance, an alignment is needed for the definition of clear and visible goals.[64] In case of operations, these would be aimed at the stability and reliability of an application, while for developers the focus should be on the features delivered for it.[44,65] Since DevOps is also targeted toward using lean and agile practices,[59] it concentrates its aim on improving people, processes and technologies capabilities, specifically in the way the work in process is limited and done in small batches, therefore contributing to the well-being of their teams.[29]

Cost reduction is among the top goals of every organization in the world. As discussed before, DevOps can help reduce costs by reducing bottlenecks in the SDLC, optimizing time to deploy changes in production and enabling better resource management.[22] This can help balance software quality with costs, helping organizations to have an increased return on investment.[66,67]

To optimize the SDLC it is essential that operations can react quickly, helping developers have their environments stable, up and running. DevOps encourages operators to use the infrastructure as a code capability in order to help manage and configure environments more quickly and in a standardized way.[68,69] This allows developers to have development and preproduction environments, which aids in the discovery of issues early in the SDLC.[70] Likewise, the environment process configuration and tools used by each team should be standardized, avoiding common situations like "it was working on my machine."[71]

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

With the mixing of IT teams by making developers and operators work together, the competences of these resources will be increased.[25] Developers will be able to get abilities that most often regard to operations, while operators will get abilities on areas of development.[65] This also contributes for improved knowledge management,[22] allowing a DevOps team to be more complete in terms of their joint competences.

On a decrease of manual work a consensus can be concluded in literature. This is accomplished by using automation. There are three major areas where it can be applied: testing,[72] the delivery pipeline,[73] and configuration or provisioning.[74] Test scripts can be automated by using tools that will perform the actions of the testers, verifying if the final output is the desired one. Thus, this capability reduces the manual work of testers as well as the risk for human error. Moreover, automated tests enable continuous testing capability which helps find integration issues earlier in the development cycle, making defect resolution faster and with less impact on production environments.[72] This also frees up the tester to create other, exploratory testing activities.

Operations are usually not only responsible to guarantee the stability of production environments but also of the lower environments. If a development team requires several development environments, each requiring operators to configure manually, a blocking of development resources may occur. DevOps encourages the usage of infrastructure-as-code to allow the operators to manage their infrastructure and environment configurations by using code, replicating said configurations for several alternative environments, speeding up configuration.[70] Furthermore, it is possible to automatically provision environments with resources based on predetermined thresholds, guaranteeing their stability and availability.[67]

DevOps encourages developers to continuously integrate their code so that issues can be found earlier.[75] However, this requires a lot of work if every time a developer checks-in his code, a manual package needs creation for other developers to review. Under DevOps, every time that a developer commits code to a code repository, a script is triggered that will automatically test and create a package or artifact, checking and giving immediate feedback if there is any error and, if successful, storing it properly.[56] From this point onwards, the developed package can be used for installation across all environments. With the package stored, one could also trigger a script that will deploy the package with new code in a test environment, making it available for testers; alternatively, once the deployment is completed, more complex automated testing can be triggered, like integration or end-to-end tests and developers informed if the new code failed in any test, speeding up the bug fixing and increasing the software's stability.[22,76]

Customer satisfaction can be seen as a consequence, resulting from a variety of previously described benefits. Since DevOps will continuously improve the stability of applications while reaching for customer feedback, customer satisfaction will increase.[22] By reducing bottlenecks on the SDLC process, the customers' feedback is deployed on the application faster, further increasing satisfaction.[77] Also, looking from a perspective in which a customer is internal, DevOps can also contribute to cost reduction.

Less failed changes can be seen as a consequence from both the standardization of processes and tools, as well as from other DevOps capabilities in general. With a standardization of processes, like those used in a deployment, for example, issues on a deployment script can be found and fixed in other environment, before reaching a production deployment.[22] With all the automation (in testing areas, for example) and continuous integration that DevOps encourages, help is obtained toward identifying issues with development work earlier on the SDLC, helping to avoid failed changes when moving to production.[78]

Employees of an organization will feel more motivated by working on a more communicative environment, in which they feel that their team will back them up.[12] This will contribute to reduced blame-games between developers and operators.[54] Due to the sharing culture that DevOps promotes, developers will learn about operators' tasks just as much as operators will learn about developers' tasks. Thus, employees will be more capable to backup each other up on different types of work.[15,79]

Due to the increased speed of development, and by enabling a faster time to market, DevOps allows organizations to experiment new solutions, features and products[29] without incurring in significant economic impacts. Start-up companies are known for creating new market segments due to the innovative solutions they create. DevOps brings a great opportunity for these organizations, which does not have much revenue, allowing a spirit of the "*Faster you fail, faster you recover*"[14(p1)].

The setup of IT Teams before DevOps were structured in a way that deployments were manually performed by single or multiple operation teams that had the responsibility for configuring and setting up production environments, database configuration, backups of software build and reversing bad builds on the new software.[56] This raises the possibility and concern of human errors, which can impact the entire service stack of an organization.[69] Automation is one of the most used capabilities in DevOps which can help on this matter. By building automatic deployment mechanisms it is possible to decrease the volume of potential outages from applications.[65] Moreover, DevOps gives the ability for developers to

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

perform their own deployments under the motto “You build it, you run it”,[73] which empowers developers to find bad builds before operators, resulting in improved deployment management.

DevOps promotes monitoring during the entire deployment pipeline, using tools to notify developers and operators in case of something going wrong, or the need for manual actions, like rolling back the software to a previous version,[63] contributing also to a better deployment management.

DevOps is usually allied with cloud implementations which help deploy security integration and carry out penetration tests between applications.[80,81] Nowadays, cloud providers offer services that promote the usage of DevOps, in which a security model for their customers is ensured.[82]

As discussed earlier, DevOps is not only focused on automating processes and improved performance, but also on cross team collaboration and interaction between people. For DevOps, or other agile software development methodologies, organizations need to have a culture that allows for these interactions. Lean, Agile, and DevOps appeared in various times to meet various requirements,[83] but they all concentrate on organizational culture by forming interdisciplinary teams, cutting waste, concentrating on the customer, embracing change, and providing value on a continual basis. Under DevOps, sharing is the key for operators and developers to work together. As such, organizational culture needs to be adapted to promote this kind of involvement.[76]

5 | SECOND SLR: EMPIRICAL EVIDENCES OF DEVOPS BENEFITS

A second SLR was carried out to confirm and evaluate the findings from the first SLR, in which all DevOps benefits were gathered from literature. To do so, the authors captured and analyzed a total of 36 DevOps implementation case studies. Each of the studies was read for data on the outcomes of introducing DevOps capabilities in a business environment. A list of these articles, their references, and basic vectors of analysis, are found on Table 2.

Due to the data provided in Table 2, it was possible to produce Figure 2 with a segregation of the case studies by continent, country, and business sector. It is possible to see that DevOps is more present in Europe or on multinational organizations that work in several countries from multiple continents. Regarding the business sector, the IT business sector clearly stands out from the other sectors. Since DevOps is a culture that is focused on IT developers and operators, it makes sense that IT organizations implement this culture before other sectors. However, from professional experience from the authors, the DevOps culture have been expanding on the financial sector (banking and insurance).

Having identified and analyzed the final list of DevOps implementation articles, we proceeded to map business benefit concept IDs to case studies in which they are mentioned. Some of the documents included findings from more than one case study; for these, we relied on decimals to differentiate implementation results from each organization as much as we possibly could. However, it is important to note that some authors merged in a single body the observations and results of multiple, different DevOps case studies, making full differentiation impossible. In total, 69 case studies were identified and reviewed as part of our research. The results of this effort are presented in Table 3 (also refer to Appendix B). Moreover, one of the case studies didn’t presented any benefit, where the authors have identified to study the benefits as their own future work. The benefit ID and benefit description columns are referring to the concepts previously presented in Table 1. Lastly, it is relevant to add that most of the case studies did not provide quantitative evidence of these benefits, but often referred to them in a qualitative manner.

An improvement in the rate by which new development is produced, deployed, and reaches the market was by a considerable margin the most widely and explicitly observed benefit of a DevOps adoption. The implementation of DevOps practices, particularly when it comes to establishing a bridge between development and operations teams,[91] was commonly pointed out as an enabling factor toward faster delivery.[108] The added flexibility associated with DevOps practices allows for new software evolutions to be implemented faster, while sustaining a quality standard.[75] Shorter response times[53] and increased deployment speed[104] are likely to be observed in a successful DevOps integration. In Luz et al.’s[28] study it is stated that “after the DevOps adoption, it was possible to make 29 deployments on a single day” whereas before, due to rigid and conflicting policies at the operational level, deployment were only scheduled to occur once, weekly.

As established, the development of synergies between teams is a foundational principle for applying DevOps practices. From our research, improved collaboration and communication between developers and operational staff was a frequently reported benefit resulting from DevOps implementation. Increased awareness of the overall software development processes, standard deployment practices and service management took place[69] as teams abandoned traditional

TABLE 2 List of DevOps implementation case studies analyzed

ID	Reference	Country	Continent	Business sector
CS.01	84	Sweden	Europe	Information technology
CS.02	85	Spain	Europe	Human resources
CS.03	86	Italy	Europe	Lighting business
CS.04	56	Denmark	Europe	Information technology
CS.05	28	Brazil	South America	Government organization
CS.06	64	Morocco	Africa	Information technology
CS.07	87	Montenegro	Europe	Banking industry
CS.08	88	Germany	Europe	Information technology
CS.09	55	USA	North America	Information technology
CS.10	89	Multinational	Multinational	Healthcare
CS.11	90	USA	North America	University project
CS.12	91	Finland	Europe	Information technology
CS.13	92	New Zealand	Oceania	Finance & insurance industry
CS.14	93	USA	North America	Government organization
CS.15	68	Spain	Europe	Information technology
CS.16	53	Multinational	Multinational	Mixed
CS.17	15	Finland	Europe	Information technology
CS.18	94	Australia	Oceania	Information technology
CS.19	95	Finland	Europe	Information technology
CS.20	96	USA	North America	Information technology
CS.21	18	Multinational	Multinational	Information technology
CS.22	97	N/A	Europe	Information technology
CS.23	98	Multinational	Multinational	Mixed
CS.24	75	(Not provided)	(Not provided)	Finance & insurance industry
CS.25	99	(Not provided)	(Not provided)	Information technology
CS.26	23	Multinational	Multinational	Mixed
CS.27	100	Multinational	Multinational	Information technology
CS.28	101	UK	Europe	Information technology
CS.29	69	Multinational	Multinational	Information technology
CS.30	102	Spain	Europe	University project
CS.31	103	USA	North America	Government organization
CS.32	104	Multinational	Multinational	Information technology
CS.33	105	Sweden	Europe	Information technology
CS.34	106	Finland	Europe	Information technology
CS.35	107	Germany	Europe	Information technology
CS.36	67	(Not provided)	(Not provided)	Information technology

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

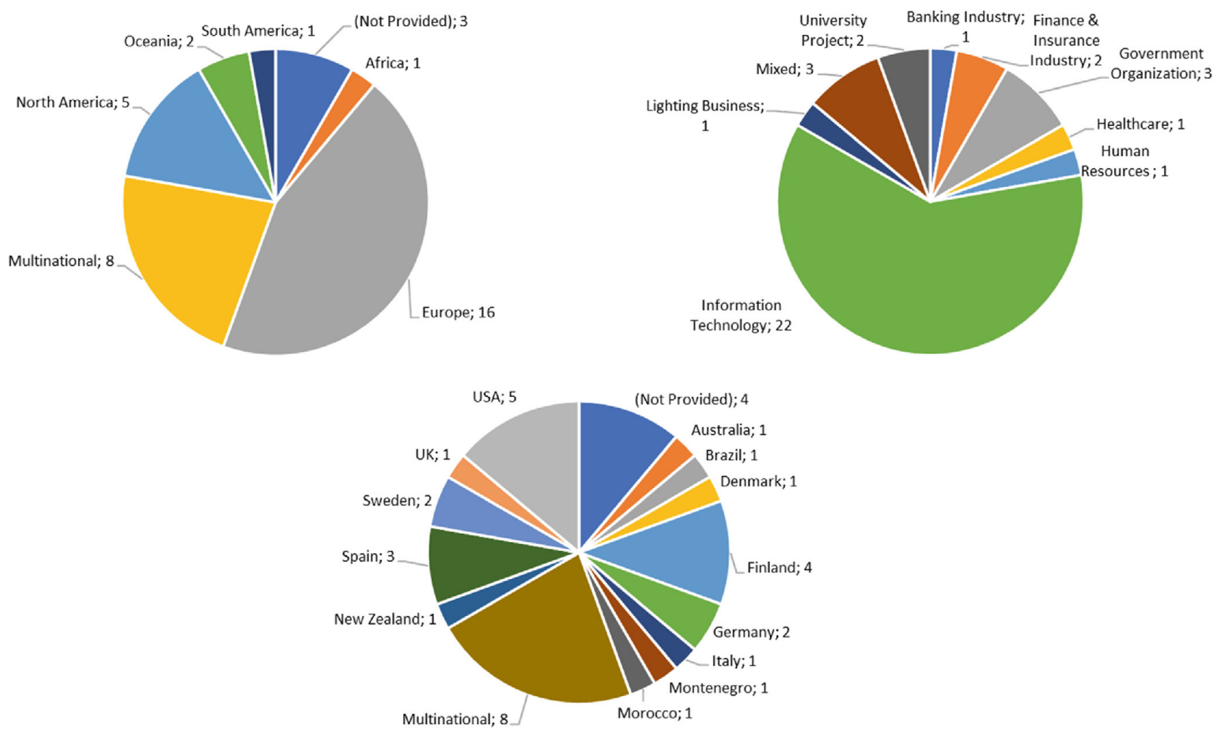


FIGURE 2 Case studies segregation

TABLE 3 DevOps benefits analysis

Benefit ID (from Table 1)	Benefit description (from Table 1)	Occurrences in case studies	Percentage of case studies
B02	Faster time to market	49	71%
B01	Cross team collaboration and communication	39	57%
B12	Decrease of manual work	38	55%
B08	Increase of team performance	30	43%
B04	Increase of code quality	27	39%
B17	Better deployment management	25	36%
B06	Improvement of system reliability	23	33%
B03	Faster and better feedback	22	32%
B10	Processes and tools standardization	19	28%
B11	Maximization of competences	20	29%
B13	Increase of customer satisfaction	19	28%
B15	Increase of employees motivation	18	26%
B09	Costs reduction	12	17%
B07	Less mean time to recover	10	14%
B19	Organizational cultural changes	9	13%
B16	More innovation	6	9%
B05	Increase of value	6	9%
B14	Less failed changes	5	7%
B18	Less security issues	2	3%

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

"work silos" in favor of DevOps.[91] Furthermore, this "empowerment of teamwork"[96] between development and operations seems to heavily tie in with other business benefits ranging from improved reliability, quality, and security[15] to competence maximization, innovation and employee motivation. Referring to Shahin's[104] work, in interviews that were held with participants of a DevOps implementation study, the opportunity of learning about overall operational and architecture aspects was often commented as a deeply useful and "growing exercise."

DevOps practices emphasize automation over manual work in the development, testing and deployment of software.[23] Over 50% of the reviewed case studies clearly mention a reduction in the volume of manual tasks. For example, in Laukkanen et al's[105] study, "manual test[ing] had been the bottleneck" for reducing feature freeze periods; with DevOps implemented, release tests for specific systems were automated, causing a reduction in the time necessary for completion. Luz et al.[28] also describe how before having DevOps implemented a vast majority of automatable tasks were done manually, often causing errors and need for rework. Similar to what we observed in our analysis of B01 (cross team collaboration and communication), the benefit of reducing manual work appears to tie in with faster delivery,[14] less failed changes, improved code quality and even employee motivation, as was observed in Gupta et al.'s[89] case study. Here, teams focused on incremental automation, focusing on a single, critical workflow at a time; upon reviewing progress, it is stated that "such small successes motivated the team", encouraging them to pursue further automation.

Although increase of value (B05), less failed changes (B14) and less security issues (B18) were not commonly and explicitly discussed in the analyzed case studies, there is room for further investigation toward better understanding how business benefits can relate to each other. Despite said links not being subject to investigation under the present research, it may not be unreasonable to consider that organizations who increase release rates and quicken their time to market (B02) are in a better position to deliver greater value to stakeholders (B05); or that those who significantly improve communication and collaboration between developers and operations (B01) may also observe a reduction in failed changes or release faults (B14).

6 | RESULTS AND DISCUSSION OF DEVOPS EMPIRICAL EVIDENCES

Table 4 presents examples for each of the 19 business benefits identified as part of our research. Where applicable, cells referring to the "Problem Solved" column are also filled in, indicating the motivation or reasoning that led to the implementation of DevOps, which led to the observed benefits. This section shows that DevOps can solve different problems on the organizations, indicating an empirical evidence of the benefits got after the DevOps implementation.

7 | CHALLENGES IN DEVOPS ADOPTION: THE OTHER SIDE OF THE COIN

Even thought, this research is about the benefits of the DevOps culture adoption, the main objective is to show what to expect when adopting DevOps. Thus, for this article some DevOps challenges will also be presented, since some of the researchers that identified DevOps benefits, were also able to identify challenges to the DevOps implementation. In Table 5, one can see which challenges were identified by the researchers that also identified benefits.

As it can be seen, some of the challenges shown in Table 5, are more related with the culture, environment, and business industry where the DevOps culture is being implemented, rather than the technologic point of view of DevOps, such as "Insufficient communication," "Deep-seated company culture" and "Industry constraints." This shows that when an organization is thinking to adopt DevOps, should self-assess if it is culturally ready for this change. Moreover, to help to mitigate this challenge, the top management of the organization should be propelling for this change so it could be example for the rest of the organization.[25] But there is a technologic challenge regarding the automation of the deployment scripts for several technologies. Organizations have multiple applications, where each of them can have different coding languages which needs its own deployment script. This requires a lot of different skills for DevOps to be able to automate these different deployment scripts.

DevOps has been evolving constantly, which could help regarding the challenge "DevOps is unclear but also evolving." The amount of publications shows that the DevOps adoption has been growing over the time, showing that organizations have been able to understand how to implement DevOps.

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 4 Case study analysis: DevOps benefit and problem solved

Benefit	Example of benefit	Problem solved	Case study
(B01)–Cross team collaboration and communication	*"The inclusion of operation team members and operation topics help the operation team to know the development topics and plan their readiness accordingly. Additionally, they take building knowledge and feedback for risk assessment without additional effort."*	*"We soon realized that with the current approach we would not be able to release the first couple of version increments. Team members in India and USA have experience in traditional software development and product management group in Germany has no experience in software development."*	CS.10
(B02)–Faster time to market	*The organization achieved a one deployment per week frequency, with one hour / one day lead time for changes.*	*"The organization size, the diversity of its departments (development, operations, security, service, QA, architecture, etc.) as well as the interaction between them, and the complexity of its processes, hampered reducing time to market, and made this company less competitive"*	CS.15
(B03)–Faster and better feedback	*"The flexibility afforded by the DevOps approach allowed the development teams to recognize, characterize and accommodate- date changes to DART's control algorithms for NEXT-C in real time. The team was able to update the test specifications and procedures in real time, and ultimately achieve the goal of demonstrating NEXT-C at Technology Readiness Level."*	*"While NEXT-C was well characterized from its own development and test perspective, there were unknowns in the specifics of DART's tailored use-case for the thruster."*	CS.14
(B04)–Increase of code quality	*"Higher levels of automation were found to drive improved quality assurance. (…) The automated DevOps production pipeline helps to ensure that every change is verified before it is pushed forward for delivery."*	*Description of a Problem / Motivation was not provided.*	CS.12
(B05)–Increase of value	*"Increase in deployment frequency from about 30 releases a month to an average of 120 releases per month."*	*"Need for a change by the business in order to remain agile and competitive. (…) Prior to DevOps, the company had been maintaining and developing its aging monolith application that was hosted in a traditional data center."*	CS.13
(B06)–Improvement of system reliability	*"The time spent in the queue for the Basic approach is about 330 times that of the Containerized approach, and similarly the queue time using the Hosted agent is 1,110 times that of the Containerized approach, which translates to significant time saved. Since all of the infrastructure is managed without any new cost incurred, yet the throughput is high, our CI/CD pipeline is very lean."*	*"We recently decided to move toward a micro-services-based architecture (…) Consequently, the number of build and release definitions would increase significantly, and the infrastructure that was utilized may no longer be sufficient."*	CS.8
(B07)–Less mean time to recover	*"This case study illustrates how rapid and simple its deployment was, in accordance with the DevOps principles, and therefore focusing on how self-service monitoring infrastructure for threats detection provided engineers—both developers and IT operators—fast and continuous feedback of the Library Energy-Efficiency System deployed into production. (…) it provides evidence of how this cybersecurity monitoring infrastructure enabled to detect threats, such as denial attacks, and helped to better anticipate spoofing problems."*	*"The development and deployment of such systems [IoT] into production as well as their operation and monitoring are highly complex due to the heterogeneity of delivery endpoints. (…) The Cluster of European Projects on Software Engineering for Services and Applications highlights the importance of ensuring Quality of Service (QoS) and correctness of IoT systems together with the complexity of such purpose as devices and software could change for various reasons such as bugs and failures, changing interfaces and implementations, and changing requirements."*	CS.30

(Continues)

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 4 (Continued)

Benefit	Example of benefit	Problem solved	Case study
(B08)–Increase of team performance	*"Considering the e-TCE system, after the DevOps adoption, it was possible to make 29 deployments on a single day. Before the DevOps adoption, and due to the rigid policies of the operations team, the deployments were schedule to occur once a week."*	*"Before DevOps, deployment activities were historically a controversial point at the TCU. Several conflicts occurred over time. Rigid procedures were created to try to avoid problems."*	CS.5
(B09)–Costs reduction	*"Most companies confirm that DevOps brings shorter response time and more frequent deployments, higher productivity, better feedback from the client and lower IT cost."*	*Description of a Problem / Motivation was not provided.*	CS.16
(B10)–Processes and tools standardization	*"Although having simpler deployment pipeline for each component or service can bring a lot of benefits but the requirement of a dedicated pipeline needs extra effort to set up the dedicated pipelines for the first time. Some of the participants reported that they were employing automation technologies such as Docker to simplify the deployment process."*	*"Our analysis of the data revealed that it was challenging for a couple of practitioners to design applications for different operations environments, in which they may have had difficulty to make consistency in a set of heterogeneous operations environments"*	CS.32
(B11)–Maximization of competences	*"The advantage is that the DevOps team teaches the student the necessary activities and attempt to integrate him\her into the team. There are no educational programs, for example, from the university that teach all necessary competencies that are required to work in a DevOps team. Hence, companies train their students or team members to get ready for the role."*	*"In the traditional silo organized IT department, there is a high level of specialist knowledge. However, in the DevOps setups, these departments are linked, and the human capitals move from highly specialized to more generalized knowledge."*	CS.26
(B12)–Decrease of manual work	*"Overall, developers are able to perform the defect validations much more quickly without having to wait to manually configure the hardware with latest software bundles having their fix in it. With this automation, developers have full control – to validate any defect they have to just pick and choose the config and within few clicks they will have a setup up and running on which, they can validate the defect in production like environment."*	*"No organizations can afford to live with manual, error prone and repeated activities in the software delivery lifecycle (…) the project teams identify this precise business need and adopt DevOps to optimize their processes, it is going to reap more fruits."*	CS.36
(B13)–Increase of customer satisfaction	*"The more and faster development team adds new features, more citizens visit the website or in the web application. (…) The deliverables may be released daily or at the end of the release cycle time. Subsequently, the development team gains faster feedback from end-users that would help in mitigating several risks"*	*Description of a Problem / Motivation was not provided.*	CS.23
(B14)–Less failed changes	*"Because every change in the code is checked at every stage of the development, and errors are discovered and resolved on the fly, the end products have fewer bugs, and the software can be readily released."*	*Description of a Problem / Motivation was not provided.*	CS.12
(B15)–Increase of employees motivation	*"The instantiation of the role rotation in the cross-functional DRR practice in our case enabled large-scale learning and KS since all team members were able to perform several roles and become more knowledgeable. (…) When team members rotate, they can take on responsibilities, develop skills, and acquire knowledge. This fosters the team's autonomy."*	*Cross-functional collaboration and team self-organization were described as major challenges.*	CS.25

(Continues)

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 4 (Continued)

Benefit	Example of benefit	Problem solved	Case study
(B16)–More innovation	*"The single-case study presented in this research was helpful to answer the two research questions. First, DevOps may be considered an approach that contributes to implementing innovation for software-defined business environments, (…) Second, the case shows that (IT) consulting companies need to transform themselves for DevOps."*	*"To develop its own consulting approach, T-Systems MMS initiated a DevOps program, which explicitly aims to improve the company's offering in the area of innovative digital services."*	CS.35
(B17)–Better deployment management	*"This has reduced errors caused by builds with wrong dependencies, incorrect deployment documents, and human errors in general, since only automated processes would deploy in the environments. (…) Initially there will be the impression that some legacy systems and technologies will not be able to be automated or benefitted by the Continuous Delivery process, but in the case of the institution of the case study, even COBOL and Power builder systems have benefited from process automation."*	*"It was identified that the deployment process executed until the beginning of this work required a lot of effort and there was a lot of bureaucracy."*	CS.24
(B18)–Less security issues	*"The success so far shows that organizations with large bureaucratic obstacles and stringent software security and accreditation requirements are able to use (Sec)DevOps processes and toolsets to produce software that meets security and accreditation requirements and ultimately satisfies their customers."*	*"Ensure that security became a continuous practice rather than being tacked on at the end."*	CS.31
(B19)–Organizational cultural changes	*"DevOps culture and mind-set, which were enriched with colocation, were observed in the wider dissemination of DevOps approach across the organization."*	*"Prior to this improvement, the team spent huge efforts in merging code and resolving merge conflicts, which were causing broken builds often."*	CS.17

TABLE 5 DevOps adoption challenges

ID	Challenge	# of references	References
C.01	Industry constraints	2	29 109
C.02	Deep-seated company culture	2	39 29
C.03	Insufficient communication	1	29
C.04	DevOps is unclear	1	29

Every new adoption for an organization takes time to learn, and DevOps is not an exception for it. To adopt DevOps, it is important to give training to the organizations employees so they can understand how to implement DevOps.

8 | VALIDITY OF THE SLRS

The authors have submitted this research to validity tests where the validity is made in four different categories, construct validity, external validity, internal validity and conclusion validity.[110] Zhou et al.[110] have performed a research to synthetize

TABLE 6 Validity tests

Pitfall description	Review test
Nonspecification of SLR's setting and sufficient details Incorrect or incomplete search terms in automatic search Incomprehensive venues or databases Inappropriate inclusion & exclusion criteria	These pitfalls are regarding the planning phase of the review. However, this research has a process and protocol correctly defined describing the decisions for the criteria, databases and search terms used. This shows a path that other researchers can follow to reproduce and replicate this research, adding more validity to this research.
Inadequate size and number of samples	For both SLRs on this study, it was possible to gather a significative amount of publications. From these samples, the authors were able to identify several benefits on the first SLR likewise, on the second review where was possible to identify several DevOps case studies.
Restricted time span	The only time restriction defined was the minimum date of research since DevOps was first presented in 2008.
Bias in study selection	To avoid the bias study selection, the authors have defined filters and criteria to select the studies on the same way for all of them.
Paper/database inaccessible	The databases used are some of the known databases by the academic/scientific and software engineering communities, showing the reliability of these databases.
Primary study duplication	To avoid duplication, the authors have applied a filter to remove duplicated articles.
Bias in data extraction	The several authors of this research have reviewed the data extracted from each author to avoid that some researchers have not identified important data.

the most common pitfalls when performing literature reviews by the different review phases. In Table 6 one can see some of these common pitfalls and how the authors have passed the test for this research.

9 | CONCLUSION

DevOps is a novel culture being adopted worldwide. The authors noticed a lack of synthetization for DevOps implementations benefits in present literature. Thus, the objective for this research was to consolidate the benefits of DevOps implementation so new practitioners know what to expect when adopting the methodology.

To accomplish this objective, the authors have chosen to perform an SLR on the benefits reported in literature. The SLR methodology is known for adding rigor to research due to the well-defined protocol that one must comply to when defining it. Additionally, a second SLR was carried out to find case studies of DevOps implementation. This second SLR was important for research, allowing for the mapping between issues that organizations faced, what were the achieved benefits, and what empirical evidence are there, respectively. Given the accomplishment of the study objective, it is possible to note that this study brings contributions to the theoretical body of knowledge by synthetizing the DevOps implementations benefits.

Regarding the findings originated from this research it is possible to state that even though there was a small number of studies in common between both SLR's, all benefits listed from the first SLR were also found on the second SLR. This demonstrates that empirical evidence exists for said benefits. It was also interesting to note that the top five benefits with more references from the first SLR are not the same as the top five benefits with more occurrences in the second SLR. Of the top five from the first SLR one can find benefits B03 and B05, while on the second SLR one finds benefits B08 and B11. Comparing B05 with B08, the authors can understand that it is easier to measure an improvement in team performance rather than a measure of value increase. As such, it makes sense to find B08 with more occurrences with empirical evidence. Furthermore, when comparing B03 with B11, one can also suppose that all the automation that DevOps encourages makes it easier to record a decrease of manual

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

work, as the effect should be immediate, while faster and better feedback often results from willingness by individuals themselves.

It is possible to see that the most reported benefits are common between the two SLR's. Those benefits are B01 and B02. This is aligned with the premises of DevOps, bridging the gap between developers and operators, working together in delivering software or products faster to their customers.

Regarding the least reported benefit it is possible to see B18 on the bottom of each SLR. It seems that this benefit is related with DevOps, but it is more specifically studied as an own discipline for security, called DevSecOps.

The fact that case study authors did not frequently provide quantitative evidence regarding the observed business benefits did increase the difficulty of establishing fully consolidated findings. This brings the opportunity of future researchers to expose metrics on how to measure the DevOps benefits, to compare how the organizations business units behave with these DevOps benefits. Another limitation to this study is due to the novelty of DevOps, the authors couldn't apply a quality filter on the SLR's for top conferences and top journals, otherwise, the total amount of articles for analysis would be low. As future work, the authors suggest performing a similar study for DevOps, but instead of benefits it could be directed at finding adoption challenges and how to overcome them. The authors believe that combining this research with a study where adoption challenges are tackled would help new DevOps practitioners clarify what is expected to be achieved with DevOps and how to go about its implementation. Moreover, this research would help organization on the decision to implement DevOps, since this research shows the trade-off between challenges and benefits. Furthermore, there may be value in studying to what extent do identify DevOps business benefits can relate to each other, building a potential series of linked, expected improvements for business.

DATA AVAILABILITY STATEMENT

Since this publication is a literature review where the literature is based on several digital libraries, all the data used on this publication can be fetched by checking the references section for each publication.

ORCID

João Faustino https://orcid.org/0000-0002-1743-5348
Daniel Adriano https://orcid.org/0000-0003-2907-855X
Rubén Pereira https://orcid.org/0000-0002-3001-5911
Miguel Mira da Silva https://orcid.org/0000-0002-0489-4465

REFERENCES

1. Ajayi BA, Hussin H. Influence of ITG on organisation performance: the mediating effect of absorptive capacity. Proceedings of the 6th International Conference on Information and Communication Technology for the Muslim World, ICT4M; 2016:1-6. doi:10.1109/ICT4M.2016.13
2. Setyadi R, Fattah A, Waseso B. Trust Effect on Business-IT Governance Alignment in Society Culture (A Case Study in Indonesia). Proceedings of the 2019 7th International Conference on Cyber and IT Service Management, CITSM 2019:1-5. doi:10.1109/CITSM47753.2019.8965411
3. Bartolini C, Salle M, Trastour D. IT service management driven by business objectives An application to incident management. Proceedings of the 2006 IEEE/IFIP Network Operations and Management Symposium NOMS 20066:45-55. doi:10.1109/NOMS.2006.1687537
4. Cavalcante V, Bianchi S, Braz A, Amorin F, Nauata N. Investigating business needs fluctuations on IT delivery operations. Proceedings of the Annual SRII Global Conference, SRII; 2014:19-26. doi:10.1109/SRII.2014.13
5. Setyadi R. Assessing trust variable impact on the information technology governance using business-IT alignment models: a model development study. Proceedings of the ICSECC 2019 - International Conference on Sustainable Engineering and Creative Computing: New Idea, New Innovation, Proceedings; 2019:218-222. doi:10.1109/ICSECC.2019.8907224
6. Wiedemann A, Forsgren N, Wiesche M. Research the practice: the DevOps phenomenon. *Commun ACM*. 2019;62(8):68.
7. Trivedi P, Sharma A. A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation. Proceedings of the 2013 2nd International Conference on Information Management in the Knowledge Economy, IMKE 2013:188-194.
8. Michener JR, Clager AT. Mitigating an oxymoron: compliance in a DevOps environments. Proceedings of the International Computer Software and Applications Conference; Vol. 1, 2016:396-398. doi:10.1109/COMPSAC.2016.155
9. Chen L. Continuous delivery: overcoming adoption challenges. *J Syst Softw*. 2017;128:72-86. doi:10.1016/j.jss.2017.02.013
10. Beck K, Beedle M, van Bennekum A, et al. Agile manifesto; 2001. Accessed January 9, 2020. https://agilemanifesto.org/

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

11. Katal A, Bajoria V, Dahiya S. DevOps: bridging the gap between development and operations. Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019; 2019:1-7; IEEE. doi:10.1109/ICCMC.2019.8819631
12. Hussaini SW. A systemic approach to re-inforce development and operations functions in delivering an organizational program. *Complex Adaptive Systems*. Vol 61. Elsevier Masson SAS; 2015:261-266. doi:10.1016/j.procs.2015.09.209
13. Rance S. In: Hanna A, ed. *ITIL 2011 Service Transition*. 1st ed. The Stationery Office; 2011.
14. Soni M. End to end automation on cloud with build pipeline: the case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery. Proceedings of the 2015 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2015; 2016:85-89. doi:10.1109/CCEM.2015.29
15. Lwakatare LE, Kilamo T, Karvonen T, et al. DevOps in practice: a multiple case study of five companies. *Inf Softw Technol*. 2017;2019(114):217-230. doi:10.1016/j.infsof.2019.06.010
16. Sharma S, Coyne B. *DevOps For Dummies*. 2nd ed. John Wiley & Sons, Inc; 2014.
17. Ebert C, Gallardo G, Hernantes J, Serrano N. DevOps. *IEEE Softw*. 2016;33(3):94-100. doi:10.1109/MS.2016.68
18. Caprarelli A, Nitto E, Tamburri D. Fallacies and pitfalls on the road to DevOps: a longitudinal industrial study; 2020:200-210. doi:10.1007/978-3-030-39306-9_15
19. Jabbari R, bin Ali N, Petersen K, Tanveer B. What is DevOps? a systematic mapping study on definitions and practices. Proceedings of the Scientific Workshop Proceedings of XP2016 on – XP '16 Workshops; 2016:1-11. doi:10.1145/2962695.2962707
20. Ivanov V. *Implementation of DevOps pipeline for Serverless Applications*. Springer; 2018. doi:10.1007/978-3-030-03673-7_4
21. Teixeira D, Pereira R, Henriques T, da Silva MM, Faustino J, Silva M. A maturity model for DevOps. *Int J Agile Syst Manag*. 2020;13(4):464-511. doi:10.1504/IJASM.2020.112343
22. Jabbari R, bin Ali N, Petersen K, Tanveer B. Towards a benefits dependency network for DevOps based on a systematic literature review. *J Softw Evol Process*. 2018;30(11):1-26. doi:10.1002/smr.1957
23. Wiedemann A, Wiesche M, Krcmar H. Integrating development and operations in cross-functional teams — Toward a DevOps competency model. SIGMIS-CPR 2019 - Proceedings of the 2019 Computers and People Research Conference; 2019:14-19. doi:10.1145/3322385.3322400
24. Lwakature LE. Devops adoption and implementation in software development practice: concept, practices, benefits and challenges; 2017:99.
25. Leite L, Rocha C, Kon F, Milojicic D, Meirelles P. A survey of DevOps concepts and challenges. *ACM Comput Surv*. 2019;52(6):1-35. doi:10.1145/3359981
26. Punjabi R, Bajaj R. User stories to user reality: a devops approach for the cloud. In: 2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2016 - Proceedings.; 2017:658-662. doi:10.1109/RTEICT.2016.7807905
27. Silva MA, Faustino J, Pereira R, Silva MM. Productivity gains of DevOps adoption in an IT team: a case study. In: Andersson B, Johansson B, S. Carlsson C, Barry ML, Linger H, Schneider C, eds. *Proceedings of the 27th International Conference on Information Systems Development*. ISD2018; 2018:13.
28. Luz WP, Pinto G, Bonifácio R. Adopting DevOps in the real world: a theory, a model, and a case study. *J Syst Softw*. 2019;157:1-16. doi:10.1016/j.jss.2019.07.083
29. Riungu-Kalliosaari L, Mäkinen S, Lwakatare LE, Tiihonen J, Männistö T. DevOps adoption benefits and challenges in practice: a case study. *Proceedings of the 17th Internacional Conference, PROFES 2016, Throndheim, Norway*. Vol 10027 LNCS. Springer International Publishing; 2016:590-597. doi:10.1007/978-3-319-49094-6_44
30. Faustino J, Pereira R, Alturas B, da Silva MM. Agile information technology service management with DevOps: an incident management case study. *Int J Agile Syst Manag*. 2020;13(4):339-389. doi:10.1504/IJASM.2020.112331
31. Teixeira D, Pereira R, Henriques TA, Silva M, Faustino J. A systematic literature review on DevOps capabilities and areas. *Int J Human Capital Inf Technol Profess (IJHCITP)*. 2020;22:1-22. doi:10.4018/IJHCITP.2020070101
32. Silva CC, Goldman A. Agile methods adoption on software development: a pilot review. Proceedings of the 2014 Agile Conference, AGILE 2014; 2014:64-65; Institute of Electrical and Electronics Engineers Inc. doi:10.1109/AGILE.2014.14
33. Rauf A, Alghafees M. Gap analysis between state of practice and state of art practices in agile software development. Proceedings of the 2015 Agile Conference, Agile 2015; 2015:102-106; Institute of Electrical and Electronics Engineers Inc. doi:10.1109/Agile.2015.21
34. Govil N, Sayrakhia M, Agnihotri P, Shukla S, Agarwal S. Analyzing the behaviour of applying agile methodologies & DevOps culture in e-commerce web application. Proceedings of the 4th International Conference on Trends in Electronics and Informatics; 2020; IEEE. doi:10.1109/ICOEI48184.2020.9142895
35. Snyder B, Curtis B. Using analytics to guide improvement during an agile-DevOps transformation. *IEEE Softw*. 2017;35(1):78-83. doi:10.1109/MS.2017.4541032
36. Dornenburg E. The path to DevOps. *IEEE Softw*. 2018;35(5):71-75. doi:10.1109/MS.2018.290110337
37. Silva MA, Faustino J, Pereira R, Mira M. Productivity gains of DevOps adoption in an IT team: a case study. Proceedings of the 27th International Conference on Information Systems Development; 2018. https://repositorio.iscte-iul.pt/handle/10071/16388
38. Furfaro A, Gallo T, Garro A, Saccà D, Tundis A. ResDevOps: a software engineering framework for achieving long-lasting complex systems. Proceedings of the 2016 IEEE 24th International Requirements Engineering Conference, RE 2016. Published online 2016:246-255. doi:10.1109/RE.2016.15
39. Shahin M. Architecting for DevOps and continuous deployment. Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference on - ASWEC '15; Vol II, 2015:147-148. doi:10.1145/2811681.2824996

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

40. Kitchenham B. Procedures for performing systematic reviews; 2004:33.
41. Okoli C. A guide to conducting a standalone systematic literature review. *Commun Assoc Inf Syst*. 2015;37(1):879-910. doi:10.17705/1cais.03743
42. Thomas G. In: Seaman J, Piper J, eds. *How to Do Your Case Study*. 2nd ed. Sage Publications Asia-Pacific Pte Ltd; 2016.
43. Webster J, Watson RT. Analyzing the past to prepare for the future: writing a literature review. *MIS Q*. 2002;26(2):xiii-xxiii. doi:10.2307/4132319
44. Veres O, Kunanets N, Pasichnyk V, Veretennikova N, Korz R, Leheza A. Development and operations - the modern paradigm of the work of IT project teams. Proceedings of the IEEE 2019 14th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2019 – Proceedings; Vol. 3, 2019:103-106. doi:10.1109/STC-CSIT.2019.8929861
45. Sousa L, Trigo A, Varajão J. DevOps – Foundations and perspectives [DevOps – Fundamentos e perspetivas]. Atas da Conferencia da Associacao Portuguesa de Sistemas de Informacao. 2019.
46. Céspedes D, Angeleri P, Melendez K, Dávila A. Software product quality in DevOps contexts: a systematic literature review. In: Mejia J, Muñoz M, Rocha ÁA, Calvo-Manzano J, eds. *Advances in Intelligent Systems and Computing*. Springer International Publishing; 2020:51-64. doi:10.1007/978-3-030-33547-2_5
47. Ottosson S. Agile principles and innovation development stig. developing and managing innovation in a fast changing and complex world: benefiting from dynamic principles; 2018:1-281. doi:10.1007/978-3-319-94045-8
48. Wahaballa A, Wahballa O, Abdellatief M, Xiong H, Qin Z. Toward unified DevOps model. Proceedings of the 6th IEEE International Conference on Software Engineering and Service Science (ICSESS); 2015:1-4. doi:10.1109/ICSESS.2015.7339039
49. Beigi-Mohammadi N, Litoiu M, Emami-Taba M, et al. A DevOps framework for quality-driven self-protection in web software systems a DevOps framework for quality-driven self-protection in web software. *System*. 2018;(October):270-274. doi:10.1145/nnnnnnn.nnnnnnn
50. Roche J. Adopting DevOps practices in quality assurance. *Commun ACM*. 2013;11(9):38-43. doi:10.1145/2524713.2524721
51. Wettinger J, Andrikopoulos V, Leymann F. Enabling DevOps collaboration and continuous delivery using diverse application. Proceedings of the On the Move to Meaningful Internet Systems: OTM 2015 Conferences; Vol 9415; 2015:348-358. doi:10.1007/978-3-319-26148-5
52. Schaefer A, Reichenbach M, Fey D. Continuous integration and automation for DevOps. *IAENG Trans Eng Technol*. 2012;170:345-358. doi:10.1007/978-94-007-4786-9
53. Díaz J, Almaraz R, Pérez J, Garbajosa J. DevOps in practice - An exploratory case study. *ACM Int Conf Proc Ser*. 2018;18-20. doi:10.1145/3234152.3234199
54. Sánchez-Gordón M, Colomo-Palacios R. Characterizing DevOps culture: a systematic literature review. In: Stamelos I, O'Connor RV, Rout T, Dorling A, eds. *Communications in Computer and Information Science*. Springer International Publishing; 2018:3-15. doi:10.1007/978-3-030-00623-5_1
55. Debroy V, Miller S, Brimble L. Building lean continuous integration and delivery pipelines by applying devops principles: a case study at varidesk. ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering; 2018:851-856. doi:10.1145/3236024.3275528
56. Kuusinen K, Balakumar V, Jepsen SC, et al. A large agile organization on its journey towards DevOps. Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA); 2018:60-63. doi:10.1109/SEAA.2018.00019
57. Machiraju S, Gaurav S. In: Srivastava S, Moodie M, Modi D, eds. *DevOps for Azure Applications*. 1st ed. Apress; 2018. doi:10.1007/978-1-4842-3643-7
58. Atwal H. *DevOps for DataOps. Practical DataOps*. Apress; 2019:161-189. doi:10.1007/978-1-4842-5104-1_7
59. Forsgren N, Humble J, Kim G. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press; 2018.
60. Bruneo D, Fritz T, Keidar-Barner S, et al. CloudWave: where adaptive cloud management meets DevOps. Proceedings of the 2014 IEEE Symposium on Computers and Communications (ISCC) Workshops; 2014:1-6. doi:10.1109/ISCC.2014.6912638
61. Perez-Palacin D, Ridene Y, Merseguer J. Quality assessment in DevOps; 2017:133-138. doi:10.1145/3053600.3053632
62. Riti P. Cloud and DevOps. *Practical Scala DSLs*. 1st ed. Apress; 2018:209-220. doi:10.1007/978-1-4842-3036-7_11
63. Elberzhager F, Arif T, Naab M, Süß I, Koban S. From agile development to DevOps: going towards faster releases at high quality - Experiences from an industrial context. *Lect Notes Bus Inf Process*. 2017;269:33-44. doi:10.1007/978-3-319-49421-0_3
64. Sahid A, Maleh Y, Belaissaoui M. An agile framework for ITS management in organizations. A case study based on DevOps. ICCWCS'17: Proceeding of the 2nd Edition of the International Conference on Computing and Wireless Communication Systems; 2017:8. doi:10.1145/3167486.3167556
65. de França BBN, Jeronimo H, Travassos GH. Characterizing DevOps by hearing multiple voices. *ACM Int Conf Proc Ser*. 2016;53-62. doi:10.1145/2973839.2973845
66. Ravichandran A, Taylor K, Waterhouse P. DevOps and real world ROI. In: Ravichandran A, Taylor K, Waterhouse P, eds. *DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory*. Apress; 2016:139-150.
67. Virmani M. Understanding DevOps & bridging the gap from continuous integration to continuous delivery. Proceedings of the 5th International Conference on Innovative Computing Technology, INTECH 2015; 2015;(Intech):78-82. doi:10.1109/INTECH.2015.7173368
68. Díaz J, Perez JE, Yague A, Villegas A, de Antona A. Devops in practice - A preliminary analysis of two multinational companies. arXiv; 2019:1-8. doi:10.1007/978-3-030-35333-9_23

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

69. Nybom K, Smeds J, Porres I. On the impact of mixing responsibilities between Devs and Ops. In: Sharp H, Hall T, eds. *Lecture Notes in Business Information Processing*. Springer International Publishing; 2016:131-143. doi:10.1007/978-3-319-33515-5_11
70. Siebra C, Lacerda R, Cerqueira I, et al. Empowering continuous delivery in software development: the DevOps strategy. In: van Sinderen M, Maciaszek LA, eds. *Software Technologies*. Communications in Computer and Information Science. Springer International Publishing; 2019:247-265. doi:10.1007/978-3-030-29157-0_11
71. Benguria G, Alonso J, Etxaniz I, Orue-Echevarria L, Escalante M. Agile development and operation of complex systems in multi-technology and multi-company environments: following a DevOps approach. *Commun Comput Inf Sci*. 2018;896:15-27. doi:10.1007/978-3-319-97925-0_2
72. Angara J, Gutta S, Prasad S. DevOps with continuous testing architecture and its metrics model. In: Sa PK, Bakshi S, Hatzilygeroudis IK, Sahoo MN, eds. *Advances in Intelligent Systems and Computing*. Springer; 2018:271-281. doi:10.1007/978-981-10-8633-5_28
73. Kaiser AK. *Reinventing ITIL® in the Age of DevOps*. Apress; 2018. doi:10.1007/978-1-4842-3976-6
74. Cuppett MS. DevOps, DBAs, and DBaaS; 2016:73-85. doi:10.1007/978-1-4842-2208-9
75. Albuquerque AB, Cruz VL. Implementing DevOps in legacy systems. In: Silhavy R, Silhavy P, Prokopova Z, eds. *Advances in Intelligent Systems and Computing*. Springer International Publishing; 2019:143-161. doi:10.1007/978-3-030-00184-1_14
76. Baudry B, Harrand N, Schulte E, et al. A spoonful of DevOps helps the GI go down. *Proc Int Conf Softw Eng*. 2018;35-36. doi:10.1145/3194810.3194818
77. Lai ST, Leu FY. *A Micro Services Quality Measurement Model for Improving the Efficiency and Quality of DevOps*. Vol 773. Springer International Publishing; 2019. doi:10.1007/978-3-319-93554-6_55
78. Mala DJ, Reynold PA. Towards green software testing in agile and devops using cloud virtualization for environmental protection. *Software Engineering in the Era of Cloud Computing*. Springer; 2020:277-297. doi:10.1007/978-3-030-33624-0_11
79. Masombuka T, Mnkandla E. A DevOps collaboration culture acceptance model. *ACM Int Conf Proc Ser*. 2018;279-285. doi:10.1145/3278681.3278714
80. Jaatun MG, Cruzes DS, Luna J. DevOps for better software security in the cloud. *ACM Int Conf Proc Ser*. 2017;F1305. doi:10.1145/3098954.3103172
81. Jaatun MG. Software security activities that support incident management in secure DevOps. *ACM Int Conf Proc Ser*. 2018. doi:10.1145/3230833.3233275
82. Rahman AAU, Williams L. Software security in DevOps: synthesizing practitioners' perceptions and practices. Proceedings - International Workshop on Continuous Software Evolution and Delivery, CSED; 2016. doi:10.1145/2896941.2896946
83. Kim G, Debois P, Willis J, Humble J. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press; 2016.
84. Šmite D, Gonzalez-Huerta J, Moe NB. "When in Rome, Do as the Romans Do": cultural barriers to being agile in distributed teams. In: Stray V, Hoda R, Paasivaara M, Kruchten P, eds. *Agile Processes in Software Engineering and Extreme Programming*. Lecture Notes in Business Information Processing. Springer International Publishing; 2020:145-161. doi:10.1007/978-3-030-49392-9_10
85. Colomo-Palacios R, Fernandes E, Soto-Acosta P, Larrucea X. A case analysis of enabling continuous software deployment through knowledge management. *Int J Inf Manag*. 2018;40:186-189. doi:10.1016/j.ijinfomgt.2017.11.005
86. De Sanctis M, Bucchiarone A, Trubiani C. A DevOps perspective for QoS-aware adaptive applications. In: Bruel JM, Mazzara M, Meyer B, eds. *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Lecture Notes in Computer Science. Springer International Publishing; 2020:95-111. doi:10.1007/978-3-030-39306-9_7
87. Šćekić M, Gazivoda M, Šćepanović S, Nikolić J. Application of DevOps approach in developing business intelligence system in bank. Proceedings of the 2018 7th Mediterranean Conference on Embedded Computing (MECO); 2018:1-4. doi:10.1109/MECO.2018.8406047
88. Mohan V, Ben Othmane L, Kres A. BP: security concerns and best practices for automation of software deployment processes: an industrial case study; 2018:21-28. doi:10.1109/SecDev.2018.00011
89. Gupta RK, Venkatachalapathy M, Jeberla FK. Challenges in adopting continuous delivery and DevOps in a globally distributed product team: a case study of a healthcare organization. In: *Proceedings - 2019 ACM/IEEE 14th International Conference on Global Software Engineering, ICGSE 2019*. IEEE; 2019:30-34. doi:10.1109/ICGSE.2019.00020
90. Sampedro Z, Holt A, Hauser T. Continuous integration and delivery for HPC: using singularity and Jenkins. *Proceedings of the Practice and Experience on Advanced Research Computing PEARC '18*. Association for Computing Machinery; 2018:1-6. doi:10.1145/3219104.3219147
91. Riungu-Kalliosaari L, Mäkinen S, Lwakatare LE, Tiihonen J, Männistö T. DevOps adoption benefits and challenges in practice: A case study. Proceedings of the 17th Internacional Conference, PROFES 2016, Throndheim, Norway; Vol 10027, 2016:590-597; LNCS. doi:10.1007/978-3-319-49094-6_44
92. Senapathi M, Buchan J, Osman H. DevOps capabilities, practices, and challenges: insights from a case study. *ACM Int Conf Proc Ser*. 2018;F1377. doi:10.1145/3210459.3210465
93. Heistand C, Thomas J, Tzeng N, et al. DevOps for spacecraft flight software. Proceedings of the IEEE Aerospace Conference Proceedings; March 2019:1-16. doi:10.1109/AERO.2019.8742143
94. Ghantous GB, Gill AQ. DevOps reference architecture for multi-cloud IOT applications. Proceedings of the 2018 IEEE 20th Conference on Business Informatics (CBI); Vol. 01; 2018:158-167. doi:10.1109/CBI.2018.00026
95. Smeds J, Nybom K, Porres I. DevOps: a definition and perceived adoption impediments. In: Lassenius C, Dingsøyr T, Paasivaara M, eds. *Agile Processes in Software Engineering and Extreme Programming*. Lecture Notes in Business Information Processing. Springer International Publishing; 2015:166-177. doi:10.1007/978-3-319-18612-2_14

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

96. Jiménez M, Rivera LF, Villegas NM, Tamura G, Müller HA, Gallego P. DevOps' shift-left in practice: an industrial case of application. In: Bruel JM, Mazzara M, Meyer B, eds. *Lecture Notes in Computer Science*. Springer International Publishing; 2019:205-220. doi:10.1007/978-3-030-06019-0_16
97. Hemon A, Lyonnet B, Rowe F, Fitzgerald B. From agile to DevOps: smart skills and collaborations. *Inf Syst Front*. 2020;22(4):927-945. doi:10.1007/s10796-019-09905-1
98. AL-Zahran S, Fakieh B. How DevOps practices support digital transformation. *Int J Adv Trends Comput Sci Eng*. 2020;9:2780-2788. doi:10.30534/ijatcse/2020/46932020
99. Hemon-Hildgen A, Fitzgerald B, Lyonnet B, Rowe F. Innovative practices for knowledge sharing in large-scale DevOps. *IEEE Softw*. 2019;37(3):30-37. doi:10.1109/MS.2019.2958900
100. Gall M, Pigni F. Leveraging DevOps for mission critical software. AMCIS; 2018. https://aisel.aisnet.org/amcis2018/ITProjMgmt/Presentations/2
101. Jones S, Noppen J, Lettice F. Management challenges for devops adoption within UK SMEs. QUDOS 2016 - Proceedings of the 2nd International Workshop on Quality-Aware DevOps, co-located with ISSTA 2016; 2016:7-11. doi:10.1145/2945408.2945410
102. Díaz J, Pérez JE, Lopez-Peña MA, Mena GA, Yagüe A. Self-service cybersecurity monitoring as enabler for DevSecOps. *IEEE Access*. 2019;7:100283-100295. doi:10.1109/ACCESS.2019.2930000
103. Bruza M. An Analysis of Multi-domain Command and Control and the Development of Software Solutions through DevOps Toolsets and Practices; 2018.
104. Shahin M, Babar MA, Zhu L. The intersection of continuous deployment and architecting process: practitioners' perspectives. Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '16. Association for Computing Machinery; 2016:1-10. doi:10.1145/2961111.2962587
105. Laukkanen E, Paasivaara M, Itkonen J, Lassenius C, Arvonen T. Towards continuous delivery by reducing the feature freeze period: a case study. Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP); 2017:23-32. doi:10.1109/ICSE-SEIP.2017.21
106. Lwakatare LE, Kuvaja P, Oivo M. An exploratory study of DevOps: extending the dimensions of DevOps with practices. Proceedings of the 11th International Conference on Software Engineering Advances (ICSEA); 2016.
107. Alt R, Auth G, Kögler C. Transformation of consulting for software-defined businesses: lessons from a DevOps case study in a German IT company. *Advances in Consulting Research*. Springer; 2018:385-403. doi:10.1007/978-3-319-95999-3_19
108. Düllmann TF, Paule C, van Hoorn A. Exploiting DevOps practices for dependable and secure continuous delivery pipelines. Proceedings of the 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE); 2018:27-30.
109. Laukkarinen T, Kuusinen K, Mikkonen T. DevOps in regulated software development: case medical devices. Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track, ICSE-NIER 2017. doi:10.1109/ICSE-NIER.2017.20
110. Zhou X, Jin Y, Zhang H, Li S, Huang X. A map of threats to validity of systematic literature reviews in software engineering. Proceedings - Asia-Pacific Software Engineering Conference, APSEC; 2016:153-160. doi:10.1109/APSEC.2016.031
111. Erich FMA, Amrit C, Daneva M. A qualitative study of DevOps usage in practice. *J Softw Evol Process*. 2017;29(6):1-20. doi:10.1002/smr.1885
112. Metzger S, Durden D, Sturtevant C, et al. eddy4R 0.2.0: a DevOps model for community-extensible processing and analysis of eddy-covariance data based on R, Git, Docker, and HDF5. *Geosci Model Dev*. 2017;10(9):3189-3206. doi:10.5194/gmd-10-3189-2017
113. Gupta V, Kapur PK, Kumar D. Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. *Inf Softw Technol*. 2017;92:75-91. doi:10.1016/j.infsof.2017.07.010
114. Chen B. Improving the software logging practices in devops. Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion, ICSE-Companion 2019; 2019:194-197; IEEE. doi:10.1109/ICSE-Companion.2019.00080
115. Barna C, Khazaei H, Fokaefs M, Litoiu M. Delivering elastic containerized cloud applications to enable DevOps. Proceedings - 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2017; 2017:65-75. doi:10.1109/SEAMS.2017.12
116. Olszewska M, Waldén M. DevOps meets formal modelling in high-criticality complex systems. Proceedings of the 1st International Workshop on Quality-Aware DevOps, QUDOS 2015 – Proceedings; 2015:7-12. doi:10.1145/2804371.2804373
117. Muñoz M, Negrete M. Reinforcing DevOps generic process with a guidance based on the basic profile of ISO/IEC 29110. In: Mejia J, Muñoz M, Rocha Á, A. Calvo-Manzano J, eds. *Advances in Intelligent Systems and Computing*. Springer International Publishing; 2020:65-79. doi:10.1007/978-3-030-33547-2_6
118. Cruzes DS, Melsnes K, Marczak S. Testing in a DevOps era: perceptions of testers in Norwegian organisations. In: Misra S, Gervasi O, Murgante B, et al., eds. *Lecture Notes in Computer Science*. Springer International Publishing; 2019:442-455. doi:10.1007/978-3-030-24305-0_33
119. Daneva M, Bolscher R. What we know about software architecture styles in continuous delivery and DevOps? In: van Sinderen M, Maciaszek LA, eds. *Communications in Computer and Information Science*. Springer International Publishing; 2020:26-39. doi:10.1007/978-3-030-52991-8_2
120. Scheaffer J, Ravichandran A, Martins A. In: McDermott S, Berendson L, Fernando R, eds. *The Kitty Hawk Venture*. 1st ed. Apress; 2018. doi:10.1007/978-1-4842-3661-1
121. Hüttermann M. *DevOps for Developers*. 1st ed. Apress; 2012. doi:10.1007/978-1-4302-4570-4
122. Davis A. DevOps. *Mastering Salesforce DevOps*. Apress; 2019. doi:10.1007/978-1-4842-5473-8_3

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

123. Kuruba M. DevOps for IT service reliability and availability. In: Karanki D, Vinod G, Ajit S, eds. *Advances in RAMS Engineering*. 1st ed. Apress; 2019:149-183. doi:10.1007/978-3-030-36518-9_6
124. Familiar B, Barnes J. DevOps using PowerShell, ARM, and VSTS. *Business in Real-Time Using Azure IoT and Cortana Intelligence Suite*. 1st ed. Apress; 2017:21-93. doi:10.1007/978-1-4842-2650-6_2
125. Muñoz M, Díaz O. DevOps: foundations and its utilization in data center. In: J. MG, M. M, M. R, W. N, R O. *Engineering and Management of Data Centers*. Springer; 2017:205-225. doi:10.1007/978-3-319-65082-1_10
126. Nagarajan AD, Overbeek SJ. A DevOps implementation framework for large agile-based financial organizations. In: H. P, C. D, H. P, C. A, D. R, R. M. *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*. Springer; 2018:1-17. doi:10.1007/978-3-030-02610-3_10
127. Wang W, Casale G, Iuhasz G. Closing the loop between ops and dev. In: Di Nitto E, Matthews P, Petcu D, Solberg A eds. *Model-Driven Development and Operationof Multi-Cloud Applications*. Springer; 2017:95-105. doi:10.1007/978-3-319-46031-4_10
128. de Feijter R, Overbeek S, van Vliet R, Jagroep E, Brinkkemper S. DevOps competences and maturity for software producing organizations. In: Gulden J, Reinhartz-Berger I, Schmidt R, Guerreiro S, Guédria W, Bera P, eds. *Lecture Notes in Business Information Processing*. Springer International Publishing; 2018:244-259. doi:10.1007/978-3-319-91704-7_16
129. Cito J, Wettinger J, Lwakatare LE, Borg M, Li F. Feedback from operations to software development—A DevOps perspective on runtime metrics and logs. In: Bruel JM, Mazzara M, Meyer B, eds. *Lecture Notes in Computer Science*. Springer International Publishing; 2019:184-195. doi:10.1007/978-3-030-06019-0_14
130. Alban D, Eynaud P, Malaurent J, Richet JL, Vitari C. *Information Systems Management: Governance, Urbanization and Alignment*. ISTE Ltd and John Wiley & Sons, Inc; 2019. doi:10.1016/B978-012226570-9/50107-1
131. Beulen E. Implementing and contracting agile and DevOps: a survey in the Netherlands. In: Kotlarsky J, Oshri I, Willcocks L, eds. *Lecture Notes in Business Information Processing*. Springer International Publishing; 2019:124-146. doi:10.1007/978-3-030-15850-7_7
132. Colavita F. DevOps movement of enterprise agile breakdown silos, create collaboration, increase quality, and application speed. In: Ciancarini P, Sillitti A, Succi G, Messina A, eds. *Advances in Intelligent Systems and Computing*. Springer International Publishing; 2016:203-213. doi:10.1007/978-3-319-27896-4_17
133. Fishman N, Stryker C. In: Minatel J, ed. *Smarter Data Science*. 1st ed. John Wiley & Sons, Inc; 2020. doi:10.1002/9781119697985
134. Johng H, Kalia AK, Xiao J, Vuković M, Chung L. Harmonia: a continuous service monitoring framework using DevOps and service mesh in a complementary manner. In: Yangui S, Bouassida Rodriguez I, Drira K, Tari Z, eds. *Service-Oriented Computing. Lecture Notes in Computer Science*. Springer International Publishing; 2019:151-168. doi:10.1007/978-3-030-33702-5_12
135. Kang H, Le M, Tao S. Container and microservice driven design for cloud infrastructure DevOps. Proceedings of the 2016 IEEE International Conference on Cloud Engineering (IC2E); 2016:202-211. doi:10.1109/IC2E.2016.26
136. Lewerentz M, Bluhm T, Daher R, et al. Implementing DevOps practices at the control and data acquisition system of an experimental fusion device. *Fusion Eng Design*. 2018;146(October):40-45. doi:10.1016/j.fusengdes.2018.11.022
137. Lewis B, Smith I, Fowler M, Licato J. Enabling DevOps for containerized data-intensive applications: an exploratory study. Proceedings of the Modern Artificial Intelligence and Cognitive Science Conference, MAICS 2017; 2017:189-190. doi:10.1145/1235
138. Palermo J. In: Murray J, Berendson L, Balzano J, eds. *NET DevOps for Azure*. 1st ed. Apress; 2019. doi:10.1007/978-1-4842-5343-4
139. Ravichandran A, Taylor K, Waterhouse P. DevOps for digital leaders; 2016. doi:10.1007/978-1-4842-1842-6
140. Rong G, Zhang H, Shao D. CMMI guided process improvement for DevOps projects. Proceedings of the International Workshop on Software and Systems Process - ICSSP '16; 2016:76-85. doi:10.1145/2904354.2904372
141. Senapathi M, Buchan J, Osman H. DevOps capabilities, practices, and challenges: insights from a case study. EASE; 2018. doi:10.1145/3210459.3210465
142. Tomlinson T. Drupal 8 DevOps. *Enterprise Drupal 8 Development*. Apress; 2017:271-279. doi:10.1007/978-1-4842-0253-1_11
143. Wettinger J, Breitenbücher U, Falkenthal M, Leymann F. Collaborative gathering and continuous delivery of DevOps solutions through repositories. *Comput Sci Res Dev*. 2017;32(3-4):281-290. doi:10.1007/s00450-016-0338-z
144. Ali N, Daneth H, Hong JE. A hybrid DevOps process supporting software reuse: a pilot project. *J Softw Evol Process*. 2020;32(7):1-23. doi:10.1002/smr.2248
145. Gotimer G, Stiehm T. DevOps advantages for testing: Increasing quality through continuous delivery. *CrossTalk*. 2016;29(3):13-18.
146. Morales JA, Yasar H, Volkman A. Implementing DevOps practices in highly regulated environments. *ACM Int Conf Proc Ser*. 2018;F1477. doi:10.1145/3234152.3234188
147. Pérez JF, Wang W, Casale G. Towards a DevOps approach for software quality engineering. WOSP-C 2015 - Proceedings of the 2015 ACM/SPEC Workshop on Challenges in Performance Methods for Software Development, in Conjunction with ICPE; 2015. doi:10.1145/2693561.2693564
148. de Kort W. In: DeWolf J, Pundick D, eds. *DevOps on the Microsoft Stack*. 1st ed. Apress; 2016. doi:10.1007/978-1-4842-1446-6
149. Rittgen P, Cronholm S, Göbel H. Towards a model for assessing collaboration capability between development and operations. In: Walker A, O'Connor R V, Messnarz R, eds. *Communications in Computer and Information Science*. Springer International Publishing; 2019:111-122. doi:10.1007/978-3-030-28005-5_9
150. Murphy GC, Kersten M. Towards bridging the value gap in DevOps. In: Bruel JM, Mazzara M, Meyer B, eds. *Lecture Notes in Computer Science*. Springer International Publishing; 2020:181-190. doi:10.1007/978-3-030-39306-9_13
151. Arulkumar V, Lathamanju R. Start to finish automation achieve on cloud with build channel: by DevOps method. *Proc Comput Sci*. 2019;2019(165):399-405. doi:10.1016/j.procs.2020.01.032
152. Medina O, Schumann E. In: Murray J, Berendson L, Balzano J, eds. *DevOps for SharePoint*. 1st ed. Apress; 2018. doi:10.1007/978-1-4842-3688-8

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

153. Koilada DK. Business model innovation using modern DevOps. Proceedings of the 2019 IEEE Technology and Engineering Management Conference, TEMSCON 2019; 2019:1-6. doi:10.1109/TEMSCON.2019.8813557
154. Rong G, Jin Z, Zhang H, Zhang Y, Ye W, Shao D. DevDocOps: enabling continuous documentation in alignment with DevOps. *Softw Pract Exp*. 2019;50(3):210-226. doi:10.1002/spe.2770
155. Forsgren N, Kersten MIK. DevOps metrics. *ACM Queue*. 2017;61(december):1-16.
156. Cukier D. DevOps patterns to scale web applications using cloud services. SPLASH 2013 - Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, and Applications: Software for Humanity; 2013;(Figure 2):143-152. doi:10.1145/2508075.2508432
157. Combemale B, Wimmer M. Towards a model-based DevOps for cyber-physical systems. In: Bruel JM, Mazzara M, Meyer B, eds. *Lecture Notes in Computer Science*. Springer International Publisheing; 2020:84-94. doi:10.1007/978-3-030-39306-9_6

How to cite this article: Faustino J, Adriano D, Amaro R, Pereira R, da Silva MM. DevOps benefits: A systematic literature review. *Softw: Pract Exper*. 2022;52(9):1905-1926. doi: 10.1002/spe.3096

APPENDIX A. CONCEPT-CENTRIC APPROACH FOR BENEFITS AND LITERATURE

This appendix provides a mapping between all the authors (references) that mention a certain DevOps benefit.

Concept ID	Reference
B01	6,12,15,17,22,25,28,29,44–46,49–51,56,57,62–66,68,69,72,75,79,82,89,92,107,111–129
B02	6,12,14,17,22,23,25,39,51–54,56,58,63,65,74,78,91,101,113,121–123,125,126,129–143
B03	6,14,15,22,23,39,44,45,49,51,53–56,58,60,63,66,72,74,75,93,109,111,117,118,121,122,125,133,138,144–150
B04	14,15,20,22,28,29,45,51,57,65,70,75–77,92,93,107,109,113,118–120,122,125,128,132,134,145,148,150–152
B05	6,17,22,28,29,45,50,56,58,60,65,68,89,92,101,118,120–122,125,133,134,138,153–155
B06	6,14,22,46,57,61,64,65,67,70,93,107,116,118,119,121–123,130,147,152,156
B07	6,14,23,45,46,56,62–64,66,68,81,122,123,135,142,145
B08	12,22,28,29,44,45,53,57,63–65,69,113,134,144,151,153
B09	22,28,51,65–67,74,107,125,129,130,143,145
B10	22,65,67–71,109,124,125,140,144,146
B11	15,22,23,25,29,54,65,66,69,92,117,118,123
B12	22,56,65,67,70,72–74,76,117,148
B13	22,29,51,55,56,58,64,65,77,113,133
B14	22,46,58,65,66,74,75,78,118,142,157
B15	12,15,54,63,69,79,92,113,121
B16	14,29,117,133,136,150,153,157
B17	56,63,65,69,73
B18	80–82,122,134
B19	15,76

APPENDIX B. IDENTIFIED BUSINESS BENEFITS PER DEVOPS IMPLEMENTATION CASE STUDY

In this appendix there is the mapping between the case studies and the benefits identified on each of these case studies.

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

ID	Case study number	DevOps benefit concept ID
CS.1	1	B01
CS.2	2	B02; B04; B17
CS 3	3	B01; B06
CS.4	4	B11
CS.5	5	B01; B02; B08; B10; B17
CS.6	6	B01; B03; B13; B04; B06; B10; B12;
CS.7	7	B01; B02; B06; B10; B17
CS.8	8	B01; B02; B06; B10; B18; B17;
CS.9	9	B01; B04; B06; B16
CS.10	10	B01; B02; B03; B10; B17;
CS.11	11	B04; B06
CS.12	12.1	B01; B02; B03; B04; B05; B06; B07; B08; B10; B12; B14; B15; B17
	12.2	B01; B02; B03; B04; B05; B06; B07; B08; B10; B12; B14; B15; B17
	12.3	B01; B02; B03; B04; B05; B06; B07; B08; B10; B12; B14; B15; B17
CS.13	13	B01; B04; B05; B10; B13; B15; B16
CS.14	14	B06; B17
CS.15	15.1	B02; B06; B07; B17
	15.2	B02; B06; B07; B17
CS.16	16.1	B02; B03; B08; B09; B12; B13
	16.2	B02; B03; B08; B09; B12; B13
	16.3	B02; B03; B08; B09; B12; B13
	16.4	B02; B03; B08; B09; B12; B13
	16.5	B02; B03; B08; B09; B12; B13
	16.6	B02; B03; B08; B09; B12; B13
	16.7	B02; B03; B08; B09; B12; B13
	16.8	B02; B03; B08; B09; B12; B13
	16.9	B02; B03; B08; B09; B12; B13
	16.10	B02; B03; B08; B09; B12; B13
	16.11	B02; B03; B08; B09; B12; B13
CS.17	17.1	B4; B6; B7; B9; B10; B11; B17; B19
	17.2	B02; B04; B08; B15; B19
	17.3	B01; B02; B04; B06; B07; B08; B15; B19
	17.4	B02; B04; B06; B07; B08; B15; B19
	17.5	B02; B04; B08; B15; B19
CS.18	18	B01; B02; B04; B08; B12
CS.19	19	B01
CS.20	20	B04; B06
CS.21	21	No benefits identified
CS.22	22	B01; B03; B08; B11
CS.23	23.1	B02; B06; B11; B13; B15; B16; B17
	23.2	B02; B03; B06; B11; B13; B16; B15
CS.24	24	B01; B02; B03; B04; B06; B07; B12 B13; B17
CS.25	25	B01; B08; B11; B12; B15; B19
CS.26	26.1	B01; B08; B11; B12; B15
	26.2	B01; B08; B11; B12; B15
	26.3	B01; B08; B11; B12; B15
	26.4	B01; B08; B11; B12; B15
CS.27	27	B01; B02
CS.28	28	B01
CS.29	29	B01; B06; B11; B12; B14; B15; B19
CS.30	30	B01; B06; B07; B08; B12; B19
CS.31	31	B01; B02; B03; B04; B05; B06; B08; B13; B15; B16; B17; B18
CS.32	32.1	B01; B02; B04; B10; B11; B12; B17
	32.2	B01; B02; B04; B10; B11; B12; B17
	32.3	B01; B02; B04; B10; B11; B12; B17
	32.4	B01; B02; B04; B10; B11; B12; B17
	32.5	B01; B02; B04; B10; B11; B12; B17
	32.6	B01; B02; B04; B10; B11; B12; B17
	32.7	B01; B02; B04; B10; B11; B12; B17
	32.8	B01; B02; B04; B10; B11; B12; B17
	32.9	B01; B02; B04; B10; B11; B12; B17
	32.10	B01; B02; B04; B10; B11; B12; B17
CS.33	33	B02; B06; B14
CS.34	34.1	B02; B12
	34.2	B02; B12
	34.3	B02; B12
	34.4	B02; B12
CS.35	35	B01; B02; B03; B05; B13; B16
CS.36	36	B02; B03; B08; B09; B12; B13; B19

1097024x, 2022, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3096 by Cochrane Portugal, Wiley Online Library on [13/01/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

CHAPTER 6

Article #5

The fifth article (A5) presents a systematic mapping of DevOps capabilities to software development cycles and identifies key capabilities and processes that significantly influence software delivery performance and quality. It provides a means to leverage DevOps within software development and maintenance. This study strongly suggested that technical capabilities in DevOps, have strong associations with the technical LCP processes. Secondly, there are DevOps measurement capabilities and agreement processes, which a significant proportion of LCPs also benefit from. From Table 6.12 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 the literature of their relationship, influencing activities and tasks of the respective LCPs.

Practical examples and a structured approach are given for integrating DevOps into software development and maintenance. Examples and a structured approach are given for integrating DevOps into software development and maintenance. This approach can be used to make an organization’s DevOps processes work better. The findings are expected to help organizations successfully adopt DevOps by providing a clear understanding of the relationship between DevOps capabilities and LCPs. In Figure 6.9 a diagram shows the process improvements driving LCP outcomes with exceptional DevOps Capabilities.

Finally, future research directions are suggested, focusing on a practical case study evaluating overcoming adoption challenges. This will require empirical validation and refinement of the proposed approach.

Article details:

- Title: Mapping DevOps Capabilities to the Software Life Cycle: A Systematic Literature Review
- Date: September 2024
- Journal: Information and Software Technology
- Scimago Journal Rank: Q1
- Publisher: Elsevier B.V.

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
0950-5849/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

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].

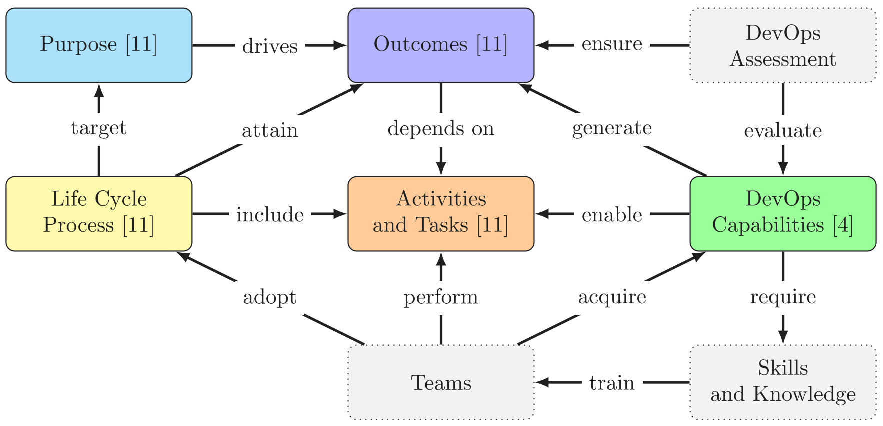


Fig. 1. Relating DevOps Capabilities [4] to LCPs[11] conceptual map.

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

Cultural

- Cross team collaboration
- Support learning and experimentation
- Open source software adoption
- Transformational leadership
- Westrum organizational culture
- Blameless Postmortems
- Job satisfaction

Measurement

- Monitoring, Observability and autoscaling
- Emergency response
- Monitor systems to inform business decisions
- Working in progress limits
- Visual management Capabilities

Process

- Continuous Improvement of processes
- Focus on people, process and technology
- Working in small batches
- Lightweight change approval
- Visibility of work in the value stream
- Customer focus/feedback
- Data-driven approach for improvements

Technical

- Continuous Integration
- Continuous Delivery automation
- Test Automation and environments
- Version Control System
- Empower teams to make decisions
- Configuration Management
- 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
- Code maintainability

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

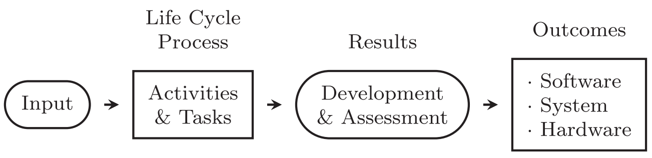


Fig. 3. Processes and Life Cycle Processes in Software Engineering (in Fig. 4).

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].

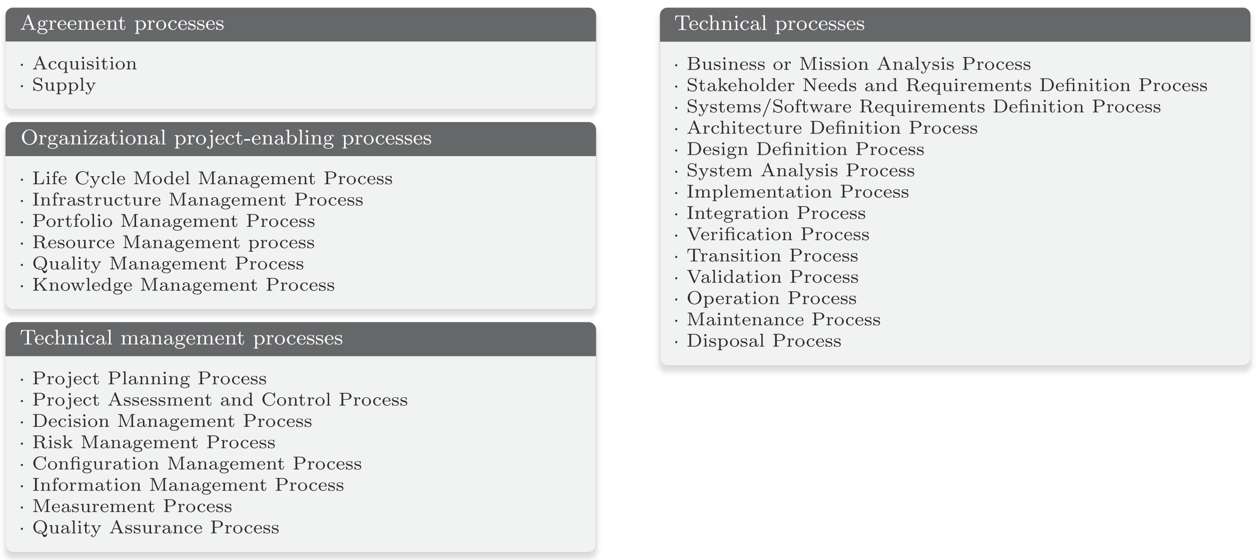
Software processes and life cycle models play a vital role in supporting organizational goals and strategies related to software consumption or development. They are integral to Information Technology (IT) governance, aiming to deliver sustainable, standardized services and achieve desired objectives. 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.



Agreement processes

· Acquisition
· Supply

Organizational project-enabling processes

· Life Cycle Model Management Process
· Infrastructure Management Process
· Portfolio Management Process
· Resource Management process
· Quality Management Process
· Knowledge Management Process

Technical management processes

· Project Planning Process
· Project Assessment and Control Process
· Decision Management Process
· Risk Management Process
· Configuration Management Process
· Information Management Process
· Measurement Process
· Quality Assurance Process

Technical processes

· Business or Mission Analysis Process
· Stakeholder Needs and Requirements Definition Process
· Systems/Software Requirements Definition Process
· Architecture Definition Process
· Design Definition Process
· System Analysis Process
· Implementation Process
· Integration Process
· Verification Process
· Transition Process
· Validation Process
· Operation Process
· Maintenance Process
· Disposal Process

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

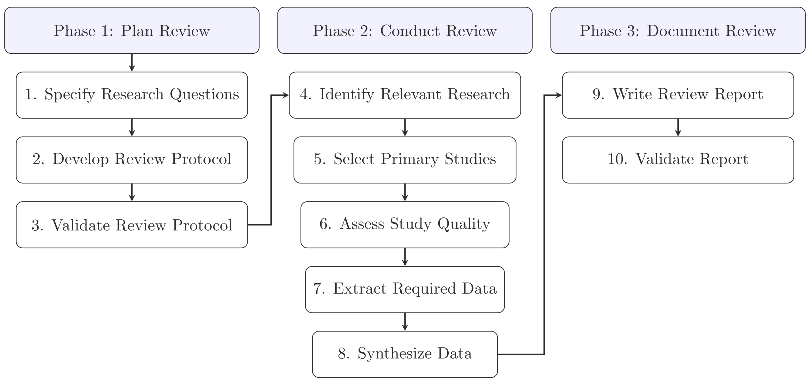


Fig. 5. The SLR process.
Source: Adapted from Kitchenham et al. [35].

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/

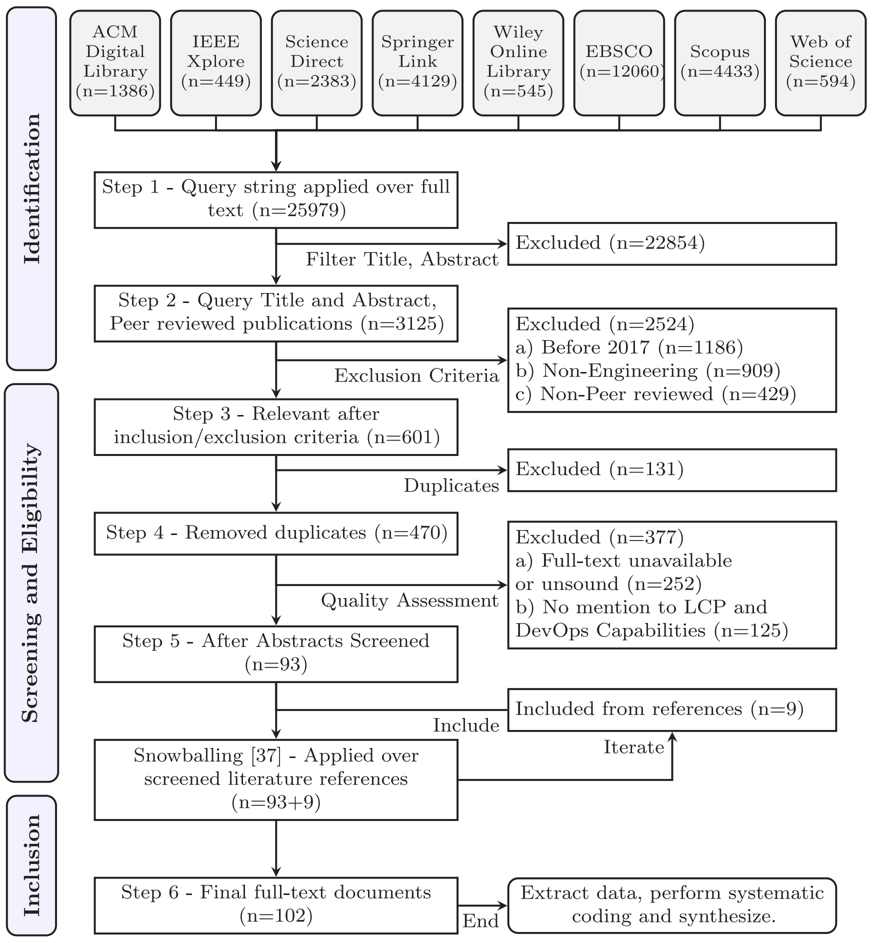

Identification
Screening and Eligibility
Inclusion
ACM Digital Library (n=1386)
IEEE Xplore (n=449)
Science Direct (n=2383)
Springer Link (n=4129)
Wiley Online Library (n=545)
EBSCO (n=12060)
Scopus (n=4433)
Web of Science (n=594)
Step 1 - Query string applied over full text (n=25979)
Filter Title, Abstract
Excluded (n=22854)
Step 2 - Query Title and Abstract, Peer reviewed publications (n=3125)
Exclusion Criteria
Excluded (n=2524)
a) Before 2017 (n=1186)
b) Non-Engineering (n=909)
c) Non-Peer reviewed (n=429)
Step 3 - Relevant after inclusion/exclusion criteria (n=601)
Duplicates
Excluded (n=131)
Step 4 - Removed duplicates (n=470)
Quality Assessment
Excluded (n=377)
a) Full-text unavailable or unsound (n=252)
b) No mention to LCP and DevOps Capabilities (n=125)
Step 5 - After Abstracts Screened (n=93)
Include
Included from references (n=9)
Iterate
Snowballing [37] - Applied over screened literature references (n=93+9)
Step 6 - Final full-text documents (n=102)
End
Extract data, perform systematic coding and synthesize.

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

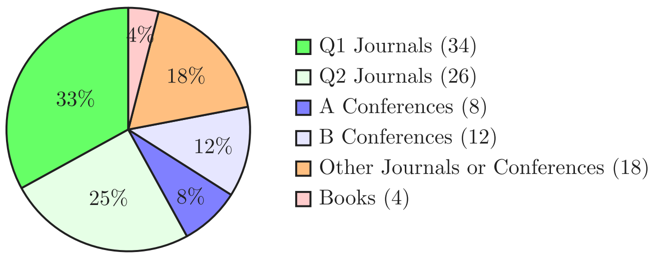


Fig. 7. Break down of publication quality based on ranking.

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.

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.

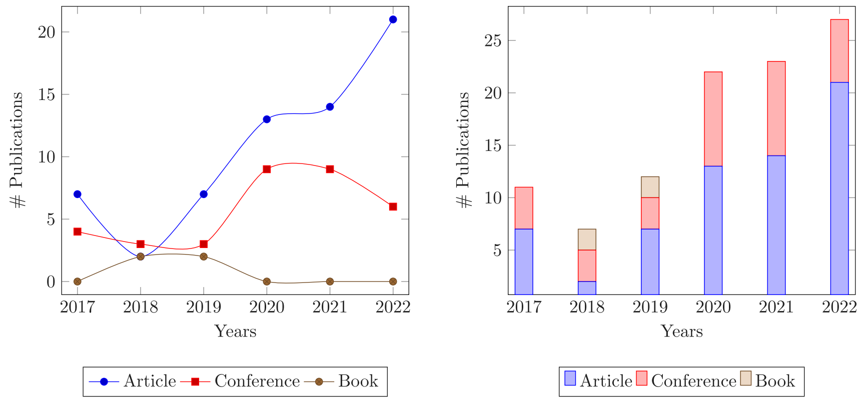


Fig. 8. Distribution of publications per type over the years.

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 non-functional 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 Analysis (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

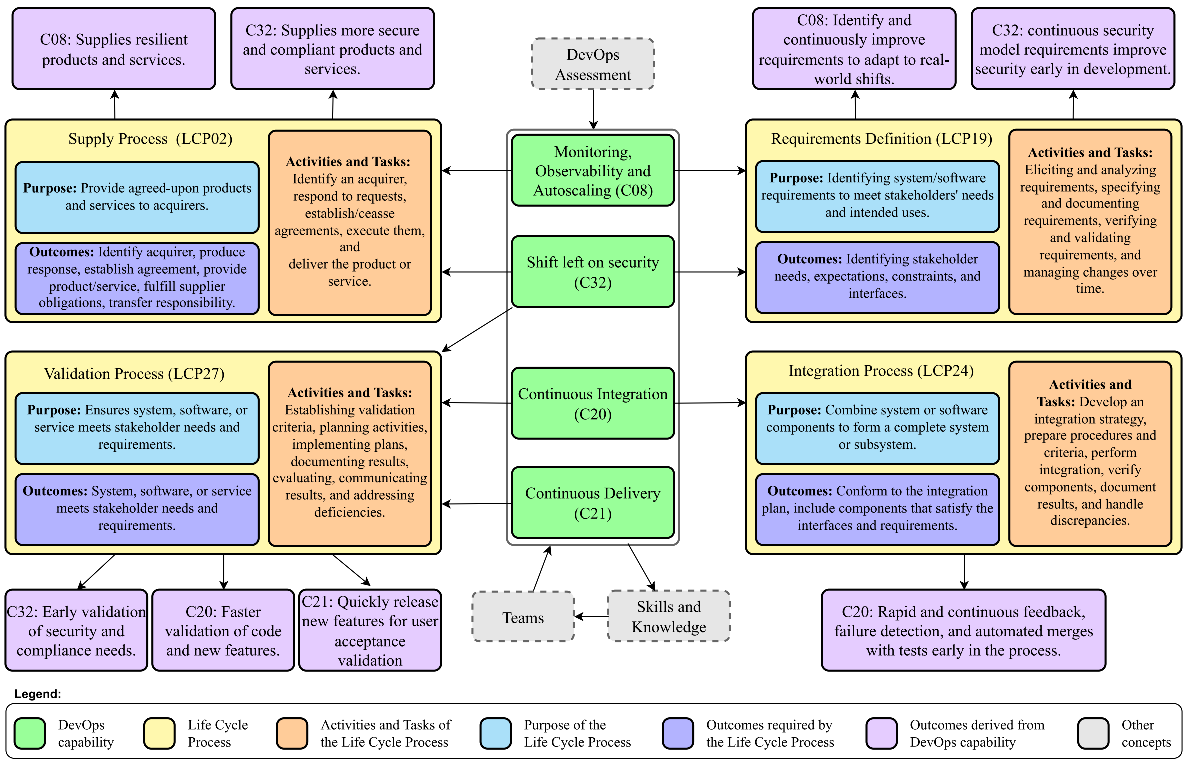

C08: Supplies resilient products and services.
C32: Supplies more secure and compliant products and services.
DevOps Assessment
C08: Identify and continuously improve requirements to adapt to real-world shifts.
C32: continuous security model requirements improve security early in development.
Supply Process (LCP02)
Purpose: Provide agreed-upon products and services to acquirers.
Outcomes: Identify acquirer, produce response, establish agreement, provide product/service, fulfill supplier obligations, transfer responsibility.
Activities and Tasks: Identify an acquirer, respond to requests, establish/ceasse agreements, execute them, and deliver the product or service.
Monitoring, Observability and Autoscaling (C08)
Shift left on security (C32)
Continuous Integration (C20)
Continuous Delivery (C21)
Requirements Definition (LCP19)
Purpose: Identifying system/software requirements to meet stakeholders' needs and intended uses.
Outcomes: Identifying stakeholder needs, expectations, constraints, and interfaces.
Activities and Tasks: Eliciting and analyzing requirements, specifying and documenting requirements, verifying and validating requirements, and managing changes over time.
Validation Process (LCP27)
Purpose: Ensures system, software, or service meets stakeholder needs and requirements.
Outcomes: System, software, or service meets stakeholder needs and requirements.
Activities and Tasks: Establishing validation criteria, planning activities, implementing plans, documenting results, evaluating, communicating results, and addressing deficiencies.
Integration Process (LCP24)
Purpose: Combine system or software components to form a complete system or subsystem.
Outcomes: Conform to the integration plan, include components that satisfy the interfaces and requirements.
Activities and Tasks: Develop an integration strategy, prepare procedures and criteria, perform integration, verify components, document results, and handle discrepancies.
C32: Early validation of security and compliance needs.
C20: Faster validation of code and new features.
C21: Quickly release new features for user acceptance validation
Teams
Skills and Knowledge
C20: Rapid and continuous feedback, failure detection, and automated merges with tests early in the process.
Legend:
DevOps capability
Life Cycle Process
Activities and Tasks of the Life Cycle Process
Purpose of the Life Cycle Process
Outcomes required by the Life Cycle Process
Outcomes derived from DevOps capability
Other concepts

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.

CHAPTER 7

Article #6

The final article (A6) explores the adoption of DevOps in a private software company by observing the factors, challenges, and strategies, and identifying significant relationships between capabilities, metrics, and Life Cycle Processes (LCPs). Information about strategies and capabilities used, challenges and benefits obtained, and ways of measuring them has been provided in the article. The findings are useful for better understanding the application of DevOps capabilities and practices in a real-world environment and obtaining helpful information for applying these practices in other organizations.

As far as the results and findings of the study are concerned, the study identifies 38 symbiotic relations between capabilities, metrics, and lifecycle processes in DevOps adoption. Moreover, the importance of a self-service repository management to increasing the internal efficiency and accountability of the company is clear. It is also noted the central role of the internal catalog in improving internal efficiency and accountability with microservices management. While adopting DevOps, the organization states that releases increased, while escaped defects in production reduced, confirming significant benefits in software quality and operational efficiency. Cloud-native and open-source technologies were beneficial to infrastructure capabilities, while enhanced support for microservices was a strong factor in the case. The conclusion confirms that product prioritization, information, automation, standardization, and visibility are supportive factors for DevOps adoption.

Article details:

- Title: Exploring DevOps Success: A Case Study on Key Capabilities, Metrics, and Lifecycle Processes
- Date submitted: September 2024
- Journal: Journal of Systems and Software
- Scimago Journal Rank: Q1
- Publisher: Elsevier B.V.

Exploring DevOps Success: A Case Study on Key Capabilities, Metrics, and Lifecycle Processes

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

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

Abstract

In the **context** of Information Technology (IT), companies are adopting DevOps to meet customer demands and to improve collaboration, among teams, and streamline software delivery. However, some organizations still find implementation challenges.

The **objective** is to examine the factors influencing the adoption of DevOps in a software company, collecting challenges, and strategies to ensure a successful transition. These factors are DevOps capabilities, metrics, and lifecycle processes. Qualitative research **methods** drive this case study, incorporating analytical methodologies, triangulation, with multiple data sources, like interviews, focus groups and document analysis, for validity and reliability.

Results include 38 symbiotic relations between capabilities, metrics, and lifecycle processes, as well as 16 key initiatives, benefits, and challenges. Self-service repository management improves efficiency and accountability. Internal catalog is crucial for evaluating maturity and managing (micro)services. Releases increased, while escaped defects and Mean Time To Recover/Restore (MTTR) reduced. Cloud native, and open source, improves efficient infrastructure management.

In **conclusion**, product prioritization, information visibility, automation and standardization are validated as key to support a successful adoption of DevOps. The findings enrich the body of knowledge, providing organizations meaningful recommendations for implementation.

Keywords: DevOps, Metrics, Performance, Adoption, Software, Information Systems

1. Introduction

In the current industry context, IT organizations are adopting the DevOps approach to meet market, customer, and internal needs [10, 29, 30, 34]. DevOps is becoming more popular in the software business because organizations want to improve software development and its delivery [21, 27, 39, 48]. Traditional silos between software development and operations caused engineering inefficiencies, delays, and restricted team cooperation. foundational to DevOps [19, 30]. By solving the gap between development and operations, the adoption of DevOps aims to improve software delivery processes and rectify these long-standing issues [6, 43]. The significance of this movement in the industry is evident in the following aspects: Firstly, DevOps empowers organizations to accelerate their time to market through a focus on automation and streamlined workflows. This rapid software release cycle enables a company to quickly adapt to market demands, thereby securing a competitive edge [27]. Second, DevOps fosters improved collaboration and communication between cross-functional teams, breaking down silos, and promoting a culture of transparency and accountability [6, 36]. Furthermore, DevOps capabilities and metrics [3–5] contribute to improved software quality and stability

[*]Corresponding author
Email addresses: ricardo_amaro@iscte-iul.pt (Ricardo Amaro), ruben.filipe.pereira@iscte-iul.pt (Rúben Pereira), mms@tecnico.ulisboa.pt (Miguel Mira da Silva)

through Continuous Integration & Delivery or Deployment (CI/CD), automated testing, and monitoring, resulting in increased velocity and greater customer satisfaction [48]. Finally, DevOps encourages continuous improvement by gathering and analyzing data throughout the software lifecycle. This allows businesses to make data-informed decisions and iterate on procedures and capabilities [22]. Overall, because DevOps improves software quality, time to market, efficiency, and continuous improvement, the software industry considers it essential. By embracing DevOps principles and practices, businesses can navigate the evolving landscape, deliver high-quality software products efficiently [13], and meet the ever-changing demands of customers and markets [10, 29]. But how are companies adopting DevOps metrics and capabilities? What obstacles do they face, and what variables help DevOps adoption?

Research Problem: Despite the growing consensus on DevOps as a key factor in improving software development and efficiency, there is still a lack of empirical evidence on how already existing capabilities [3] and metrics [5] influence the successful adoption of DevOps in software organizations. Additionally, the lack of comprehension of the subtle effect of these factors on the software lifecycle processes [22] prevents organizations from implementing DevOps effectively. As a result, more research is needed to determine the specific capabilities and metrics that contribute to successful DevOps adoption, as well as how these practices impact different stages of the software development lifecycle.

Objective: To address these shortcomings, this study aims to identify and analyze DevOps capabilities and metrics that significantly impact DevOps adoption in a software company. This research uses a case study to investigate deeper into how these factors affect software lifecycle processes in practice. This narrower approach allows the research to provide actionable insights that can help organizations prioritize and implement DevOps practices most likely to succeed. Organizations can then improve their efforts and increase the probability of adopting DevOps successfully.

The research questions in Table 1 can assist in identifying the factors that influence DevOps adoption in a software company. By addressing these research questions, the case study aims to provide insight into

Table 1: Research questions

RQ1	What DevOps capabilities, metrics and Lifecycle processes play a key role in enhancing DevOps adoption?
RQ2	Are there any significant correlations in DevOps capabilities, metrics, and lifecycle processes?
RQ3	How are key strategies, or initiatives, facilitating the adoption of DevOps?
RQ4	During the adoption of DevOps, which key benefits and challenges stand out and why?

the critical aspects of successful DevOps adoption, how DevOps capabilities and metrics affect successful adoption by examining how these aspects connect to various activities in the software lifecycle.

Article structure: Section 1 is the introduction, in Section 2 the research background and related work review is presented together with a proposed conceptual framework, Section 3 describes the research methodology, Section 4 presents the results, in Section 5 a discussion is done, and Section 6 concludes with final remarks, limitations, and future work.

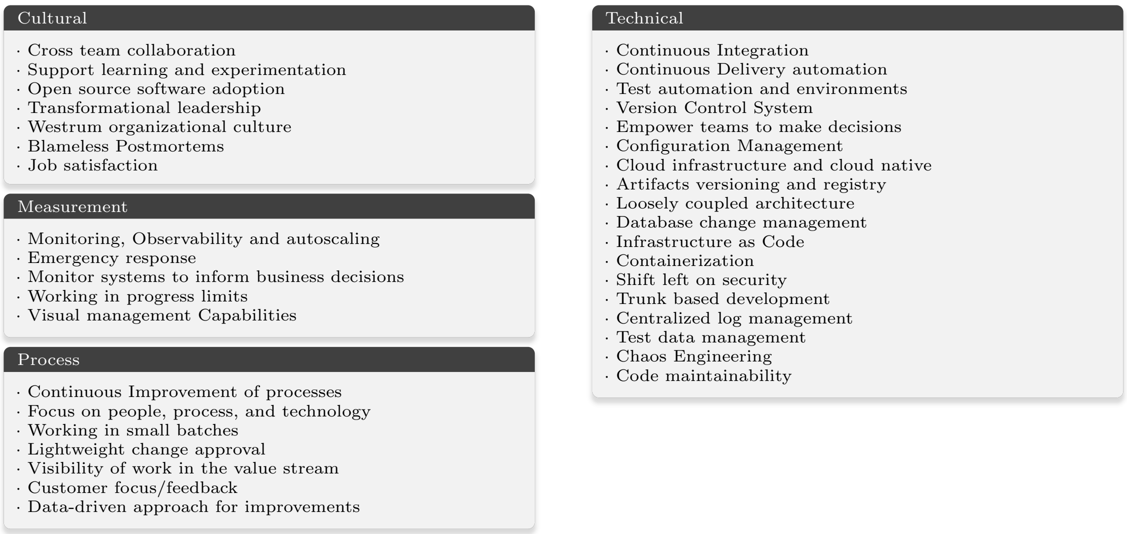
2. Background and Related Work

The framework for this DevOps adoption case study is based on a wide range of sources, including those from software development, operations, and management. Examines the adoption of DevOps, its benefits, challenges, and impact on software companies. With a clear guiding structure, the provided framework helps to understand and explain the specific phenomenon and problem being addressed.

2.1. DevOps Capabilities and Metrics

DevOps refers to the collaboration between Developers (Dev) and Operations (Ops) teams to eliminate engineering silos, shorten the development lifecycle, deliver high-quality software continuously and improve reliability [4, 43]. While DevOps had no universal definition, the IEEE DevOps Standard 2675 (2021) [22, p. 23] brought a definition, stating that DevOps methods promote stakeholder communication and collaboration to develop, build, improve, and operate software and systems products and services. According to Amaro et al. [4], DevOps emphasizes empathy and cross-functional collaboration in software development teams[11]. The components of DevOps are often summarized as CALMS: Culture, Automation, Lean, Measurement, and Sharing[21]. DevOps evolved from agile IT service management [9], focusing on new processes to deliver fast-changing software [6, 21].

This study builds on top of two previous Multivocal Literature Reviews (MLRs), and an Systematic Literature Review (SLR) in Section 2.3, published by the authors [3, 5], following the guidelines proposed by Garousi et al. (2019) [16] and Kitchenham et al. (2007) [26], where the capabilities and metrics of DevOps were found, synthesized and discussed, followed by a Design Science Research (DSR) which validated the capabilities and metrics, while proposing an evaluation artifact [4] showing DevOps capabilities influencing main metrics. The grand objective of the preceding research, which this work continues, is to find, explore and validate the key factors that improve DevOps adoption success. The authors identified the need for a via case study, given the lack of empirical evidence in exploring successful DevOps adoption in a software organization. As other previous authors have mentioned, understanding DevOps capabilities and metrics in practice is crucial for DevOps adoption [3, 5, 14, 38, 43, 45]. Since the purpose of this investigation is to explore how these concepts contribute to overcome researched challenges (Section 2.3) leading to a successful implementation of DevOps in a software company, this study uses the DevOps capabilities vector researched in the previous literature review [3], as shown in Figure 1. Similarly, while DevOps capabilities are



Cultural

· Cross team collaboration
· Support learning and experimentation
· Open source software adoption
· Transformational leadership
· Westrum organizational culture
· Blameless Postmortems
· Job satisfaction

Measurement

· Monitoring, Observability and autoscaling
· Emergency response
· Monitor systems to inform business decisions
· Working in progress limits
· Visual management Capabilities

Process

· Continuous Improvement of processes
· Focus on people, process, and technology
· Working in small batches
· Lightweight change approval
· Visibility of work in the value stream
· Customer focus/feedback
· Data-driven approach for improvements

Technical

· Continuous Integration
· Continuous Delivery automation
· Test automation and environments
· Version Control System
· Empower teams to make decisions
· Configuration Management
· 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
· Code maintainability

Figure 1: Categorization of DevOps Capabilities, adapted [3].

fundamental, to gain a comprehensive understanding of DevOps adoption, it is equally important to measure and monitor various metrics that reflect DevOps performance [21, 41]. As part of the authors' research, the DevOps metrics were also elicited, synthesized and discussed in a recent literature review by Amaro et al. [5], these investigated metrics were also categorized to provide practical evidence of the effectiveness in DevOps adoption. In the previous research, it was identified that future work should address how DevOps capabilities relate to metrics in a real life situation, which is being addressed in this article. Figure 2 provides a detailed categorization of these metrics [5].

Both DevOps capabilities and metrics, as shown in Figures 1 and 2, provide valuable material for discussing the adoption of DevOps in this case study.

Change

- Mean Lead-time for Changes (MLT)
- Deployment Frequency (DF)
- Change Failure Rate (CFR)
- Deployment Duration Time
- Defect Escape Rate
- Cycle Time Value (CTV)
- Deployment Size
- Mean time to Failure (MTTF)
- Pipeline Automated Tests Success Rate
- Automated Test Code Coverage

Business

- Customer Tickets Volume and Feedback
- Customer Usage and Traffic

Operational

- Mean Time To Recover/Restore (MTTR)
- Service Availability and Uptime
- Mean Time To Detection (MTTD)
- Application Response Time
- Service Level Agreements (SLAs) and Objectives (SLOs)
- Production Error and Incident Rate
- Work in Progress (WIP) /Load
- Unplanned Work Rate (UWR)
- Wait Time

Cultural

- Westrum Organizational Culture Measures
- Team Happiness
- Talent retention

Figure 2: Categorization of DevOps Metrics, adapted [5].

2.2. DevOps Life Cycle Processes

The ISO/IEC/IEEE 12207:2017 standard provides a reference model for structuring the software life cycle into different processes, known as Software Life Cycle Processes (LCPs). It addresses the requirements of software engineering approaches, accommodating the incremental and iterative nature of DevOps in IEEE Standard 2675-2021 [22]. The standard establishes common terminology and serves as a reference for activities like process definition, modeling, and assessment within its 30 LCPs, as seen in Figure 3. The framework categorizes DevOps capabilities into cultural, measurement, process, and technical domains, and links them to specific metrics and LCPs as seen in Figure 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. This life cycle framework of processes and activities is concerned with the all life cycle, acting as a common reference for communication and alignment between stakeholders [46].

Organizational project-enabling processes

- Life Cycle Model Management Process
- Infrastructure Management Process
- Portfolio Management Process
- Resource Management process
- Quality Management Process
- Knowledge Management Process

Technical management processes

- Project Planning Process
- Project Assessment and Control Process
- Decision Management Process
- Risk Management Process
- Configuration Management Process
- Information Management Process
- Measurement Process
- Quality Assurance Process

Agreement processes

- Acquisition
- Supply

Technical processes

- Business or Mission Analysis Process
- Stakeholder Needs and Requirements Definition Process
- Systems/Software Requirements Definition Process
- Architecture Definition Process
- Design Definition Process
- System Analysis Process
- Implementation Process
- Integration Process
- Verification Process
- Transition Process
- Validation Process
- Operation Process
- Maintenance Process
- Disposal Process

Figure 3: Software Life cycle processes, adapted [46].

2.3. DevOps Adoption Benefits and Challenges

In another previous Systematic Literature Review (SLR) done by the authors, following the guidelines proposed by Kitchenham et al. [26], the benefits of DevOps were collected and synthesized [13]. A research protocol, inclusion and exclusion criteria, using well-known scientific databases, and search strings were defined. Inclusion and exclusion criteria were based on language (English and Portuguese), only scientific publications (journals, conferences, and books) and publication date (after 2008). In the initial analysis of the 98 publications found, a noticeable lack of synthesis of DevOps benefits was seen. The authors then performed a second literature review in which an additional 36 publications related to case studies were found, analyzed, and synthesized. This second review was able to consolidate the benefits found, map these challenges faced by organizations, and gather empirical evidence to support the findings, aimed to provide practitioners with a clear understanding of what to expect when adopting DevOps. This was crucial to providing a comprehensive overview of the benefits and challenges seen in Tables 2 and 3 associated with DevOps adoption. Furthermore, some case studies reviewed did not present any benefits, indicating a gap that the authors identified as a potential area for future work. The combination of this research with a case study in which adoption challenges are addressed will help new DevOps practitioners clarify what is expected to be achieved with DevOps and how to go about its successful adoption.

Several studies have explored DevOps adoption [1, 28, 48], revealing its capabilities, benefits, challenges, and impacts. Waseem et al. [51] conducted a systematic mapping study on DevOps, highlighting current research in this area. Hemon-Hildgen et al. [20], Lévy et al. [31], Perez et al. [37] presented case studies on DevOps adoption in several organizations, clearly highlighting implementation benefits, but also adoption challenges.

Understanding DevOps benefits and challenges is essential before conducting this case study, since the benefits lead to DevOps adoption and set a standard for success [1, 38, 45]. The challenges provide insight into potential obstacles during adoption [39, 48, 49]. Mitigating risks and managing expectations can be achieved by incorporating these challenges into DevOps adoption strategies [2, 12, 34]. To summarize, according to the author's previous Systematic Literature Review (SLR) [13], tables 2 and 3 list a number of challenges and benefits that were validated by the 36 adoption cases.

Table 2: Benefits of DevOps Adoption

Cross-team collaboration and Efficiency	Cross-team collaboration and communication, faster time to market, faster and better feedback loops, and increase of team performance. Often lead to faster development cycles and delivery performance.
Quality and Reliability Improvements	Increase of code quality, improvement of system reliability, and less mean time to recover. Contribute to the overall robustness and reliability of products.
Cost Efficiency, Automation and Delivery Improvements	Costs reduction, decrease of manual work, less failed changes, and better deployment management. Relate to operational efficiencies and cost benefits.
Value, Culture and Customer Satisfaction	The increase in value, organizational culture, and customer satisfaction is related to organizational shifts toward a DevOps culture, which puts customer satisfaction first.
Innovation and Skill Maximization	More innovation, maximization of competences, and processes and tools standardization
Security and Motivation Improvements	Less security issues and increase of employees' motivation. Shift left security techniques boost confidence and empowerment, employees are more motivated.

The literature Table 2 highlights the prevalent benefits of DevOps adoption in software development. Research suggests that DevOps adoption can be challenging (Table 3), despite its many benefits.

Finally, knowing the benefits and challenges of adopting DevOps gives a full picture of what to expect when a software company starts using it. Benefits show possible improvements, and challenges show possible problems. Software companies can use this information to make smart decisions, come up with good DevOps adoption plans, and add to the body of DevOps knowledge.

2.4. Conceptual Framework

The conceptual framework for this study illustrated in Figure 4 is derived from the research background and related work review done in this section and is designed to show the relationships between the different variables in the study. This will help in understanding and interpreting the specific phenomenon and problem

Table 3: Challenges of DevOps Adoption

Industry Constraints	Industry-specific restrictions can hinder DevOps adoption. This challenge requires flexible approaches that consider industry-specific laws, standards, and safety measures to establish appropriate security and compliance controls in the DevOps pipeline.
Deep-Seated Company Culture	Dysfunctional culture can hinder DevOps adoption. The biggest obstacle is an embedded company culture that rarely changes and requires cultural transformation to support development operations.
Insufficient Communication	The lack of effective communication within teams, can severely impact the DevOps adoption process, emphasizing the importance of improving communication channels and practices.
Lack of Clarity	The implementation, dissemination of knowledge across teams and geographic dispersion among team members, indicates a crucial need for education and clear articulation of DevOps capabilities accross the organization.

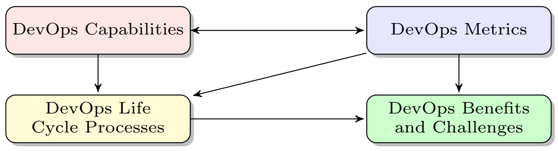


Figure 4: Proposed Conceptual Framework of DevOps Adoption.

being addressed. The proposed framework is categorized into four main components: DevOps Capabilities, DevOps Metrics, DevOps LCPs, leading to DevOps Benefits and Challenges. The relationships between these components are explained in Table 4. The conceptual framework guides the case study as it looks at

Table 4: Relations in the Conceptual Framework of DevOps Adoption

DevOps Capabilities	Influence both the **DevOps Metrics** and **DevOps Life Cycle Processes** [3, 22, 30, 48]. These capabilities, which are categorized into Cultural, Measurement, Process, and Technical, provide the necessary skills and tools for implementing and managing DevOps capabilities [3].
DevOps Metrics	Provide the means to measure the effectiveness and impact of the DevOps Capabilities and LCPs [4, 8, 38, 49]. These metrics are categorized into Change, Business, Operational, and Cultural [4, 5].
DevOps Life Cycle Processes	As defined by the IEEE Standard 2675-2021[22] standard, are influenced by the DevOps Capabilities and provide the structure and activities for implementing DevOps capabilities[6, 33, 49].
DevOps Benefits and Challenges	All concepts contribute to the outcomes. Benefits provide a clear understanding of the potential improvements that can be achieved through DevOps, while the challenges offer insights into the potential obstacles that may be encountered during the adoption process [11–13, 17, 48, 54].

the current state of DevOps adoption and the role of DevOps capabilities and metrics in making adoption successful in a software company. The arrows in Figure 4 represent the influence and direction of the relationships between the components. The DevOps Capabilities affect DevOps metrics and LCPs. These, in turn, influence DevOps benefits and challenges.

3. Research Methodology

This study uses qualitative research to analyze the contextual and human factors that affect DevOps adoption. The sample consists of 28 semi-structured interviews, archival documents and focus groups from a software company. Data were collected to understand the impact of DevOps on software quality and team collaboration. Thematic analysis was used to map interview data to predefined DevOps concepts of capabilities, metrics and LCPs.

This method covers the subjective experiences, motivations, and challenges of software engineering, a complex and people-oriented field [7]. Case study research [55], specifically, can explore the practical context of DevOps adoption, like organizational and cultural changes. The case shows how DevOps capabilities influence metrics and LCPs in organizations [3, 4, 22]. Case study model can analyze individual, group, organizational or social phenomena. In Figure 5, six phases make up the sequential and iterative investigation, as explained in Table 5.

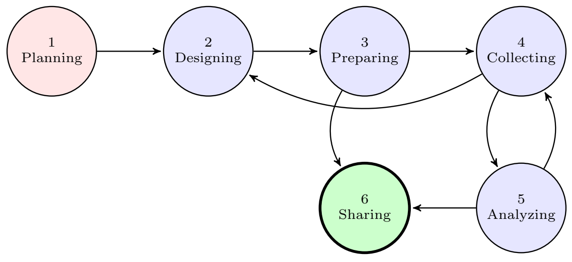


Figure 5: The Case Study Research process, adapted [55].

This method collects data, finds patterns and draws conclusions to better understand DevOps in software engineering. A case study is carried out to evaluate and confirm the performance of DevOps in relation to the concepts presented in the conceptual framework Figure 4. According to Runeson and Höst [42], these are crucial for software development. This case study shows how DevOps affects complex and dynamic software development. The case study research process is both linear and iterative, allowing for improvements in the

Table 5: Six Phases of a Case Study.

1. Planning	Determine the case study relevant situation, compared with other research methods. Know what a case study inquiry is and how it works. Decide whether to do a case study.
2. Designing	Case definition, theory development, and design discovery. Test validity against construct, internal, external, and reliability criteria. Single-case, multiple-case, holistic, and embedded designs are major. Define case study selection, methodologies, and procedures.
3. Preparing	Refine researcher skills, train for specific cases if required, and develop a rigorous case study protocol. Conduct a pilot case and address procedural uncertainties.
4. Collecting	Follow the case study protocol, establish a case study database, and maintain a transparent chain of evidence. Gather data from multiple sources, including interviews, documentation, archival data, and focus groups. Be cautious of social aspects when collecting and presenting data.
5. Analyzing	Structured data is separated from interpretations. A data analysis method like pattern matching, explanation building, time series analysis, logic models, or cross-case synthesis is necessary. Review current research and apply broad strategies to specific methods.
6. Sharing	Meaningful results are presented to ensure completeness and importance. Considerations of alternative viewpoints provide enough evidence for reader interpretation. Focusing on report flow and involvement, and iterating the report-writing process for quality.

design and data collection based on the initial findings illustrated in Figure 5. After the first round of data collection and analysis, adjustments can be made to improve the overall study.

3.0.1. Case Definition

This case study focuses on a private software company, here referred to as the "Company" or "Organization", which operates in the computer software industry. Founded in 2007, the company has established its headquarters in the United States and serves a worldwide market. The company relies on more than 1,200 people and is ranked as a leader in the Gartner's 2020 Magic Quadrant for Digital Experience Platforms[1]. The company offers enterprise products, services, and technical support for Drupal, an open-source Web Content Management Platform[2], through its software-as-a-service offerings. Venture capital backing and industry benchmarks show its rapid growth and transformation.

[1] https://www.gartner.com/reviews/market/digital-experience-platforms
[2] https://drupal.org

3.0.2. Case Study Protocol

This study uses an exploratory case study approach [55] to examine the company's adoption of DevOps. This type of case study examines phenomena with no clear outcomes. A case study approach [42, 55] enables in-depth analysis of complex phenomena in real-life contexts. The exploratory nature of this study allows it to understand DevOps adoption in ways that quantitative methods cannot. Its ability to examine real-world situations makes it suitable for software engineering research.

Construction validity: In this case study, data triangulation is used, interviews are analyzed, and a logical chain of events is defined, using multiple sources of evidence and methods to mitigate the impact of any single interpretation of data and validate the findings[42]. According to several authors [15, 55], data triangulation is essential for evaluating methodological rigor. Yin [55] suggests four types of triangulation: data through multiple sources, researcher through multiple evaluators, theory through multiple perspectives on the same data set, and methodological through complementary methods. Interviews and historical data are used during triangulation to retain high-quality data and teach readers about the study's procedure [15]. Without the ability to replicate a case study, triangulation proves its reliability, as detailed in Section 3.3.

Data Collection and Analysis Procedures: The Figure 7 case study protocol collects data using multiple methods for triangulation: **Semi-structured interviews:** [40] with key personnel involved in the DevOps transition were conducted, along with **archival documents** and metrics [42] like DORA assessments, process descriptions, and performance reports [40]. Engage in **focus group** discussions with multiple teams to gather diverse perspectives and validate findings. From coding and thematic analysis of qualitative data to cross-analysis to identify patterns and correlations, data analysis will follow a systematic approach [15, 35]. All participants received informed consent forms for confidentiality and withdrawal. Data is anonymized to protect individuals and the company. The study follows ethical research standards [42] to ensure transparency and integrity.

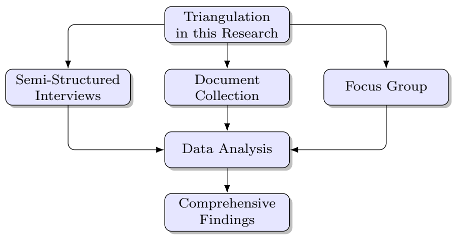


Figure 6: Validation using triangulation of methods and data in this study.

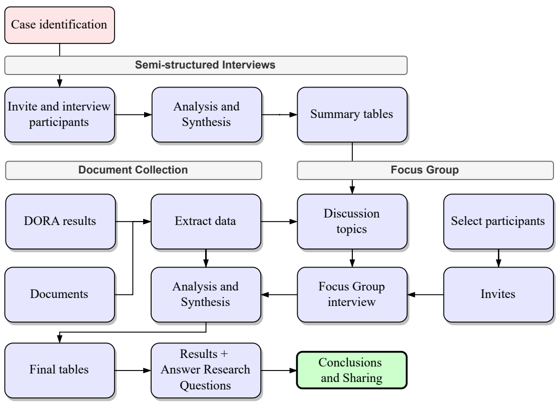


Figure 7: Case Study Protocol Process, adapted from Runeson and Höst [42].

The sequence of research activities, from case identification, data collection, focus groups, and analysis, to study conclusions and dissemination, is visualized by the diagram protocol in Figure 7. The triangulation approach [15], as described in Case Study Protocol, is supported through multiple methods and data sources, which also give a detailed account of the DevOps adoption process within the software company.

3.1. Data Collection

Before data collection, preparation is crucial to ensure relevance, reliability, and validity. This phase involves selecting participants, establishing study objectives, and defining data collection tools.

3.1.1. Semi-structured Interviews

Appendix Appendix A outlines an interview guide with open-ended questions to gather in-depth responses on DevOps capabilities, metrics, lifecycle processes, and correlations. A DevOps-experienced volunteer was interviewed for a pilot. The session's feedback improved interview questions' clarity and relevance to research questions. Semi-structured videoconference interviews were held. All practitioners in the organization could sign up for 45-minute slot interviews at their preferred time and date between June and September 2023 on a webpage. Table 6 lists all participants, a diverse sample of engineers, team leads, managers, and stakeholders who agreed to be interviewed. The invitation letter Appendix B explains the study's purpose and expected contributions of interviewees. The decision to stop conducting semi-structured interviews is because everyone who agreed to be interviewed had been interviewed. The entire process is documented, along with the

Table 6: List of semi-structure interviews

ID	Date	Position	Location	Familiar with DevOps
I01	2023-06-23	Staff Engineer	Canada	More than 3 years
I02	2023-06-23	Indvidual contributor	Canada	More than 3 years
I03	2023-06-27	Team Lead	Canada	Between 1 year and 3 years
I04	2023-06-29	Manager	USA	Between 1 year and 3 years
I05	2023-06-30	Manager	USA	More than 3 years
I06	2023-07-03	Indvidual contributor	India	Between 1 year and 3 years
I07	2023-07-03	Manager	France	More than 3 years
I08	2023-07-04	Indvidual contributor	India	Less than 1 year
I09	2023-07-04	Director	Canada	More than 3 years
I10	2023-07-07	Manager	Canada	More than 3 years
I11	2023-07-07	Team Lead	USA	More than 3 years
I12	2023-07-10	Team Lead	USA	More than 3 years
I13	2023-07-13	Indvidual contributor	India	More than 3 years
I14	2023-07-13	Manager	USA	More than 3 years
I15	2023-07-14	Indvidual contributor	India	More than 3 years
I16	2023-07-14	Indvidual contributor	India	Between 1 year and 3 years
I17	2023-07-14	Indvidual contributor	USA	More than 3 years
I18	2023-07-20	Indvidual contributor	India	Between 1 year and 3 years
I19	2023-07-21	Team Lead	USA	More than 3 years
I20	2023-08-08	Indvidual contributor	Netherlands	More than 3 years
I21	2023-08-09	Indvidual contributor	India	Between 1 year and 3 years
I22	2023-08-11	Director	USA	More than 3 years
I23	2023-08-21	Indvidual contributor	USA	Between 1 year and 3 years
I24	2023-08-21	Team Lead	USA	More than 3 years
I25	2023-08-21	Indvidual contributor	USA	More than 3 years
I26	2023-08-25	Indvidual contributor	India	Between 1 year and 3 years
I27	2023-09-08	Director	USA	More than 3 years
I28	2023-09-08	Director	USA	More than 3 years

interview guide in Appendix Appendix A, the invitation in Appendix Appendix B, the execution of the interviews in Section 3.2, and the analysis of results Section 4, to ensure the transparency and reproducibility of the study.

3.1.2. Document Collection

The documents analyzed are in most part assessments, diagrams, presentations, and performance metrics. All relevant documents are listed in Table 7, constituted of archival data and metrics that were identified, including DevOps Research and Assessment (DORA) assessment reports from 2021 to 2023, flow diagrams related to the CI/CD pipeline, and presentations on DevOps culture, engineering, and automation. Access to these documents was secured with the necessary permissions. A plan was created to analyze the documents. This included creating a template to record the document's purpose, DevOps capabilities, and data or metrics. Data was extracted from documents using systematic data retrieval and analysis. This was achieved by

Table 7: List of relevant documents

ID	Title	Description	Pages	Type	Date
D01	DORA Assessment	DORA Assessment from 2021	17	Assessment	2021
D02	DORA Assessment	DORA Assessment from 2022	17	Assessment	2022
D03	DORA Assessment	DORA Assessment from 2023	17	Assessment	2023
D04	DevOps culture presentation	Journey kickoff presentation from 2020	42	Presentation	2020
D05	DevOps CI/CD pipeline diagram	Transformation leadership Vision	5	Presentation	2021
D06	DORA Presentation	High Performance Software Delivery by DORA	45	Presentation	2022
D07	Pipeline Onboarding status	Progress of DevOps pipeline adoption over time	1	Metrics	2023
D08	Escaped Defects	Escaped Defects over DevOps adoption time	1	Metrics	2023
D09	Release success rate	Release Success Rate over DevOps adoption time	1	Metrics	2023

employing QualCoder[3] for thematic coding and other Free/Libre/Open Source Software (FLOSS) [32], such as Zotero[4], Calc Spreadsheets[5] and Python[6] with Pandas[7], to facilitate data manipulation.

3.1.3. Focus Group

The focus group validated findings and gathered diverse perspectives. Focus groups are a valuable qualitative research method that can be used in case study research. They are particularly helpful in analyzing complex issues since they provide insightful information on the experiences, perceptions, opinions, beliefs, and attitudes of the participants [15, 55]. Focus groups, a form of triangulation, improve the credibility and validity of case study findings by allowing researchers to verify data from various sources, such as interviews, observations, and document analysis [35]. This methodological triangulation strengthens the research outcomes' consistency and robustness, making them more reliable. The literature on qualitative research methodologies, notably Denzin's triangulation, suggests that focus groups can reduce biases and improve study understanding [15]. The design of the focus group sessions followed the guidelines provided by McDonagh [35] and Teixeira et al. [48]. The focus group was designed for open discussions. The session covered DevOps adoption topics, with questions and suggestions that encouraged discussion. A moderator was selected based on experience in facilitating group discussions and knowledge of DevOps capabilities, metrics, and LCPs. Seven participants were selected to represent a range of roles and experiences with DevOps within the Company listed in Table 8. Invitations were sent with clear information about the session's objectives and the importance of their contributions. As in the interviews, the focus group participants received

Table 8: List of focus group panel participants

ID	Date	Position	Location	Familiar with DevOps
G01	2023-10-27	Indvidual contributor	India	Between 1 year and 3 years
G02	2023-10-27	Manager	USA	More than 3 years
G03	2023-10-27	Manager	USA	Between 1 year and 3 years
G04	2023-10-27	Director	USA	More than 3 years
G05	2023-10-27	DORA Expert	USA	More than 3 years
G06	2023-10-27	Manager	USA	More than 3 years
G07	2023-10-27	Team Lead	Canada	Between 1 year and 3 years

informed consent forms, and the confidentiality of the discussions was emphasized [42]. With participant consent, the focus group was recorded and thorough notes were collected to capture the discussions. Ethics and data protection were followed when recording and taking notes. Results were analyzed by recording

[3]https://github.com/ccbogel/QualCoder
[4]https://www.zotero.org/
[5]https://www.libreoffice.org/discover/calc/
[6]https://www.python.org/
[7]https://pandas.pydata.org/

discussions, coding data with QualCoder, and using spreadsheets to find patterns and themes. This was done with a summary of cross-referenced interview and document findings and focus group observations and recommendations.

3.2. Data Analysis

To analyze the interview data, thematic analysis of the interview data was used with a coding scheme based on research questions, the framework of Section 2 and the themes that emerged from the data. Analysis was performed using FLOSS tools freely available, such as QualCoder, Zotero and LibreOffice. The results of the qualitative analysis of interview data, focus group and documents reveal significant patterns and insights presented in Section 4 to answer the research questions. Coding was done involving thematic analysis of interviews and focus group transcripts. Using pattern matching and direct questions, correlations were identified between DevOps capabilities, metrics, and lifecycle processes.

This case used data from various sources, as shown in this section.

3.2.1. Semi-structured Interviews

A total of 28 semi-structured interviews were conducted with various people, covering the spectrum of the key roles involved in the DevOps adoption process in the company. The sample includes engineers, team leads, managers, and other stakeholders. One of the authors conducted interviews. The 45-minute interviews sought to obtain diverse perspectives on DevOps adoption challenges, benefits, along with strategies outlined in Section 3.1 and Appendix Appendix A. Data were collected through semi-structured interviews with permission to take notes and record. The decision to stop conducting semi-structured interviews is based on research questions, sample size, and exhaustion of interviewees, which are also explained in this section. Next, a short summary of each interview is presented, accompanied by the identification of some codes extracted from the text, analyzed, for which the results are presented in more detail in Section 4.

***I01*:** The first interviewee emphasized ”the benefit seen from the CI/CD pipeline implementation on software teams”, including enhanced collaboration (C01) and standardization in the Software Development Life Cycle (SDLC). With new teams joining DevOps, capabilities have grown. The interviewee evaluated various capabilities and metrics, indicating varying levels of success in areas such as *Continuous Integration* (CI) (C20), configuration management (C25), and centralized log management (C34). DevOps automates processes to improve team happiness, deployment duration, and recovery time (M21, M04). DF (M02) was identified as a key metric for improvement. It was stated that collaboration standardizing DevOps tooling improved delivery, and cost savings despite team dependencies, and resistance to change (C21,L02,M02).

***I02*:** The second interview covered stack-wide networking and code deployment. We test, compile, build, and deploy packages in L28 test and production environments. The interviewee favored the new CI/CD pipeline for its simplicity, structure, and detailed documentation (L08). DevOps metrics like monitoring, observability, and engineering notifications to address issues early (C08,M09). The interviewee also highlighted ”the need for a better knowledge management and observability” as automation increases, to ensure trust in automated processes (C12, L08). The team member concluded with DevOps' pros and cons, including (M10) automated testing and deployment pipeline confidence.

***I03*:** In I03 DevOps adoption discusses moving from Jenkins struggles for CI (C20) and deployment (C21) to automated processes with a dedicated team (C21). This change helps to code, to tests, and to deploy automatically (M10). DevOps capabilities (C01, C02) and metrics (M01, M02, M03, M11) are examined for their impact on the software lifecycle (L08), with a certain focus on experimentation (C02), cross-team communication (C01), and proactive monitoring (C08). The interviewee also notes the importance of DevOps in supporting a learning culture (C02), improving job satisfaction (C07), and keeping up with best practices. Acknowledged that ”migrations and training are challenges”, along with benefits such as efficiency, automation (C21) and alignment with the latest methodologies.

***I04*:** The fourth interview was about the adherence to DevOps in the organization, and the implementation of the CI/CD pipeline (C20, C21) as well as tools such as Jfrog (C32) and SonarQube (C22). The main obstacles include ”resistance to change and a lack of understanding of DevOps as a culture rather than a team”. The benefits of adopting DevOps are noted as consistency, scalability, and improved workflows

(C13), which facilitate team transitions and issue diagnosis. The importance of visual management (C12) and data analysis (C19) in enhancing organizational capabilities. However, there are struggles with cross-team collaboration and communication (C01), as well as a "fear of committing to dates", which hinders several definition and quality processes (L07,L18,L19). Finally, a cultural shift towards continuous improvement (C13) and a more constructive approach to addressing workflow inefficiencies is needed.

***I05*:** Organizational culture (C05), practices, and issues were discussed in the fifth interview. Participant noted progress in "breaking down the silos" and improving collaboration (C01), but complex projects with multiple teams require more growth (C14). After years of focus, DevOps improved job satisfaction (C07). He also praised the company's (C03) open source software adoption. After reviewing measurement practices, centralized logging (C34) and monitoring tools were implemented, but automated problem detection and notification (C09) needed improvement. For sprint management, teams with (C11) WIP limits were advised. Finally, the interviewee highlighted the importance of visualizing dependencies (C12) and monitoring software versions and usage (M24), acknowledging recent improvements, but recognizing the need for ongoing progress.

***I06*:** In the sixth interview, it was mentioned that the company uses open source software and is DevOps-focused. The implementation requires collaboration and communication (C01). Applications are "closely monitored to ensure quality and customer satisfaction" (C08, M18). The interviewee highlighted deployment procedures (C21), continuous integration (C20), and continuous delivery (M02) improvements in DevOps capabilities and practices. Additional topics discussed included talent retention (M22), project planning (L09), and information management (L14). The interviewee emphasized the need of Agile methods and information management (L14) for cross-team cooperation. Addressed both the challenges in transferring applications to the CI/CD pipeline (C21) and the benefits of open communication and knowledge sharing (L08). The ideas of the participant shows that the DevOps capabilities and metrics can enhance IT.

***I07*:** The DevOps expert talks about the company's DevOps progress in the seventh interview. The participant evaluated DevOps resources and metrics, finding good progress in the CI/CD pipeline (C20, C21) and familiarity with containerization (C31). Explaining the vision and purpose of these changes to the team could be improved (C04). The participant notes that open source tools (C03) and proactive monitoring (C08) are widely used, but sometimes the need for them is unclear. Participant also mentions the use of a platform (L04) for better observation and decision-making (M12, M14). Challenges include resistance to change and the need for better communication and explanation of DevOps capabilities and culture. The participant suggests that "with time and as the benefits become clearer, the adoption should increase from its current level".

***I08*:** The engineer discusses "DevOps evolution" and how it "affects the software lifecycle". DevOps capabilities and metrics are evaluated, with cross-team collaboration and communication (C01) scoring well due to significant improvements. Also, highly rated are proactive monitoring (C08) and deployment automation (C21), indicating their importance in DevOps. The participant suggested visual aids (L12) to help new team members learn DevOps. Participants' evaluations and comments statec that DevOps capabilities improve collaboration (C01), monitoring (C08), and automation (C21), but they also highlight areas that need more clarity and communication. Assessing the effectiveness of these practices requires metrics such as DF (M02) and mean lead time for changes (M01). Quality Management (L07) and Architecture Definition (L20) help DevOps adoption continuously.

***I09*:** Interview nine discusses SDLC and DevOps metrics. The interviewee said team collaboration and communication (C01) is moderately effective, "showing progress but room for improvement". Containerization (C31) and release automation (C21) are growing in importance. Proactive failure notification (C09) and emergency response are also effective. The participant highlights the significance of visual management capabilities (C12), featuring the Cortex[8] tool that significantly exposes operations. Although continuous improvement processes (C13) have been well received, smoother change control is needed. Key metrics for operational efficiency include deployment size (M07), CFR (M03), and automated test code coverage (M10). Standardization is essential for collaboration and security, but prioritization, visibility, and standardization are major issues. The interview is optimistic but cautious about the adoption of DevOps, with some well-implemented capabilities and others for improvement.

[8]https://cortex.io

***I10*:** Interview 10 explores how metrics affect software lifecycles. In DevOps culture, people work together and talk to each other (C01) The organization uses open source software (C03) because it is open source. Variant levels of certainty among the teams (M21) regarding the implementation and testing of ideas, indicate the "necessity for a more uniform culture". Emergencies and proactive failure notification need improvement (C09). Utilizing tools like CNCF[9] and ArgoCD[10] can enhance continuous process and workflow" improvement (C13). Participant praises emphasis on people, process, and technology (C14) and change approval process improvements (C16). Onboarding and training for CI/CD tools, operational standards, validation (L27), and understanding change downstream are discussed. Adopting DevOps benefits from automation and security (C32).

***I11*:** The interviewee assessed the adoption and capabilities of DevOps in the company. Cross-team collaboration and communication (C01) were highly evaluated, stating a "strong culture of cooperation", but "information is still scattered". Positive support for learning, culture, and experimentation (C02) indicated an innovative environment. The DevOps stack and CI/CD (C20, C21) use open source extensively. Despite better implementation, package management issues with low visibility and governance over certain repositories and packages hampered developer operations (L23). While observability tools were in place, project health metrics (M01) and proactive monitoring (C08) improved. Critics cite emergency response (C09) security issues. Participant highly rates DevOps capabilities like CI (C20), deployment, test automation (C22), and version control (C23). Team empowerment (C24) and loosely coupled architecture (C28) were appreciated, but talent retention (M22) and culture were concerns. According to the interview, DevOps adoption has "strong technical capabilities and a focus on process improvement", but governance, monitoring, and organizational culture need improvement.

***I12*:** Team collaboration (C01) was the most impactful aspect of DevOps in the next interview, as it maintains product confidence and guides issue resolution. The coordination of internal services and third-party vendors can create challenges (C03). The CI/CD pipeline innovations encourage learning and experimentation (C02, C20, C21). Open source software is popular and constantly improved (C03). Positive transformation leadership contributes to a generative culture, but "some processes can be frustrating" (C04). Nice monitoring and observability, but poor proactive monitoring (C08). Improving DF (M02), size (M07), and defect escape rate (M05) requires ongoing integration and deployment. Implement security shift-left (C32) "without slowing development". The interview emphasizes DevOps for service availability (M12), quality (L16), and operations (L28).

***I13*:** The interviewee stresses DevOps collaboration and communication for team and organizational success (C01). Open-source Kubernetes and Docker are needed to test new methods. Leadership transformation is essential for team direction (C04). The participant (C05, C06) said "blameless postmortems and DevOps culture are encouraging communication and innovation to help learning from production incidents". Monitoring and autoscaling ensure operational stability and a relaxing on-call experience (C08). Proactive failure notifications are necessary to maintain SLA (M15). Monitoring system usage may reduce costs, but the impact is unknown (M24). Visual management and work in progress limits help to improve focus (C11, C14). DevOps emphasizes continuous delivery and integration. Automated and data-driven testing, team empowerment, configuration management and customer focus (M23). The interview (M02, M03) highlights high DF and low CFR in implementation, log management and chaos engineering. The importance of MTTR in production is evident (M11). It was stated last, that "DevOps simplifies processes, reduces errors and allows you to focus on new automation instead of manual work" (L09-L23).

***I14*:** Next, a six-year DevOps veteran discusses the shift from manual quality assurance to a strong culture. The senior engineer emphasized the "shift from reactive to proactive" quality measures like test automation (C22) and shift-left, growing the quality team from three to over 20 engineers. The implementation of DevOps has led to increased DF (M02), improved incident rates (M16), and reduced defects (M05). Manual errors are "reduced by our automation, as speed improved, along efficiency, and collaboration" (C01). DevOps participant emphasizes executive support for cultural change (C04), learning, and experimentation. Continuous integration (C20) enhances quality and efficiency. Automation increases job satisfaction and

[9]https://www.cncf.io
[10]https://argoproj.github.io

helps teams solve complex problems (C07). DevOps enhances customer experiences by tackling issues faster and boosting service reliability and stability. (M16).

***I15*:** The interview stressed the significance of DevOps adoption on Software Life Cycle Process (SLCPs), emphasizing collaboration, communication, and proactive monitoring (C01, C08). The engineer interviewed reported improved DF (M02), incident management, and quality assurance (L16) with automation. Adopting CI/CD (C20,C21) has helped maintain service availability (M12) and team happiness (M21). The participant also notes the "importance of blameless postmortems" (C06) in reducing fear of failure and fostering a culture of continuous improvement (C13). The challenges faced include "adapting to new tools and the need for better documentation" to facilitate the implementation of the quality management process (L07, C22, M09, M10, M05), where the service catalog is helpful. In general, the participant highly evaluates various DevOps capabilities and metrics, indicating the critical role in improving observability (C08), reducing unplanned work (M18), and ensuring talent retention (M22) within the team.

***I16*:** The next participant values collaboration and communication for learning and job satisfaction (C01, C02, C07). Saying "learning from mistakes in blameless postmortems builds team trust" (C06). Monitoring and observability are needed to prove DevOps' efficacy and maintain system stability (C08,M12). For successful DevOps adoption, the participant emphasizes visual management, value stream transparency, and people, processes, and technology (C12, C14, C17, L10). Continuous integration, delivery, deployment, test automation, version control, and a loosely coupled architecture are all critical project technicalities. Metrics like DF, CFR, and mean time-to-failure are crucial for measuring DevOps success (M02, M03, M08). Resistance to change and the need for cross-team collaboration are acknowledged, as are the benefits of faster deployments, automation, and a blameless culture that contributes to team happiness (M21).

***I17*:** The interview discusses DevOps adoption and the shift from startup to metrics-driven culture (M05, M12, and L15). Workload management using priority support (L10, L17) and collaboration and communication (C01). He stresses "transparency and truthfulness in reporting" to aid decision-making (C04,C19). The participant also stresses job satisfaction (C07) and its relationship to DevOps capabilities. Resistance to change, competitive advantage, and incident management (C08), especially across time zones, are mentioned. The interview shows a shift to a more mature, measurement-focused organization that values data-driven approaches (C19), continuous integration (C20), customer focus (C18), and proactive resource management to boost profitability and service availability (M12, L04, L05).

***I18*:** The participant discusses DevOps adoption, including its challenges and successes, in the interview. Participant evaluates DevOps capabilities like learning support (C02), transformational leadership (C04), and blameless postmortems (C06), demonstrating "strong leadership and an experimentation-friendly culture" (C05). Time zone differences and overreliance on senior members hinder collaboration and progress (C01). While metrics like lead time for changes (M01) and MTTR (M08) are mentioned, areas like alert response and emergency response still need improvement. Visual management (C12) with Cortex, proactive monitoring (C08), and working in small batches (C15) are also discussed. Finally, stated that "Jenkins focuses on process automation" (C20) and test automation improvement (C22).

***I19*:** The participant praised the organization's DevOps capabilities, especially open source software adoption (C03) and version control (C23), indicating recent adoption of new practices and tools. However, "not built here meant rejecting outside solutions". Moderate ratings for collaboration (C01), cost management, and proactive monitoring (C08) suggest improvement. Testing and monitoring procedures should be more secure, standardized, and consistent. Quality and efficiency are indicated by high ratings for CFR (M03), pipeline automation (M09), and test coverage (M10). C07 and M21 addressed employee turnover, frequent tool changes, and job satisfaction.

***I20*:** The interviewee emphasized the importance of DevOps capabilities and metrics in the SLCP. Participant considers collaboration (C01) and open source adoption (C03) high-impact skills. Emergency response (C09) and autoscaling (C08) affect teamwork. Customer-focused feedback (C18) and working in small batches (C15) affect many aspects of the participant's context. Containerization (C31) created a "clean environment and reduced regressions and bugs", changing engineers' views on components in three years. Participant emphasized the significance of DF (M02) for promoting smaller, frequent releases. Talent retention (M22) and job satisfaction (C07) affect DevOps adoption and organizational success.

***I21*:** The interview highlights DevOps' benefits for team collaboration, communication (C01), and application development over infrastructure management (L04). DevOps allows more frequent feature delivery (C21), better integration (C20), and infrastructure-free work. The stakeholder says "DevOps improved programming, quality control, and operations coordination" (C01). Teams have adopted new methods through experience and learning (C02,C03,L01). Open-source C03 pipelines find vulnerabilities. M04 and M02 "deployment duration and frequency accelerated feature releases and correction". Product development speed, quality, and security improved by overcoming documentation and learning curves. Implementing DevOps in the software lifecycle improved decision-making (L11), team motivation (M21), and customer follow-up (M23).

***I22*:** The interviewee evaluates DevOps adoption, including capabilities and metrics, across the software lifecycle. Open source technology (C03) and Kubernetes accelerate progress. Participant acknowledges continuous improvement (C13) but note a lack of business perspective in monitoring systems (C10). Continuous integration (C20), delivery (C21), and configuration management (C25) are regarded as critical, while containerization (C31) and cloud infrastructure (C26) are deemed essential. Improve "test automation and trunk-based development" (C22, C33). Quick incident response requires metrics like DF (M02) and MTTF (M13). Acquisitions (L01) impact DevOps more than project planning (L09). Managing decisions (L11) is valued. The interview describes the company's critical but optimistic DevOps journey, which emphasizes continuous improvement and strategic use of open source (C03) and cloud native technologies.

***I23*:** Customer-driven feature requests (C18) and team learning (L08,C01) are discussed in the interview as DevOps adoption and practices affect the company. Cultural factors like blameless postmortems (C06) boost high job satisfaction (C07). The participant suggests tracking more metrics to improve observability, proactive monitoring (C08), and emergency response (C09). Customer feedback (C18), people, process, and technology (C14) are also discussed in DevOps success. Continuous integration (C20) and deployment, code infrastructure (C30), and security (C32) are well implemented, but "metrics tracking and documentation could be improved". Participant stresses the "importance of service availability" (M12), uptime, and adherence to SLAs and Service Level Objectives (SLOs) (M15).

***I24*:** The interviewee discusses how LCPs, capabilities and metrics affect DevOps adoption. The participant appreciates open source software adoption (C03) and controlling more with fewer manual work. Cross-team collaboration (C01) and transformational leadership (C04) were questioned, suggesting organizational culture is limiting progress. Change approval issues (C16) and value stream visibility (C17), especially for products, are also mentioned. Participant questions progress despite positive metrics like DF (M02) and service availability (M12) due to "testing issues".

***I25*:** The interview highlights the importance of enhancing DevOps adoption through better team collaboration and communication, along with fostering a cultural shift. Despite using open source software (C03), the company could make more contributions. Progressive monitoring, observability, and auto-scaling (C08) are well implemented, but value stream visibility (C17) and customer focus (C18) are not. Continuous integration (C20) and deployment are in place, but the quality varies. Test automation (C22) and environment provisioning are both average. Decision-making empowerment (C24) and configuration management (C25) are relatively strong, but "manual processes remain". Cloud native infrastructure, particularly "Kubernetes, has streamlined deployment, monitoring, and recovery". However, there are significant gaps in test data management (C35), chaos engineering (C36), and distributed tracing.

***I26*:** In the interview, the participant discusses the experience with DevOps adoption, highlighting the migration from a Jenkins system to a cloud-based solution with the help of a platform engineer (C01). "This migration saved us significant time by hosting a private Jenkins instance and copying the existing job configurations". The participant notes a gap in the team's CI/CD pipeline, specifically the lack of an automated process for deployment and subsequent test triggering (C21). Metrics such as response time to issues (M14), job satisfaction (M21), and emergency response (M08) are highly evaluated, while continuous delivery and deployment are noted as areas that "need improvement due to manual intervention" requirements (C21). The participant expresses a desire to establish a proper CI/CD pipeline and recognizes the importance of capabilities such as version control (C23), automated testing (C22), and collaboration for successful DevOps implementation.

***I27*:** The interviewee discusses DevOps adoption factors and the importance of certain capabilities and metrics throughout the software life cycle. Job satisfaction (C07), learning culture (C02), and cross-project collaboration (C01) are crucial to DevOps adoption. For DevOps success, the participant emphasizes proactive monitoring (C08) and continuous improvement (C13). The metrics DF (M02), CFR (M03), and service availability (M12) are crucial for assessing DevOps performance. Feature parity, deterministic builds, and output parsing in CI/CD hinder DevOps adoption. The participant said, "successful DevOps capabilities are characterized by standardization and reliability, which can increase job satisfaction" (C07) and organizational efficiency. Integration and management of DevOps capabilities also require acquisition (L01) and infrastructure management (L03).

***I28*:** The interview discussed the organization's adoption of DevOps, as well as its preference for open source tools (C03) and cultural shift toward more engaging technologies. During evaluation, the organization led in release quality (M03) and proactive response and uptime (M12) throughout the software lifecycle. In particular, people-centric focus (C14), loosely coupled architecture, and software acquisition need improvement. The participant discusses CI/CD pipeline setup and SRE practices (C08), "despite challenges in prioritization and DevOps implementation". The organization also aims to deploy continuously and standardize tools and processes to "promote observability and shared knowledge" in DevOps.

Finalizing Interviews: The last interview took place in September 2023, as everyone who agreed to be interviewed within the organization had been interviewed. There is a noticeable repetition of themes with residual appearances in subsequent interviews, as seen in Tables 11, 12 and 13. The consistency in the capabilities, metrics, processes, initiatives, challenges, and benefits mentioned suggests that additional interviews would not yield significantly different insights. In qualitative research, it is crucial to recognize when additional data do not improve the questions [18], while requiring multiple stakeholder perspectives.

3.2.2. Relevant Documents

The documents collected include DORA assessment surveys, DevOps vision documents, and historical performance metrics (refer to Table 7). The adoption process is better understood with these qualitative and quantitative DevOps performance and practice data. Transformational engineering culture, high-performance software delivery, and platform automation roadmaps. These documents allow interviewees' feedback to be cross-checked and triangulated. Engineering culture transformation, high-performance software delivery, and platform automation roadmaps. Document analysis cross-checks and triangulates feedback.

D01,2,3 - DORA 2021 to 2023 assessments: The DORA assessments from 2021, 2022, and 2023 provide a comprehensive analysis of the organization's main business unit DevOps capabilities and capabilities over a three-year period. These assessments measure key metrics such as Deployment Frequency (DF), Mean Lead-time for Changes (MLT), Mean Time To Recover/Restore (MTTR), and Change Failure Rate (CFR), indicating the level of performance of the software delivery. They are crucial for tracking and evaluating DevOps deployment, detecting trends, improvements, and opportunities for improvement. Continuous evaluation over three years provides insights and lessons for software delivery optimization. The **2021** assessment establishes a baseline for the maturity of DevOps in the organization at the outset of the adoption process. Identifies areas for improvement and helps set goals for the DevOps transformation journey. The **2022** assessment shows the progress made after a year of adopting DevOps capabilities. Provides information on the effectiveness of the strategies implemented, highlights successes, and pinpoints ongoing challenges derived from the change of processes. The **2023** assessment provides an overview of DevOps capabilities, highlighting sustained improvements, consolidation of gains, and increased maturity. It also helps highlight areas that need greater attention or new initiatives to develop DevOps. These assessments provide a comprehensive timeline of an organization's journey, enabling stakeholders to evaluate return on investment, make data-driven decisions, and benchmark against industry standards for software delivery and operational capabilities (see Figure 8). DORA assessments help organizations make strategic decisions, evaluate DevOps impact on software delivery efficiency, and analyze performance metrics to determine its effects on release frequency, time-to-market, and software quality.

D04 - DevOps culture presentation Transformation and adoption begin with the presentation of the DevOps 2020 culture. This talk introduces DevOps concepts such as collaboration, automation, and continuous improvement to help organizations transform. The presentation teaches new members the DevOps culture in training materials. It ensures that everyone is aware of the organization's vision and strategy. New

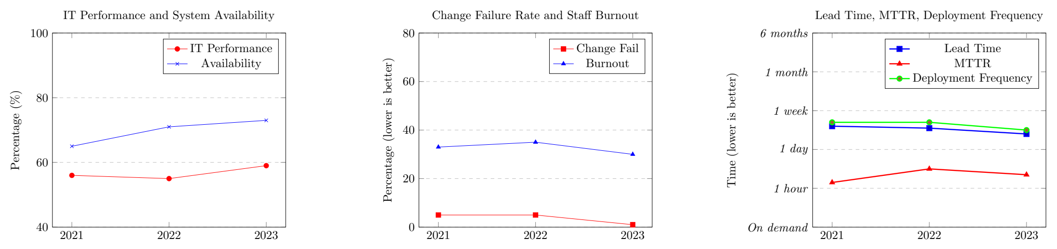


Figure 8: DORA assessment results from 2021 to 2023.

processes, tools, and functions are explained throughout the slides. Explaining the DevOps culture helps educate and unite the company, providing the basis for structural and procedural changes.

D05 - DevOps CI/CD pipeline diagrams In 2020, the organization began its DevOps journey. The 2021 document is an improvement. The report provides strategic ideas for implementing a CI/CD pipeline within a unified Platform Engineering (PE) vision. The organization's transformational leadership and commitment are shown in this key component of the DevOps adoption plan. Additionally, it lists pipeline stages, engineering responsibilities, and quality, compliance, and reliability tools. A vision for integrating DevOps into the ever-changing organization's workflow. It emphasizes the need of a "north star" vision for driving the transformation towards improved DevOps culture and provides a full overview and conceptual framework for displaying code flow from development to production. Overall, this architecture diagram aimed to standardize CI/CD and release automation. The DevOps adoption strategy focuses on a successful standard pipeline that adheres to DevOps principles.

D06 - DORA Presentation - "High-Performance Software Delivery" The DevOps Research and Assessment (DORA) team's 2022 presentation promotes DevOps and Site Reliability Engineering (SRE) best practices for high-performance software delivery. In this presentation, the DORA team shares their insights and best practices to help the organization improve software delivery quality. In a fast-changing technology landscape, it motivates the company to innovate. The presentation presents the latest study from DORA on the impact of DevOps on software delivery performance. The key performance measures are DF, MLT, MTTR, and CFR. It also advises on DevOps and SRE implementation, including cultural shifts, automation, pipelines, and monitoring and observability.

D07,8,9 - Relevant Metrics D07, D08, and D09 indicators are crucial for this case study since they give measurable data on DevOps adoption and integration. Absolute numbers have been excluded to protect data confidentiality, since scales and percentages provide enough insights for this case study.

D07) Adoption progress of *DevOps and theCI/CD pipeline.* A pipeline is a system that automates various software processes, allowing for timeline visibility into the organization's source code repositories and encouraging more efficient, standardized, compliant, and automated workflows. The steady increase in this metric indicates positive adoption. Figure 9 shows repository management, including integration into CI/CD pipelines, archiving outdated repositories, and monitoring their active and ownership status. These graphs illustrate the company's DevOps-enhanced repository management efforts. The standard CI/CD has improved code awareness and determined repository ownership. The number of repositories increased significantly in 2022, while the number of unowned repositories was reduced by more than 88%, demonstrating the increased efficiency of the version control system.

D08,D09) *Escaped defects* (D08) are usually found in development companies. See Figure 10 for measurements. Accordingly, excaped defects are issues found during production. Implementing DevOps should reduce defects, improving software quality and reliability. The *Release success rate* (D09) measures the percentage of releases pushed to production without incidents, errors, or failures. Figure 10 indicates a rise in successful releases. Successful updates highlight the organization's capacity to roll out new updates without disruptions, credited to DevOps practices like automated testing and *Continuous Integration* (CI). These metrics document the company's DevOps adoption experience, emphasizing successes and areas for improvement. The amount of escaped bugs and pipeline onboarding state imply that DevOps is improving

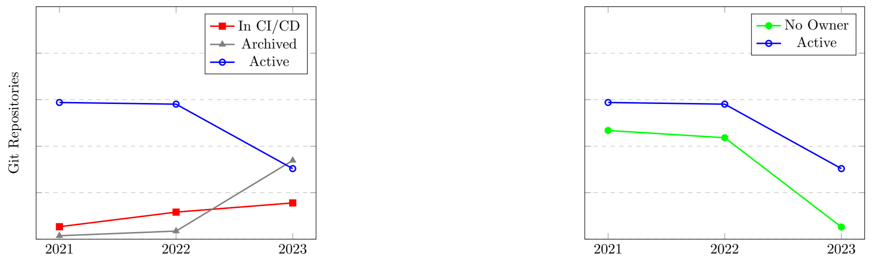


Figure 9: CI/CD Pipeline onboarding and repositories with no clear owner over time.

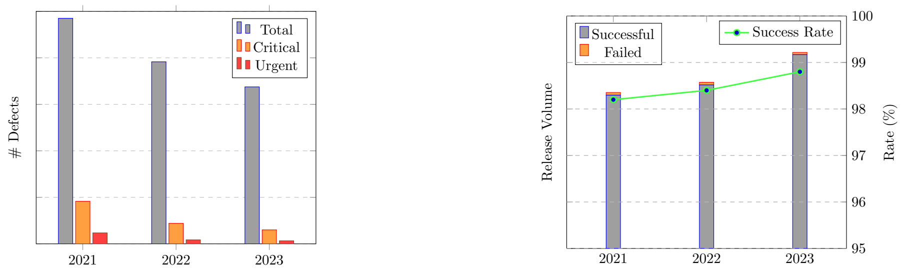


Figure 10: Escaped defects and Release volume over time.

software delivery efficiency and quality. Before adoption or baseline performance, these measures can be compared to industry norms. This lets the company define measurable goals. This complete view can help other organizations transform.

3.2.3. Focus Group

An expert focus group discussed and validated the interviews and documents findings. The focus group was led by the lead researcher, encouraged participatory discussion and idea sharing, enriching data for analysis (see Figure 7). Company employees and a Google DORA observer examined DevOps capabilities, metrics, lifecycle processes, and their relationships. Key themes included the importance of cross-team collaboration, proactive monitoring, continuous improvement, and integration of the CI/CD pipeline which are in line with the IEEE Standard for DevOps [22] and the 37 identified DevOps capabilities [3] and 24 metrics [4] from previous research. The most notable initiatives identified within the company were the adoption of the CI/CD pipeline, the implementation of an internal catalog with maturity assessment, and the establishment of a common cloud native platform. In addition to confirming the interview findings, the participants identified challenges such as resistance to change, the need for prioritization, and improved communication. The panel agreed that DevOps improved velocity, efficiency, and delivery performance and that customer success and team happiness were essential. This was consistent with the findings of a systematic literature review that emphasizes better collaboration and faster delivery. The discussion concluded that DevOps initiatives should fit organizational goals and value continual learning and experimentation. The focus group validated and refined the interviews, papers, and metrics to correctly reflect the thoughts and experiences of various roles in DevOps adoption.

3.3. Case Study Validity

Per Yin [55] four tests are considered to validate the reliability of case studies, including Construct Validity, Internal Validity, External Validity, and Reliability. In this particular case study, all these tests were performed except for internal validity, as Yin [55] suggests that it is not applicable to exploratory research. The construct validity test used multiple data sources, such as semi-structured interviews, document analysis, and a focus group. To check for external validity, a review of the existing literature was done in Section 2.

This strongly suggests that there needs to be continuity in the study of certain DevOps capabilities, metrics, and LCPs [3, 4, 22], which makes this research stand out. Replicability was addressed in the research methodology, which provides the required reliability of the case study.

By utilizing multiple data sources, triangulation enhances the validity and reliability of this exploratory case study, thereby enhancing the credibility and adaptability [42]. Specifically, under *Methodological triangulation*, a semi-structured interview approach, archival documents, measures, and a focus group were the sources of data collection. Cross-referencing this interview data with archival text and focus group discussions made triangulation possible, thus achieving fuller comprehension of the DevOps adoption process. A spreadsheet was employed for the analysis and correction of missing data with notes. *Member checking* entailed disseminating the findings to the subjects that were involved in this research so as to ascertain the precision of the details and interpretations. *Rich description* offered in this study identifies the context and individuals involved in them while ensuring confidentiality. This has made it possible for the reader to compare with other similar contexts, thus facilitating evaluation of the findings in terms of transferability and taking into account any external variables that may affect the outcomes. Analytical methods and a diverse sample of interviewees provided *clarification of bias*, including those inherent to qualitative methods and related to participant organizations, improving generalization of findings. *Peer Review* includes other researchers and experts to review the process and the findings to provide an external check. An *Audit Trail* was kept via documenting research procedures (data collection & analysis) to bolster transparency as well as responsibility. A *Case Study Database* was created to consolidate notes, papers and other information for enhancing the dependability of the case study. *Prolonged Engagement* with participants and maintaining ongoing communication during the study helped to understand their perspectives and ensure that all results, including unexpected ones, were fully captured and considered.

4. Results

The findings of the research question are presented here, along with the corresponding results, without any interpretation or discussion at this stage.

4.1. Main Findings

The research findings, as seen in Table 9, provide a summarized view of the study, highlighting the importance of DevOps in improving software quality and reducing delivery times. The study highlights the importance of leadership in driving, managing and assessing DevOps adoption maturity.

These findings collectively demonstrate the multifaceted impact of DevOps integration, emphasizing the need for a well-orchestrated approach to realize operational efficiencies and enhance software delivery outcomes.

4.2. Measuring DevOps adoption Impact Results

A percentile-based impact scale (Table 10) is adopted in this case study to quantify and compare findings across research questions, facilitating structured discussion. According to Guest et al. [18], a percentile-based approach is used to estimate the likelihood of themes in qualitative research, while Verner et al. [50] uses percentiles to analyze the distribution of responses on motivational factors and project outcomes. A percentile is a statistical measure of the percentage of dataset observations below a certain value. By dividing a data set into 100 equal parts, percentiles can be used to compare scores or values with the rest of the data [47]. The use of percentiles allows researchers to compare and evaluate data, as noted by several authors [6, 50]. This is useful for understanding a value's position in a distribution and comparing data points to the whole set, helping researchers draw more accurate conclusions. Percentile scales are crucial in technology research and industry for assessing innovation impact on market share, adoption rates, and revenue, revealing technology effectiveness and company success [10]. These tools are also used to compare data in finance, healthcare, and demographics. Disparities and inequalities are compared in this analysis. Inequality is often measured by comparing two opposite percentiles in the scale [18]. The ratios above and below the 90th and 10th percentiles of interviewees' sum and average ratings in Tables 11, 12, and 13 indicate the impact disparity

Table 9: A summary of the main findings in this research

ID	Main finding	Main source	Confirmed by	References
F1	Exceptional and High Impact Capabilities	Interviews	Docs, Focus Group	Listed in Section 4.3
F2	Exceptional and High Impact Metrics	Interviews	Docs, Focus Group	Listed in Section 4.4
F3	Exceptional and High Impact LCP	Interviews	Docs, Focus Group	Listed in Section 4.5
F4	23 Key relationships in capabilities and metrics	Interviews	Docs, Focus Group	Listed in Section 4.4
F5	11 Key relationships in capabilities and LCPs	Interviews	Docs, Focus Group	Listed in Section 4.4
F6	4 Key relationships in metrics and LCPs	Interviews	Docs, Focus Group	Listed in Section 4.4
F7	6 Most acknowledged strategies and initiatives	Interviews	Docs, Focus Group	Listed in Section 4.5
F8	5 Most acknowledged benefits	Interviews	Docs, Focus Group	Listed in Section 4.6
F9	5 Most acknowledged challenges	Interviews	Docs, Focus Group	Listed in Section 4.6
F10	DevOps CI/CD Pipeline Adoption is driven by Transformational leadership	Interviews	Docs, Focus Group	Ref. in I12, I13, I18, I24, D04, D05
F11	Lesson: Document baseline metrics before and during DevOps adoption	Document (DORA)	Focus group	Ref. in D01, D02, D03, D07, D08, D09
F12	Moving repository management to a controlled self-service system speeds up the archival of old repositories and increased clear ownership by 88%	Document (CI/CD)	Focus group, Metrics	Ref. in D05, D07, D08, D09
F13	Having an Internal Catalog is essential for Maturity Assessment and reduce time to identify ownership of microservices	Interviews	Docs, Focus Group	Ref. in I12, I15, I17, I22, I25, D07
F14	While releases increased, escaped defects and MTTR have decreased. Quality Assurance (QA), CI/CD Pipeline, Automated Tests, and Code Coverage have contributed to this	Interviews	DORA, Metrics, Focus group	Ref. in I02, I06, I14, I18, I23, D05, D07, D08, D09 and Section 4.4
F15	Acquiring FLOSS is key to improving DevOps adoption. In this case study, Cloud Native represents a well-structured example for infrastructure management	Interviews	Docs, Focus Group	Ref. in Section 4.3, Section 4.5 and Section 4.4
F16	Less relations were found between Metrics and LCPs indicating a possible field of further investigation	Interviews	Focus group	Ref. in Section 4.4

between two points of the distribution. The distribution is more unequal with a higher ratio. This ratio lets us compare the case study's inequality of capabilities, metrics, and lifecycle processes. The 90–10 percentile

Table 10: Percentile-Based Impact Classification for Thematic Ranking.

Impact Level	**Percentile Range**	**Impact Definition**
Exceptional	$> 90th$	Represents values exceeding the 90th percentile, indicating an exceptionally higher sum or average within the dataset.
High	$75th < 90th$	Covers values between the 75th and 90th percentiles, denoting a high impact that is not quite at the peak of the scale.
Increased	$50th < 75th$	Encompasses values between the median and the 75th percentile, showing a higher impact than average.
Reduced	$25th < 50th$	Includes values between the 25th and 50th percentiles, indicating a moderate impact that is less than the median.
Low	$10th < 25th$	Reflects values between the 10th and 25th percentiles, suggesting a lower level of impact.
Minimal	$< 10th$	Shows values below the 10th percentile, representing the minimal impact within the dataset.

ratio should be considered when considering inequality. The 90 to 50 percentile ratio may reveal inequality in the upper half of the income distribution. The 50th percentile (median) represents the value below which 50% of data points fall [47].

4.3. RQ1: What DevOps capabilities, metrics, and Lifecycle processes play a key role in enhancing DevOps adoption?

The influence of DevOps capabilities on the Company's adoption is presented first. Table 11 is showing results from 28 interviews. How often was each relation mentioned in interviews. Presenting the main DevOps capabilities identified as having the most positive impact on DevOps adoption. Supported by direct quotes from interviewees, focus group discussions and documentation. Table 11 provides an overview of DevOps capabilities, including Cultural, Measurement, Process, and Technical categories, their important findings, familiarity, and impact.

Table 11: Impact of DevOps Capabilities [3] in DevOps adoption.

			Freq.	Impact Level			Familiarity
Category	ID	Capability	#	SUM	AVG	max $SUM \cup AVG$	$freq_C(\#)$
Cultural	C01	Cross team collaboration and communication	26	204	7.85	Exceptional	Exceptional
	C02	Support learning culture and experimentation	14	99	7.07	Reduced	Increased
	C03	Open source software adoption	17	133	7.82	High	High
	C04	Transformational leadership	14	112	8.00	Increased	Increased
	C05	Performance/Westrum organizational culture	8	54	6.75	Low	Low
	C06	Blameless Postmortems/reduced fear of failure	15	114	7.60	Increased	Increased
	C07	Job satisfaction	14	111	7.93	Increased	Increased
Measurement	C08	Proactive Monitoring, Observability and autoscaling	24	198	8.25	Exceptional	Exceptional
	C09	Emergency response/proactive failure notification	20	168	8.40	High	High
	C10	Monitor systems to inform business decisions	11	79	7.18	Reduced	Low
	C11	Working in progress limits	8	53	6.63	Low	Low
	C12	Visual management Capabilities	13	101	7.77	Increased	Reduced
Process	C13	Continuous Improvement of processes/workflows	24	180	7.50	High	Exceptional
	C14	Focus on people, process and technology	17	131	7.71	Increased	High
	C15	Working in small batches	12	85	7.08	Reduced	Reduced
	C16	Lightweight change approval	16	114	7.13	Increased	Increased
	C17	Visibility of work in the value stream	12	82	6.83	Low	Reduced
	C18	Customer focus/feedback	15	102	6.80	Reduced	Increased
	C19	Data-driven approach for improvements	12	89	7.42	Reduced	Reduced
Technical	C20	Continuous Integration	24	207	8.63	Exceptional	Exceptional
	C21	Continuous Delivery/Deployment automation	23	188	8.17	Exceptional	High
	C22	Test Automation and environments	20	156	7.80	High	High
	C23	Version Control System	19	180	9.47	Exceptional	High
	C24	Empower teams to make decisions/changes	12	97	8.08	Increased	Reduced
	C25	Configuration Management	15	130	8.67	Exceptional	Increased
	C26	Cloud infrastructure and cloud native	12	108	9.00	Exceptional	Reduced
	C27	Artifacts versioning and registry	15	123	8.20	High	Increased
	C28	Loosely coupled architecture	13	100	7.69	Reduced	Reduced
	C29	Database change management	7	45	6.43	Minimal	Minimal
	C30	Infrastructure as Code	16	133	8.31	High	Increased
	C31	Containerization	18	155	8.61	High	High
	C32	Shift left on security	16	117	7.31	Increased	Increased
	C33	Trunk based development	11	94	8.55	High	Low
	C34	Centralized log management	11	90	8.18	Increased	Low
	C35	Test data management	7	39	5.57	Minimal	Minimal
	C36	Chaos Engineering	7	34	4.86	Minimal	Minimal
	C37	Code maintainability	6	41	6.83	Low	Minimal

Legend: Frequency: ($\#$) Number of interviews mentioning this item;
Impact Level: (SUM) Sum of evaluations given in interviews; (AVG) Average evaluations from interviews; $max(SUM \cup AVG)$ The maximum level from the union of SUM and AVG.
Familiarity: ($freq_C(\#)$) The familiarity of the item given the frequency of mentions.

The **Cultural** category emphasizes the strong impact and familiarity of cross-team collaboration and communication (C01) in DevOps adoption. Open source software adoption (C03) and transformational leadership (C04) also have high influence and familiarity, contributing to a DevOps culture.

In the **Measurement** domain, Proactive Monitoring, Observability, and Autoscaling (C08) excels in impact and familiarity, highlighting the significance of monitoring in DevOps success. Emergency response/proactive failure notice (C09) has significant impact and familiarity.

For the **Process** category, Continuous Improvement of processes/workflows (C13) is identified as having high impact and exceptional familiarity, highlighting the essence of continuous improvement in successful DevOps implementation. Focus on people, process, and technology (C14) also demonstrates high impact and familiarity, pointing towards its importance in DevOps processes.

In **Technical**, capabilities such as Continuous Integration (C20) and Version Control System (C23) receive exceptional ratings in both impact and familiarity, indicating their foundational role in DevOps capabilities. Additionally, Configuration Management (C25) and Cloud infrastructure and cloud native (C26) are marked as exceptionally impactful. Overall, the analysis underlines the importance of collaboration Cross team collaboration and communication (C01), Proactive Monitoring, Observability, and autoscaling

(C08), continuous improvement of processes/workflows (C13), and core technical practices like Continuous Integration (C20) and Cloud infrastructure and cloud native (C26) in driving DevOps adoption. It also highlights areas with potential for growth or improvement, such as database change management (C29) and chaos engineering (C36), suggesting these as areas where DevOps capabilities could evolve further to maximize success.

The study also provides findings related to the impact of DevOps capabilities and metrics on the company's DevOps adoption journey. Table 12 shows the results of 28 interviews, with the number of mentions for each relation.

Table 12: Impact of DevOps Metrics [5] in DevOps adoption.

			Freq.	Impact Level			Familiarity
Category	ID	Metric	#	SUM	AVG	max $SUM \cup AVG$	$freq_C(\#)$
Change	M01	Mean Lead-time for Changes (MLT)	14	112	8.00	High	Increased
	M02	Deployment Frequency (DF)	21	168	8.00	High	High
	M03	Change Failure Rate (CFR)	13	106	8.15	Exceptional	Reduced
	M04	Deployment Duration Time	18	143	7.94	High	Increased
	M05	Defect Escape Rate	12	93	7.75	Increased	Reduced
	M06	Cycle Time Value (CTV)	7	43	6.14	Minimal	Minimal
	M07	Deployment Size	10	71	7.10	Reduced	Reduced
	M08	Mean time to failure (MTTF)	11	78	7.09	Reduced	Reduced
	M09	Pipeline Automated Tests Success Rate	17	134	7.88	Increased	Increased
	M10	Automated Test Code Coverage	19	159	8.37	Exceptional	High
Operational	M11	Mean Time To Recover/Restore (MTTR)	14	112	8.00	High	Increased
	M12	Service Availability and Uptime	22	190	8.64	Exceptional	Exceptional
	M13	Mean Time To Detection (MTTD)	11	84	7.64	Reduced	Reduced
	M14	Application Response Time	8	65	8.13	High	Minimal
	M15	SLAs and SLOs	17	130	7.65	Increased	Increased
	M16	Production Error and Incident Rate	10	77	7.70	Reduced	Reduced
	M17	Work in Progress (WIP) /Load	7	43	6.14	Minimal	Minimal
	M18	Unplanned Work Rate (UWR)	9	47	5.22	Minimal	Low
	M19	Wait Time	9	62	6.89	Low	Low
Cultural	M20	Westrum Organizational Culture Measures	14	94	6.71	Reduced	Increased
	M21	Team Happiness	24	185	7.71	Exceptional	Exceptional
	M22	Talent retention	19	143	7.53	High	High
Business	M23	Customer Tickets Volume and Feedback	24	175	7.29	Exceptional	Exceptional
	M24	Customer Usage and Traffic	11	88	8.00	High	Reduced

Legend: Same as in Table 11.

Change category, which includes metrics M01 through M10, emphasizes the importance of deployment and change management in DevOps capabilities. Change Failure Rate (M03) and Automated Test Code Coverage (M10) have a particularly strong impact, for minimizing failures and ensuring high coverage through automated testing. Metrics such as DF (M02) and Deployment Duration Time (M04) emphasize deployment efficiency and frequency. Additional metrics like MLT (M01) and Pipeline Automated Tests Success Rate (M09) are increasingly valued by DevOps professionals. Focus is needed on lesser-known areas like CTV (M06) and MTTF (M08).

Operational Metrics M11-M19 show the functional excellence and stability needed for successful DevOps adoption. Availability and Uptime (M12) is very impactful and well known, emphasizing its necessity for service reliability. MTTR (M11) and Application Response Time (M14) point out fast recovery and responsive applications. However, indicators such as work in progress/load (M17) and the rate of unplanned work (M18) are still unfamiliar, offering potential for growth.

Cultural M20-M22 metrics emphasize the importance of a positive organizational culture for successful DevOps adoption. Strong and familiar team happiness (M21) shows that DevOps productivity depends on team happiness. Retaining qualified individuals to maintain DevOps capabilities is also important (M22). The Westrum [52] measures (M20) are becoming more know, but still need improvement.

Business Metrics such as Customer Tickets Volume and Feedback (M23) and Customer Usage and Traffic (M24) show how DevOps capabilities affect customer happiness and engagement. Customer feedback improves DevOps, as shown in Customer ticket volume and feedback (M23). Customer engagement is crucial to the company's success, as shown by the Customer Usage and Traffic (M24). The high ratings for these measures suggest that DevOps skills increase business performance, underlining the necessity to monitor and adapt to consumer needs.

The impact levels of lifecycle processes in DevOps adoption are provided in Table 11.

Table 13: Impact of DevOps Life Cycle Processes [22] in DevOps adoption.

			Freq.	Impact Level			Familiarity
Category	ID	Life Cycle Process	#	SUM	AVG	max $SUM \cup AVG$	$freq_L(\#)$
Agreement	L01	Acquisition process	19	138	7.26	Exceptional	Exceptional
	L02	Supply process	11	78	7.09	Increased	Increased
Organizational Project-Enabling	L03	Life Cycle Model Management process	6	49	8.17	Exceptional	Low
	L04	Infrastructure Management process	17	137	8.06	Exceptional	Exceptional
	L05	Portfolio Management process	6	44	7.33	Reduced	Low
	L06	Human Resource Management process	7	44	6.29	Low	Reduced
	L07	Quality Management process	14	107	7.64	Exceptional	Exceptional
	L08	Knowledge Management process	10	79	7.90	High	Increased
Technical Management	L09	Project Planning	13	93	7.15	High	High
	L10	Project Assessment and Control	8	54	6.75	Reduced	Reduced
	L11	Decision Management	9	61	6.78	Reduced	Reduced
	L12	Risk Management	13	85	6.54	Increased	High
	L13	Configuration Management	7	57	8.14	Exceptional	Reduced
	L14	Information Management	6	46	7.67	Increased	Low
	L15	Measurement	9	68	7.56	Increased	Reduced
	L16	Quality Assurance	16	131	8.19	Exceptional	Exceptional
Technical	L17	Business or Mission Analysis	6	37	6.17	Minimal	Low
	L18	Stakeholder Needs and Requirements Definition	12	80	6.67	Increased	Increased
	L19	System/Software Requirements Definition	7	41	5.86	Low	Reduced
	L20	Architecture Definition	14	104	7.43	High	Exceptional
	L21	Design Definition	4	22	5.50	Minimal	Minimal
	L22	System Analysis	6	41	6.83	Reduced	Low
	L23	Implementation	18	143	7.94	Exceptional	Exceptional
	L24	Integration	12	90	7.50	Increased	Increased
	L25	Verification	11	85	7.73	High	Increased
	L26	Transition	8	62	7.75	High	Reduced
	L27	Validation	8	53	6.63	Reduced	Reduced
	L28	Operation	11	83	7.55	Increased	Increased
	L29	Maintenance	13	98	7.54	High	High
	L30	Disposal	3	20	6.67	Reduced	Minimal

Legend: Same as in Table 11.

In **Agreement**, Acquisition process (L01) and Supply process (L02) highlight the importance of procurement and supply in the DevOps life cycle. The emergence of the acquisition process (L01) shows how important acquisition is in ensuring successful adoption and implementation of DevOps capabilities. The supply process (L02), while also important, exhibits increased impact and familiarity, indicating a recognition of the supply process's role in supporting DevOps environments.

The **Organizational Project-Enabling** category, featuring processes L03 through L08, considers the management aspects crucial for fostering a conducive DevOps ecosystem. An exceptional impact is observed in the Management process (L04) and Quality Management process (L07), pointing to the paramount importance of robust infrastructure and quality oversight in DevOps. Knowledge Management process (L08) is highly regarded, emphasizing the value of effectively managing knowledge for DevOps success.

In the **Technical Management** sphere, processes L09 through L16 explore the technical oversight necessary for DevOps. Configuration Management (L13) and Quality Assurance (L16) stand out with exceptional impact, highlighting their indispensable roles in maintaining system integrity and ensuring quality within DevOps capabilities. The high impact of Project Planning (L09) and Risk Management (L12) highlights the importance of meticulous planning and risk mitigation in the successful execution of DevOps projects.

Finally, the **Technical** category, encompassing processes L17 through L30, addresses the core technical activities integral to DevOps. Implementation (L23) has an exceptional impact and familiarity, highlighting efficient implementation practices in the adoption of DevOps. Architecture Definition (L20) and Maintenance (L29) are highly impactful, stressing architectural practices and ongoing maintenance to support DevOps efforts.

4.4. RQ2: Are there any significant correlations in DevOps capabilities, metrics, and lifecycle processes?

This research question describes the top 10% correlations between DevOps capabilities, metrics, and lifecycle processes found in the case study. The data analysis identifies these links, as shown in Tables 14, 16, and 18. From interviews and a focus group, the Capabilities and LCPs relations are described here.

Table 14: Top correlations mentioned between Capabilities and Metrics.

ID	**M01**	**M02**	**M03**	**M04**	**M05**	**M06**	**M07**	**M08**	**M09**	**M10**	**M11**	**M12**	**M13**	**M14**	**M15**	**M16**	**M17**	**M18**	**M19**	**M20**	**M21**	**M22**	**M23**	**M24**
C01	0	1	1	1	0	2	0	1	0	0	2	**3**	0	0	0	1	0	2	1	1	**4**	0	2	2
C02	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4**	2	1	0
C03	1	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	2	0	0	0
C04	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	**3**	1	**5**	0	0
C05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	**5**	0	2	0	0
C06	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	0	0
C07	0	0	0	2	0	0	0	0	2	1	0	0	0	0	0	0	1	0	0	1	**9**	**5**	0	0
C08	1	0	0	0	0	0	0	0	0	1	**6**	**11**	**3**	2	**3**	**3**	0	0	0	0	1	0	0	2
C09	1	1	1	0	0	0	0	0	0	0	2	**4**	**3**	0	**4**	1	0	0	0	0	1	0	1	1
C10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
C11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
C12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
C13	1	1	**4**	2	1	0	0	1	1	2	1	0	0	0	0	0	0	1	1	1	2	**3**	2	1
C14	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	**4**	2	0	0
C15	1	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C16	2	**3**	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C17	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	1	0	0
C18	1	0	0	0	0	0	0	0	0	0	0	1	1	0	2	0	0	1	0	0	1	0	**12**	2
C19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
C20	**7**	**7**	1	**4**	1	1	0	2	**5**	**3**	**3**	1	1	0	2	1	0	0	2	0	0	0	1	0
C21	**5**	**10**	1	**8**	2	2	**3**	1	**3**	**3**	**6**	2	0	0	0	0	1	0	2	0	0	0	1	0
C22	2	2	0	2	2	1	0	2	**5**	**7**	0	2	1	0	0	1	0	0	0	0	0	0	2	0
C23	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
C24	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0
C25	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
C26	1	1	1	1	1	1	1	1	1	1	1	2	0	0	1	0	0	0	0	0	0	0	0	0
C27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C28	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C30	0	1	1	0	0	0	0	0	0	0	0	2	0	0	**3**	0	0	0	0	0	0	0	0	0
C31	1	1	0	1	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C32	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
C33	2	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C35	0	0	0	0	0	0	0	1	2	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0
C36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Legend: Same Capabilities IDs from Table 11 and Metrics IDs from Table 12.

Other noteworthy findings stand out when associating Table 18 with the metrics in documents D07, D08, D09 from Section 3.2.2, which also share insightful data related to escaped defects, QA, and the adoption of a standard CI/CD pipeline. A clear relation is observed: although the release volume increased significantly, escaped defects decreased over time, as did the MTTR after a period of adaptation. For such an improvement, Quality Assurance (L16), Pipeline Automated Tests Success Rate (M09), Automated Test Code Coverage (M10) have had a positive contribution to Defect Escape Rate (M05). The graphs from Figure 9 show a substantial and determined effort of the organization to improve its version control system (C23) capabilities for repository management over a three-year period. The organization has made a significant effort, with an increase 88%, to ensure proper ownership of services, restructuring its repository portfolio, and implementing CI/CD practices to improve efficiency and quality management (L07). These efforts reflect a mature approach to software development and project management (L09), which is likely to lead to higher productivity, better code quality (L16), and a more effective use of resources [30]. These efforts resulted in more releases and better code quality, as previously seen in Figure 10. The SDLC has become more efficient and of higher quality since CI/CD pipeline and other DevOps capabilities were implemented.

Table 15: Relations summary for Capabilities and Metrics.

Capability	Metrics relations	Interviews
C01: Cross-team collaboration and communication	Linked to **Team Happiness (M21)** and **Service availability and uptime (M12)**. Effective collaboration and communication can lead to a more satisfied and cohesive team [43], which in turn can improve the reliability and availability of services	I01, I03, I09, I15, I16, I19, I26
C02: Support for learning culture and experimentation	Associated with **Team happiness (M21)**. A culture that encourages learning and experimentation can contribute to a more engaging and fulfilling work environment, leading to higher team happiness	I03, I16, I18, I21
C04: Transformational leadership	Related to **Talent retention (M22)** and **Westrum Organizational Culture (M20)** [52]. Transformational leaders can foster a positive culture that aligns with Westrum's typology of generative cultures, which can help retain talent by creating a supportive and innovative work environment	I08, I09, I13, I18, I20, I25
C05: Perfor-mance/Westrum organizational culture	The Westrum [52] model assesses the cultural aspects that contribute to high performance, such as cooperation, trust, and information flow, naturally related to measuring **Organizational Culture (M20)**	I05, I08, I13, I15, I18
C07: Job satisfaction	Related to **Team Happiness (M21)** and **Talent retention (M22)**. Job satisfaction is a key factor in team happiness and can significantly impact an organization's ability to retain skilled employees	I02, I03, I05, I08, I10, I11, I13, I14, I18, I20, I22, I23
C08: Proactive Monitoring, Observability, and Autoscaling	Related to **Service Availability and Uptime (M12)**, **MTTR (M11)**, **MTTD (M13)**, **SLAs and SLOs (M15)**, and **M16 Production Error and Incident Rate** Proactive monitoring and observability help identify service availability issues. This accelerates discovery and resolution, lowering MTTR. While autoscaling ensures service continuity during workload fluctuations. Observability and monitoring [4] are essential for meeting SLAs and SLOs in consistent performance levels, resulting in shorter MTTD	I02, I11, I12, I14, I15, I16, I18, I21, I22, I23, I24, I25, I26, I27
C09: Emergency Response/Proactive Failure Notification	Connected to **SLAs and SLOs (M15)**, **Service Availability and Uptime (M12)**, and **MTTD (M13)**. Emergency response strategies are vital for upholding SLAs and SLOs when unforeseen failures occur and must be addressed swiftly. Emergency response systems need automated detection functions to speed up failure detection	I08, I09, I12, I18, I19, I22, I23
C13: Continuous Improvement of processes/workflows	Related to **CFR (M03)** and **Talent retention (M22)** continuously enhancing processes and workflow can reduce change failures. Continuous improvement can boost work satisfaction and retention by involving employees in significant process improvements.	I08, I09, I14, I16, I19, I22
C14: Focus on people, process, and technology	Affects **Team Happiness (M21)** with balanced focus on people, processes, and technology contributes to a positive work environment, which can increase team happiness and productivity	I23, I25, I26, I28
C16: Lightweight change approval	Relates to **DF (M02):** Lightweight change approval processes can streamline deployments, allowing for more frequent and efficient release cycles	I01, I13, I18
C18: Customer focus/feedback	Measured by **Customer Tickets Volume and Feedback (M23)** Focusing on customers provides actionable input that can be utilized to improve products and services, minimizing customer tickets over time	I02, I04, I08, I10, I11, I13, I14, I15, I18, I19, I20, I23
C20: Continuous Integration	**DF(M02)** is likely to increase since CI allows frequent and smaller batches of code to be integrated and tested. **MLT (M01)** can reduce due to the streamlined process of integrating and testing changes. **Pipeline Automated Tests Success Rate (M09)** is an indicator of integration process quality because it shows the percentage of automated tests that pass during CI. **Deployment Duration Time (M04)** can be shortened as the CI process often includes automated deployment steps. **MTTR (M11)** may improve, as the CI process helps to quickly identify and address problems. **Automated Test Code Coverage (M10)** is an important metric that can be positively impacted by CI, as it encourages the creation of tests alongside new code commits	I01, I02, I07, I10, I11, I12, I13, I14, I15, I16, I18, I20, I21, I22, I23
C21: Continuous Delivery/Deployment automation	Improves **DF (M02)** by allowing a more consistent and reliable deployment process. **Deployment Duration Time (M04)** is often reduced due to automation of deployment tasks. **MTTR (M11)** can be improved because automated deployments allow faster recovery processes. **MLT (M01)** is likely to decrease as the path from code commit to deployment is automated and simplified. **Automated Test Code Coverage (M10)** can increase as the delivery process can enforce the requirement for sufficient test coverage before deployment. The **Pipeline Automated Tests Success Rate (M09)** should improve as the deployment process is automated, ensuring that only changes that pass all tests are deployed. **Deployment Size (M07)** can be managed more effectively, as the *Continuous Delivery or Deployment* (CD) process allows for smaller and more frequent deployments	I01, I02, I07, I08, I09, I11, I12, I13, I14, I15, I16, I18, I21, I22, I23, I25, I26, I27
C22: Test Automation and Environments	**Automated Test Code Coverage (M10)**, as it involves creating a comprehensive suite of automated tests that cover a significant portion of the codebase. The **Pipeline Automated Tests Success Rate (M09)** is also related, as a robust test automation strategy should result in a higher success rate of tests in the CI/CD pipeline	I10, I11, I13, I14, I15, I16, I22, I26
C30: Infrastructure as Code	**SLAs and SLOs (M15)**, as it allows consistent and reliable provisioning and management of infrastructure, which can lead to improved system reliability and performance	I07, I13, I19
C31: Containerization	**Deployment Size (M07)**, as it allows more granular control over the resources and dependencies of applications, potentially leading to smaller and more efficient deployments	I09, I13, I16

4.5. RQ3: How are key strategies, or initiatives, facilitating the adoption of DevOps?

This section answers the research question by looking at the main strategies, initiatives, or engineering drivers that the participants said had an effect on the adoption of DevOps. These are shown in Table 20. The analysis reveals several key strategies and initiatives that drive the adoption of DevOps within the organization. The implementation of the CI/CD pipeline significantly improves software delivery and reduces time to market. Internal catalog creation and maturity assessments help discover areas for growth and enable continual improvement. DevOps concepts, scalability, and agility are promoted by adopting a shared

Table 16: Top correlations mentioned between Capabilities and Life Cycle Processes.

ID	L01	L02	L03	L04	L05	L06	L07	L08	L09	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20	L21	L22	L23	L24	L25	L26	L27	L28	L29	L30
C01	0	0	0	0	1	0	0	4	3	2	2	1	1	2	1	1	1	4	1	5	1	0	1	0	0	0	0	1	0	0
C02	0	0	0	2	0	0	0	3	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C03	6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
C04	1	0	0	0	0	1	0	0	2	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C05	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C06	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
C07	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0
C08	2	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	3	2	0
C09	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C10	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C12	2	0	1	0	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C13	1	1	1	1	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0
C14	0	1	0	0	0	2	0	1	2	2	0	0	0	0	0	0	0	3	0	0	0	1	0	0	0	0	0	0	1	0
C15	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
C16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C17	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
C18	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0	0
C19	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
C20	1	1	0	0	0	0	4	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	3	5	1	2	1	0	0	0
C21	1	2	0	1	0	0	1	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	2	3	1	2	0	2	0	0
C22	0	0	0	1	0	0	3	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	1	1	5	0	1	0	0	0
C23	0	0	0	2	0	0	0	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0
C24	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C25	0	0	0	3	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C26	1	1	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	1	0	0	0	1	1	0	0
C27	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C28	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
C29	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C30	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	0	0	0	0	1	0	0
C31	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0
C32	3	0	0	0	1	0	1	0	0	0	0	3	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0
C33	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C34	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C35	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Legend: Same Capabilities IDs from Table 11 and Life Cycle Processes (LCPs) IDs from Table 13.

cloud native platform. Monitoring and observability practices provide system performance insight and assist ongoing improvement, while automated test code coverage maintains program reliability. Teams collaborate better when the tool chain is automated and integrated, promoting automation and continual development [23]. Furthermore, a strong DevOps culture requires security and compliance, leadership support, and service-level objectives. Scaling the SRE practice and configuration as code provides growth opportunities [25]. This helps identify DevOps adoption strategies and practices in this case study. DevOps maturity, workflow efficiency, and software development practice improvement can be achieved by emphasizing these strategies and initiatives [4].

4.6. RQ4: During the adoption of DevOps, which key benefits and challenges stand out and why?

In this section, the main benefits and challenges of DevOps adoption are discussed. Table 21 provides a comprehensive overview of key benefits, while Table 22 provides challenges observed during the adoption of DevOps within the organization.

Table 17: Relations summary for Capabilities and LCPs.

Capability	LCPs relations	Interviews
C01: Cross team collaboration and communication	Essential for **Architecture Definition (L20)**, requiring stakeholder involvement and consensus. Effective communication across teams is critical for effectively capturing and understanding **Stakeholder Needs and Requirements Definition (L18)**. In the **Knowledge Management process (L08)**, cross-team collaboration enhances knowledge exchange and management. Aids in **Project Planning (L09)** cooperation	I02, I04, I06, I09, I11, I13, I18, I19, I20, I22, I23, I25
C02: Support learning culture and experimentation	Fosters an environment where knowledge is continuously updated and shared, which is directly related to **Knowledge Management process (L08)**. A learning culture encourages people to learn and share information, which is important for improving DevOps capabilities	I02, I06, I16
C03: Open source software adoption	Referred to as a strategic move in the **Acquisition process (L01)**, as it offers cost-effective, flexible, and community-supported solutions that can be integrated into the DevOps lifecycle	I02, I05, I06, I14, I21, I22
C04: Transformational leadership	Important for **Decision Management (L11)**, as it involves guiding and inspiring teams to embrace change and innovation, which is essential for effective decision-making within the DevOps lifecycle	I09, I20, I22
C08: Proactive Monitoring, Observability and autoscaling	Related to **Operation (L28)**, as it enables active management and automatic adjustment of system resources to maintain optimal performance and reliability during operation	I06, I12, I25
C12: Visual management Capabilities	Support **Portfolio Management process (L05)** by providing visual tools and dashboards that help track the progress, status, and health of various projects within a portfolio, facilitating maturity assessment, better decision-making and resource allocation	I01, I05, I09, I17
C14: Focus on people, process and technology	Integral to **Stakeholder Needs and Requirements Definition (L18)**, ensuring that the needs of all stakeholders are considered and that the processes and technology are aligned to meet those needs effectively	I11, I16, I18
C18: Customer focus/feedback	Vital for **Stakeholder Needs and Requirements Definition (L18)**, as it ensures that the products or services being developed are closely aligned with customer expectations and market demands	I11, I16, I18
C20: Continuous Integration	Fundamental to **Integration (L24)**, of frequent code changes into a shared repository, with automated tests. It also supports **Quality Management process (L07)** by ensuring that quality is maintained through automated testing and **Implementation (L23)** by enabling smoother transitions of code into production	I01, I06, I11, I13, I14, I15, I18, I19, I21, I25, I26
C21: Continuous Delivery/Deployment automation	Streamlines **Integration (L24)** by automating the deployment process, ensuring that new features can be released quickly and reliably	I06, I14, I21
C22: Test Automation and environments	Crucial for **Verification (L25)** and **Quality Assurance (L16)**, as it allows consistent and efficient product testing. It also supports **Quality Management process (L07)** by ensuring that testing is an integral part of the development lifecycle	I04, I05, I12, I13, I14, I15, I16, I25, I26
C25: Configuration Management	Related to **Configuration Management (L13)** and **Infrastructure Management process (L04)**, as it involves maintenance and control of the product's configuration and infrastructure, ensuring consistency and reliability	I05, I12, I13, I14, I22
C26: Cloud infrastructure and cloud native	Improves **Infrastructure Management process (L04)** by providing scalable, flexible and resilient cloud infrastructure solutions	I06, I10, I14, I22, I25, I28
C30: Infrastructure as Code	Supports **Infrastructure Management process (L04)** by allowing infrastructure to be managed and provisioned through code, which increases efficiency and consistency	I06, I12, I14, I19, I25
C32: Shift left on security	Integrating security in early development relates to **Risk Management (L12)** by proactively managing security risks, and **Acquisition process (L01)** ensuring security considerations from the acquisition phase	I03, I04, I05, I07, I12, I23

Table 21: List of key benefits found.

Known Benefits	Mentions	Level
Velocity, Efficiency, and Delivery Performance	**18**	Exceptional
Customer success, Reliability and Quality	**13**	Exceptional
Consistency, Standardization, and Best Practices	**12**	Exceptional
Job satisfaction, Automation and reduced errors	10	High
Security and Compliance	4	High
Cost Reduction, Savings and Value	3	Increased
Monitoring and Observability	3	Increased
Improved Collaboration and Communication	2	Increased
Mindset Transformation for Growth	2	Increased
Diagnose problems	1	Reduced
Reducing Silos	1	Reduced
Sharing knowledge	1	Reduced
Simplify software delivery	1	Reduced
Start being mature on DevOps	1	Reduced
Transparency	1	Reduced
Understand failures easier	1	Reduced

Table 22: List of key challenges found.

Known Challenges	Mentions	Level
Resistance to Change	**13**	**Exceptional**
Prioritization of Capabilities and Tools	**12**	**Exceptional**
Skills and Knowledge	**10**	**High**
Cross dependency of teams	**6**	**High**
Documentation Gaps, Examples, and Videos	**4**	**High**
Pipeline migration from old methods	3	Increased
Experimenting Tooling	2	Increased
Human resourcing	2	Increased
Lack of understanding of the culture	2	Increased
Understanding DevOps concepts	2	Increased
Versatile Problem Solving for Unique Use Cases	2	Increased
Communication	1	Reduced
Deterministic builds	1	Reduced
Fear of failure	1	Reduced
Friction on security issues	1	Reduced
Integration tests	1	Reduced
Lack of distributed tracing	1	Reduced
Lack of visibility over resource consuming apps	1	Reduced
Parsing Output	1	Reduced
Understanding changes impact	1	Reduced
Visibility of new features	1	Reduced

Table 18: Top correlations mentioned between Metrics and Life Cycle Processes.

ID	L01	L02	L03	L04	L05	L06	L07	L08	L09	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20	L21	L22	L23	L24	L25	L26	L27	L28	L29	L30
M02	0	2	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	1	0	1	1	0	0	0
M03	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
M04	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
M05	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M06	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
M07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
M08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M09	0	0	0	0	0	0	5	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0
M10	1	1	0	0	0	0	4	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
M11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	2	0
M12	2	0	0	2	0	0	1	0	1	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	2	0
M13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	2	1	0
M14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M15	0	0	0	0	1	0	0	0	1	0	0	2	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	2	0	0
M16	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M17	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
M18	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
M19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M20	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M21	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
M22	1	0	0	0	0	2	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
M23	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
M24	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Legend: Same Life Cycle Processes (LCPs) IDs from Table 13 and Metrics IDs from Table 12.

Table 19: Relations summary for Metrics and LCPs.

Metric	LCPs relations	Interviews
M05: Defect Escape Rate	Related to **Quality Assurance (L16)** as it measures the rate at which defects pass through the QA process and are discovered only after the product is released. [27, 48] It is an indicator of the effectiveness of the QA process in identifying and solving issues prior to release	I02, I04, I14, I15, I17
M09: Pipeline Automated Tests Success Rate	Associated with **Quality Management process (L07)**. It indicates the percentage of automated tests that pass successfully and reflects the health and reliability of the codebase. A high success rate suggests that the quality management processes are effectively catching defects	I04, I14, I15, I19, I25
M10: Automated Test Code Coverage	**Quality Assurance (L16)** and **Quality Management process (L07)**. It measures the extent to which the codebase is covered by automated tests. High code coverage is indicative of a thorough testing process, which is crucial to both ensuring quality [27] and managing it throughout the development lifecycle	I02, I09, I13, I14, I15, I19, I25, I26

Table 20: Main Strategies and Initiatives.

Known Strategies and Initiatives	Mentions	Level
CI/CD Pipeline Adoption	**25**	Exceptional
Internal Catalog & Maturity Assessment	**9**	Exceptional
Common Cloud Native Platform	**7**	**High**
Automated Test Code Coverage	**5**	**High**
Monitoring and Observability	**5**	**High**
Toolchain Automation and Integration	**5**	**High**
Security and Compliance	4	Increased
Artifact Registry	3	Increased
Executive Support and Leadership	3	Increased
SLOs and Dashboards	3	Increased
Blameless postmortems	1	Reduced
Configuration as code	1	Reduced
Continuous Learning	1	Reduced
Dedicated chat channel	1	Reduced
DORA	1	Reduced
Scale SRE practice	1	Reduced

The adoption of DevOps within an organization presents both significant benefits and challenges. An increase in customer success, reliability, and quality are some of the benefits that can be found in the analysis of both tables. These improvements show that the company can make software faster and better. Customers are happier and the product is improved [10, 13]. Placing an emphasis on consistency and standardization can lead to more efficient, better software development and deployment processes overall, which in turn leads to improvement of operational efficiencies [53]. On the flip side, an organization has to face several challenges while adopting DevOps [43]. Poor culture and organizational transformation are among the primary ones [4, 38]. A company must train staff and collaborate across domains because of skills and knowledge gaps [24]. Furthermore, it is difficult to break old habits and adopt new tools. Recruiting and training staff, as well as fostering a collaborative and continuous improvement culture, are significant challenges in DevOps [30]. In addition, it is important to have effective communication channels and an organizational culture that facilitates learning and innovation [24]. In summary, DevOps benefits companies but requires them to change culture, learn new skills, and use new tools. To achieve sustainable growth and competitiveness in today's dynamic business landscape, organizations must effectively address challenges and leverage DevOps benefits [32].

5. Discussion

The following discussion of the IT company's DevOps adoption journey is based on the proposed conceptual framework in Section 2.4. It aims to give an overview of how the capabilities, metrics, LCPs, benefits, and challenges of DevOps adoption all affect its success with contributions to scientific knowledge.

DevOps Capabilities: The organization's main goals, such as CI/CD pipeline adoption, internal catalog, and maturity assessment, are in line with the conceptual framework's focus on DevOps skills that affect metrics and LCPs. These capabilities, particularly in the cultural and technical domains, were crucial in addressing the need for increased velocity, efficiency, and delivery performance [3, 14]. Resistance to change and the difficulty of integrating DevOps into existing processes highlight the importance of effective communication and follow-up skills. Senapathi et al. [43] proposed guided experimentation and continuous learning, which is strongly supported by open source [32], such as Cloud Native [4].

DevOps Metrics: The case study emphasizes the importance of DevOps metrics to assess the efficiency of the capabilities and processes implemented. The increase in the number of releases, the decrease in defects despite the increase in the number of releases, and the improvement in operational efficiency and software quality [5] These metrics, classified as change, operational, and business, provided tangible evidence of the benefits of DevOps adoption, which is consistent with the framework's categorization and the cited literature [4, 8]. Efforts to improve version control system capabilities for repository management led to an 88% improvement in service ownership and repository portfolio management, indicating a mature approach to software development and project management.

DevOps Life Cycle Processes: The adoption of standard CI/CD pipelines shows the influence of DevOps capabilities on lifecycle processes [46]. These projects made the SDLC more efficient and better, in line with the theory that DevOps skills shape LCPs and are important for using DevOps capabilities well. This approach, which aligns with Jabbari et al. [23], Shahin et al. [44], emphasizes the relevance of collaboration, constant improvement and commitment to operational excellence.

DevOps Benefits and Challenges: This case study presents the benefits and challenges of DevOps adoption [12, 13, 54]. Strategic actions improved efficiency, quality, and customer satisfaction [10], but also faced challenges like resistance to change and integration complexity. The results strongly suggest that there is a direct relationship between LCPs, metrics, and DevOps benefits and challenges. This gives us a good picture of what could be better and what could get in the way of adoption [41, 45].

Finally, the journey of DevOps adoption within this IT organization, as analyzed through the lens of the proposed conceptual framework, offers significant insights into the dynamics of DevOps adoption. The interdependencies between DevOps capabilities, metrics, LCPs and the resultant benefits and challenges highlight the need for product prioritization, information visibility, automation, and standardization to successfully adopt DevOps should be highly considered by organizations and their leaders.

6. Conclusion

This study analyzed the complexities of DevOps adoption in a software company, addressed the research problem — understanding the real-world application and interconnection of DevOps capabilities, metrics, and software lifecycle processes — and provided insights that improve our understanding of effective DevOps adoption. Also revealed key findings: high-impact DevOps capabilities and metrics are indispensable for success; lifecycle processes significantly influence DevOps outcomes; there were 38 symbiotic relationships between capabilities, metrics, and lifecycle processes found. Identified 16 major initiatives, benefits, challenges, and the fundamental role of transformational leadership in driving adoption. The findings also include:

self-service repository management increases efficiency and ownership; an internal catalog is vital for maturity assessment and microservice management; increased releases with decreased defects and MTTR; Open source, namely Cloud Native, is key for infrastructure management; And finally, the relationship between metrics and LCPs requires more investigation. These findings have broad implications. They provide guidance for organizations on their DevOps journey, emphasizing the need to prioritize high-impact capabilities and metrics. They also advocate for a holistic approach to DevOps adoption, stressing the importance of understanding and leveraging the relationship between DevOps capabilities, metrics, and lifecycle processes. Additionally, by acknowledging key challenges and benefits, organizations can prepare for potential issues while aligning their expectations with realistic outcomes.

6.1. Validity and limitations

Important methods were used to raise the reliability and solidity of the qualitative research. *Methodological Triangulation*: For findings verification, the researchers rely on semi-structured interviews, document analysis, and focus groups. Correction as well as completeness of data via annotations and member checking. *Focus Group Discussion*: To validate and refine interviews and document observations, an expert focus group was held. *Data triangulation and reliability*: Research consistency and reliability were achieved by triangulating interview, focus group, and document data. Audits and peer review for research transparency and accountability. *Validity*: The researchers focused on construct validity, external validity, and reliability to assure study accuracy, applicability, and consistency. Construct validity via a relevant example and writing detailed interview questions, and external validity on context-specific insights to avoid generalization. *Limitations and Ethical Considerations*: The study was conducted within one of the authors' organizations, offering unique insights that may not be found in external investigations. A rigorous methodology that ensures objectivity and dependability helps to reduce the potential conflict of interest. Other potential biases, company context, and sample size were noted in the study. To assure ethics, the researchers acquired informed consent, maintained confidentiality, and triangulated data to better understand the research.

6.2. Future work

Future research could build on this work in a number of areas relevant to improving the understanding and application of DevOps. In future evaluations, it should be explored how companies can overcome cultural resistance during DevOps changes. Align metrics and LCPs: The differences between metrics and LCPs are worthy of investigation. Research in the future could deepen on adoption of DevOps influencing productivity, customer satisfaction and innovation. Open source for DevOps success: Open source, particularly Cloud Native technologies, is essential for infrastructure management. Future studies could therefore focus on optimizing the impact of open source tools. Standardization and collaboration: DevOps priorities such as visibility, standardization issues and benefits should be studied.

References

[1] Akbar, M.A., Rafi, S., Alsanad, A.A., Qadri, S.F., Alsanad, A., Alothaim, A., 2022. Toward successful DevOps: A decision-making framework. IEEE access : practical innovations, open solutions 10, 51343–51362. doi:10.1109/ACCESS.2022.3174094.

[2] Alamin, M.A.A., Uddin, G., Malakar, S., Afroz, S., Haider, T., Iqbal, A., 2022. Developer discussion topics on the adoption and barriers of low code software development platforms. Empirical Software Engineering 28. doi:10.1007/s10664-022-10244-0.

[3] Amaro, R., Pereira, R., Mira da Silva, M., 2022. Capabilities and Practices in DevOps: A Multivocal Literature Review. IEEE Transactions on Software Engineering 1, 20. doi:10.1109/TSE.2022.3166626.

[4] Amaro, R., Pereira, R., da Silva, M.M., 2023. Capabilities and Metrics in DevOps: A Design Science Study. Information & Management , 32doi:10.1016/j.im.2023.103809.

[5] Amaro, R., Pereira, R., da Silva, M.M., 2024. DevOps Metrics and KPIs: A Multivocal Literature Review. ACM Computing Surveys doi:10.1145/3652508.

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

[7] Budgen, D., Brereton, P., 2006. Performing systematic literature reviews in software engineering, in: Proceedings of the 28th International Conference on Software Engineering, Keele University and Durham University Joint Report, New York, NY, USA. pp. 1051–1052. doi:10.1145/1134285.1134500.

[8] Davis, J., Daniels, R., 2016. Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale. "O'Reilly Media, Inc.", USA.

[9] Debois, P., 2011. DevOps from a Sysadmin Perspective. Login - The Usenix Magazine 36, 3.

[10] Di Gangi, P., Wasko, M., Hooker, R., 2010. Getting Customers' Ideas to Work for You: Learning from Dell how to Succeed with Online User Innovation Communities. MIS Quarterly Executive 9. URL: https://aisel.aisnet.org/misqe/vol9/iss4/4.

[11] Díaz, J., Almaraz, R., Pérez, J., Garbajosa, J., 2018. DevOps in practice - An exploratory case study, in: Proceedings of the 19th International Conference on Agile Software Development: Companion, Agile Alliance; Agile Portugal; ScaleUp Porto, Universidad Politécnica de Madrid, CITSEM, Madrid, Spain. p. 3. doi:10.1145/3234152.3234199.

[12] Díaz, J., López-Fernández, D., Pérez, J., González-Prieto, Á., 2021. Why are many businesses installing a DevOps culture into their organization? Empirical Software Engineering 26, 50. doi:10.1007/s10664-020-09919-3, arXiv:2005.10388.

[13] Faustino, J., Adriano, D., Amaro, R., Pereira, R., da Silva, M.M., 2022. DevOps benefits: A systematic literature review. Software: Practice and Experience 52, 1905–1926. doi:10.1002/spe.3096.

[14] Forsgren, N., Kersten, M., 2018. DevOps Metrics. Communications of the ACM 61, 44–48. doi:10.1145/3159169.

[15] Fusch, P., Fusch, G., Ness, L., 2018. Denzin's Paradigm Shift: Revisiting Triangulation in Qualitative Research. Journal of Sustainable Social Change 10. doi:10.5590/JOSC.2018.10.1.02.

[16] Garousi, V., Felderer, M., Mäntylä, M.V., 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology 106, 101–121. doi:10.1016/j.infsof.2018.09.006, arXiv:1707.02553.

[17] Ghantous, G.B., Gill, A.Q., 2017. DevOps: Concepts, practices, tools, benefits and challenges, in: 21st Pacific Asia Conference on Information Systems: Societal Transformation Through IS/IT, PACIS 2017, Association for Information Systems, School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia. p. 12. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85075594314&partnerID=40&md5=dd1517b2fb59b692caf9f3099a61aa1b.

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

[19] Hemon, A., Lyonnet, B., Rowe, F., Fitzgerald, B., 2020. From Agile to DevOps: Smart Skills and Collaborations. Information Systems Frontiers 22, 927–945. doi:10.1007/s10796-019-09905-1.

[20] Hemon-Hildgen, A., Rowe, F., Monnier-Senicourt, L., 2020. Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk and work conditions. European Journal of Information Systems 29, 474–499. doi:10.1080/0960085X.2020.1782276.

[21] Humble, J., Molesky, J., 2011. Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal 24, 6–12.

[22] IEEE, 2021. IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021. IEEE Std 2675-2021 1, 91. doi:10.1109/IEEESTD.2021.9415476.

[23] Jabbari, R., bin Ali, N., Petersen, K., Tanveer, B., 2016. What is DevOps? A Systematic Mapping Study on Definitions and Practices, in: Proceedings of the Scientific Workshop Proceedings of XP2016, ACM, New York, NY, USA. p. 11. doi:10.1145/2962695.2962707.

[24] Kankanhalli, A., Tan, B.C., Wei, K.K., 2005. Contributing knowledge to electronic knowledge repositories: An empirical investigation. MIS Quarterly: Management Information Systems 29, 113–143. doi:10.2307/25148670.

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

[26] Kitchenham, B., Charters, S., 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report. Technical report, ver. 2.3 ebse technical report. ebse.

[27] Kumar, R., Goyal, R., 2020. Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). Computers and Security 97, 101967. doi:10.1016/j.cose.2020.101967.

[28] Kumara, I., Garriga, M., Romeu, A.U., Di Nucci, D., Palomba, F., Tamburri, D.A., van den Heuvel, W.J., 2021. The do's and don'ts of infrastructure code: A systematic gray literature review. Information and Software Technology 137, 106593. doi:10.1016/j.infsof.2021.106593.

[29] Laudon, J.P.L..K.C., 2017. Management Information Systems: Managing the Digital Firm, Global Edition. Pearson Education, USA.

[30] Leite, L., Rocha, C., Kon, F., Milojicic, D., Meirelles, P., 2019. A survey of DevOps concepts and challenges. ACM Computing Surveys 52, 35. doi:10.1145/3359981, arXiv:1909.05409.

[31] Lévy, L.N., Bosom, J., Guerard, G., Amor, S., Bui, M., Tran, H., 2022. DevOps Model Appproach for Monitoring Smart Energy Systems. Energies 15, 27. doi:10.3390/en15155516.

[32] Lindberg, A., Schecter, A., Berente, N., Hennel, P., Lyytinen, K., 2024. The Entrainment of Task Allocation and Release Cycles in Open Source Software Development: MIS Quarterly. MIS Quarterly 48, 67–93. doi:10.25300/MISQ/2023/16789.

[33] Lopez-Fernandez, D., Diaz, J., Garcia-Martin, J., Perez, J., Gonzalez-Prieto, A., 2021. DevOps Team Structures: Characterization and Implications. IEEE Transactions on Software Engineering , 1doi:10.1109/TSE.2021.3102982, arXiv:2101.02361.

[34] Luz, W.P., Pinto, G., Bonifácio, R., 2019. Adopting DevOps in the real world: A theory, a model, and a case study. Journal of Systems and Software 157, 110384. doi:10.1016/j.jss.2019.07.083.

[35] McDonagh, Deana, J.L., 2019. Focus Groups: Supporting Effective Product Development. CRC Press, London. doi:10.4324/9780203302743.

[36] Mishra, A., Otaiwi, Z., 2020. Devops and Software Quality: A Systematic Mapping. Computer Science Review 38, 14. doi:10.1016/j.cosrev.2020.100308.

[37] Perez, J., Gonzalez-Prieto, A., Diaz, J., Lopez-Fernandez, D., Garcia-Martin, J., Yague, A., 2022. DevOps Research-based Teaching Using Qualitative Research and Inter-Coder Agreement. IEEE Transactions on Software Engineering 48, 3378–3393. doi:10.1109/TSE.2021.3092705.

[38] Rafi, S., Akbar, M.A., Yu, W., Alsanad, A., Gumaei, A., Sarwar, M.U., 2022. Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis. Applied Soft Computing 116, 108377. doi:10.1016/j.asoc.2021.108377.

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

[40] Raworth, K., Sweetman, C., Narayan, S., Rowlands, J., Hopkins, A., 2012. Conducting Semi-Structured Interviews. Oxfam.

[41] Rodríguez, P., Haghighatkhah, A., Lwakatare, L.E., Teppola, S., Suomalainen, T., Eskeli, J., Karvonen, T., Kuvaja, P., Verner, J.M., Oivo, M., 2017. Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software 123, 263–291. doi:10.1016/j.jss.2015.12.015.

[42] Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14, 131. doi:10.1007/s10664-008-9102-8.

[43] Senapathi, M., Buchan, J., Osman, H., 2018. 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, ACM. Association for Computing Machinery, New York, USA. pp. 57–67. doi:10.1145/3210459.3210465.

[44] Shahin, M., Ali Babar, M., Zhu, L., 2017. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE Access 5, 3909–3943. doi:10.1109/ACCESS.2017.2685629.

[45] Smeds, J., Nybom, K., Porres, I., 2015. DevOps: A Definition and Perceived Adoption Impediments, in: Lecture Notes in Business Information Processing. Springer, USA. volume 212, pp. 166–177. doi:10.1007/978-3-319-18612-2_14.

[46] Standard, I., 2017. ISO/IEC/IEEE International Standard - Systems and software engineering – Software life cycle processes. ISO/IEC/IEEE 12207:2017(E) First edition 2017-11 , 157doi:10.1109/IEEESTD.2017.8100771.

[47] Stephen Eldridge, 2024. Percentile | Definition, Quartile, & Facts | Britannica. URL: https://www.britannica.com/topic/percentile.

[48] Teixeira, D., Pereira, R., Henriques, T., Silva, M.M.D., Faustino, J., 2020. A maturity model for DevOps. International Journal of Agile Systems and Management 13, 464. doi:10.1504/IJASM.2020.112343.

[49] Trigo, A., Varajão, J., Sousa, L., 2022. DevOps adoption: Insights from a large European Telco. Cogent Engineering 9. doi:10.1080/23311916.2022.2083474.

[50] Verner, J.M., Babar, M.A., Cerpa, N., Hall, T., Beecham, S., 2014. Factors that motivate software engineering teams: A four country empirical study. Journal of Systems and Software 92, 115–127. doi:10.1016/j.jss.2014.01.008.

[51] Waseem, M., Liang, P., Shahin, M., 2020. A Systematic Mapping Study on Microservices Architecture in DevOps. Journal of Systems and Software 170. doi:10.1016/j.jss.2020.110798.

[52] Westrum, R., 2004. A typology of organisational cultures. Quality and Safety in Health Care 13, 22–27. doi:10.1136/qshc.2003.009522.

[53] Wiedemann, A., Wiesche, M., Gewald, H., Krcmar, H., 2020. Understanding how DevOps aligns development and operations: A tripartite model of intra-IT alignment. European Journal of Information Systems 29, 458–473. doi:10.1080/0960085X.2020.1782277.

[54] Winkler, F., Westner, M., 2023. A Systematic Literature Review of DevOps Success Factors and Adoption Models, in: Proceedings of the 12th International Symposium on Information and Communication Technology, Association for Computing Machinery, New York, NY, USA. pp. 525–532. doi:10.1145/3628797.3628883.

[55] Yin, R.K., 2018. Case Study Research and Applications: Design and Methods. Sixth edition ed., SAGE, Los Angeles.

Table A.23: Questions used in the Semi-structured interviews

ID	Related RQs	Type	Question
1			**Background Information**
1.1	-	Open-ended	Can you please describe your role in the organization and your experience?
1.2	-	Open-ended	How do you see your current DevOps adoption?
2			**DevOps Capabilities Impact**
2.1	RQ1	Closed-ended	What DevOps capabilities have the most positive impact on the company's DevOps Adoption?
2.2	RQ1	Open-ended	Please describe those capabilities.
2.3	RQ1	Open-ended	Do you know how they are implemented, and evaluated?
3			**DevOp Metrics Impact**
3.1	RQ1	Closed-ended	What DevOps metrics have the most positive impact on the company's DevOps Adoption?
3.2	RQ1	Open-ended	Please describe those metrics.
3.3	RQ1	Open-ended	Do you know how they are implemented, and evaluated?
3.4	RQ2	Closed-ended	What DevOps metrics have a direct correlation to main capabilities?
3.5	RQ2	Open-ended	Why are they correlated?
4			**Life Cycle Processes Impact**
4.1	RQ1	Closed-ended	What Life Cycle Processes (LCPs) had the most positive impact on the DevOps adoption?
4.2	RQ1	Open-ended	Please describe those LCPs.
4.3	RQ1	Open-ended	Do you know how they are implemented, and evaluated?
4.4	RQ2	Closed-ended	What LCPs have a direct correlation to main capabilities?
4.5	RQ2	Closed-ended	What LCPs have a direct correlation to main metrics?
4.6	RQ2	Open-ended	Why are they correlated?
5			**Strategies and initiatives for DevOps adoption**
5.1	RQ3	Closed-ended	Do you know what strategies or initiatives have been implemented for DevOps adoption?
5.2	RQ3	Open-ended	Which inititiatives most impact your day-to-day work?
6			**DevOps adoption Challenges and Benefits**
6.1	RQ4	Open-ended	What are the main benefits and challenges of DevOps in the organization?
6.2	RQ4	Open-ended	How do these benefits and challenges impact your daily work?
7			**Ending**
7.1	-	Closed-ended	On a scale of 1-10, how would you rate the DevOps adoption effort in the organization?
7.2	-	Open-ended	Do you have any last comments?

Appendix B. Invitation Letter

Invitation to participate in a case study research.

Hello {name},
I am excited to invite you to participate in an interview for our case study research project within the context of our DevOps adoption.
Your insights and experiences will be invaluable in helping us better understand the adoption of DevOps Capabilities and Metrics within the organization's Software Life Cycle Processes. I will conduct the interview, which will last for about 45 minutes, via videoconference. Please consider using the following link to schedule the interview: {link}
If you know of anyone else who might be interested in attending, please forward this invitation to them as well.
Best regards,
{researcher name}

Figure B.11: Invitation letter to participate in the case study research.

CHAPTER 8

Conclusion

This last chapter contains a summary of the key contributions based on the considerations in Section 1.3. The key takeaways are broken down into four sections that provide guidance and potential directions for future research. The *summary and discussion* section explains the results of the articles in accordance with addressing our main research question and the key problem for this thesis, stated in Section 1.2. It also discusses the key concepts and presents a framework for improving DevOps adoption success in Table 8.1 aimed to address the observed key challenges of DevOps adoption. This provides, at the same time, a practical and comprehensive overview of the key results. The *closing remarks* provide valuable insights and contributions regarding the practical and theoretical applications of these findings. In Section 8.3 the limitations of this work are explained and mitigations are given. The final section of the chapter, titled *future work* outlines potential areas for further research and topics that should be explored after this work.

8.1 Summary and Discussion

This thesis aims to propose and evaluate solutions to achieving successful DevOps adoption in IT Organizations. Considering this main goal, in this section, the research questions of this thesis are summarized and discussed.

8.1.1 RQ1: What are the key DevOps capabilities, metrics, and processes that have the most positive impact on DevOps adoption?

In order to summarize the research conducted to address this question, an overview is provided for each of the three key vectors, detailing the individual contributions made by each study to advance the research and knowledge for this thesis.

Key DevOps Capabilities The resulting key capabilities are listed, categorized and discussed in Article 1 of Chapter 2 DevOps capabilities are stated as essential skills and knowledge required to perform specific practices effectively within DevOps. In Tables 2.7 and 2.8 a comprehensive list of 37 Capabilities, their definitions, and their dynamic nature over time. The information about capabilities can be found in key figures and tables including the definition of DevOps capability in Table 2.10, conceptual map proposed in Figure 2.10, as well as the relationship among capabilities, practices and outcomes provided in Figure 2.12. In addition, it highlights that capabilities are dynamic through some findings from both academic and industry perspectives to reach a consensus.

Article 2 (Chapter 3) discusses DevOps capabilities through the improvement of key performance indicators and highlights the need to use appropriate DevOps Metrics to determine the performance of these capabilities.

In Article 3 (Chapter 4) the DevOps capabilities are approached through Design Science Research (DSR) methodology building on top of MLR and semi-structured interviews, leading to 37 DevOps capabilities classified, and validated. Table 4.13 summarizes the relations between capabilities and metrics, while Table 4.17 proposes a validated artifact with categorized DevOps capabilities influencing main metrics.

DevOps capabilities in Article 4, are discussed from foundational areas including culture, automation, lean methods, measurement and sharing. Important points to note include enhanced collaboration, quicker time to market, more automation improved management of deployment environment, better stability as well as increased security. These challenges are mainly related to cultural preparedness, communication problems, and automation technology complexities.

In Article 5 (Chapter 6) DevOps capabilities are approached by identifying, classifying, and mapping them to the 30 Life Cycle Processes (LCPs) in the IEEE DevOps standard [23]. This is shown in Table 6.7 for the total number of relations between categories of DevOps Capabilities and LCPs and in Table 6.7 with the total number of relations and average number of relations for each LCP category. A strong correlation was found between Technical DevOps capabilities and technical LCPs. It was found significant impact of Measurement Capabilities and Agreement Processes shown in Table 6.10 and exceptional impact in eight relations, other eleven have very high impact from Table 6.12 The study highlights and tries to address challenges in adopting DevOps Capabilities effectively.

Finally, Article 6 (Chapter 7) defines DevOps capabilities as essential elements that facilitate the adoption of DevOps practices within an organization. DevOps Capabilities are approached through the CALMS framework: Culture, Automation, Lean, Measurement, and Sharing. The conceptual framework in Figure 7.4 and Table 7.4 illustrates the relationships between DevOps capabilities, metrics, LCPs, and benefits/challenges. Capabilities such as cross-team collaboration (C01), proactive monitoring (C08), continuous improvement (C13), continuous integration (C20), and cloud infrastructure (C26) are highlighted as being crucial for successful DevOps adoption in this case study. The study also identifies areas for improvement, like database change management (C29) and chaos engineering (C36). DevOps capabilities are highlighted by improved collaboration, monitoring, continuous improvement, and technical practices. While the most important challenges are Resistance to Change, Prioritization of Capabilities and Tools, Skills and Knowledge, Cross dependency of teams, and Documentation Gaps. The Key strategies observed that contributed to successful DevOps adoption are CI/CD Pipeline, Internal Catalog & Maturity Assessment, Common Cloud Native Platform, Automated Test Code Coverage, Monitoring and Observability and Tool chain Automation and Integrations.

Key DevOps metrics In Article 1 (Chapter 2) DevOps metrics are briefly discussed as means of assessing maturity levels, visualize management processes, and understand relationships between concepts. Metrics can be used to face DevOps adoption challenges by providing visual management tools, assessing maturity levels, ensuring continuous improvement through feedback loops, and making informed business decisions.

The main article focusing on DevOps metrics is Article 2 (Chapter 3). Here a DevOps metrics definition is proposed as a "Quantifiable, business-relevant, trustworthy, actionable, and traceable indicators that aid organizations in making data-driven decisions to continuously improve their DevOps and software delivery process". In Figure 3.7 Key 22 DevOps Metrics are elicited from the MLR, while Figure 3.9 proposes steps to put DevOps metrics into practice.

Article 3 (Chapter 4) validates 37 DevOps capabilities and 24 metrics. The study emphasizes the importance of these metrics for management to attain the success and efficiency in implementing DevOps. While the MLR identified 22 key metrics out of an initial 58, the interviews strongly emphasized two additional metrics, resulting in 24 listed in Table 4.14. The relation of key metrics like MLT, DF, CFR, MTTR, between others with key capabilities are described per their significant importance in this regard. Details on each metric relevance and importance, examples, and categories are provided. All key DevOps metrics are identified and validated.

Article 4 of Chapter 5 goes into detail on measuring tasks and finding the improvements. It discusses the importance of measurement and monitoring within DevOps. The utilization of infrastructure as code and monitoring tools facilitates the management and identification of issues within teams.

In Article 5 (Chapter 6) metrics are primarily seen under the umbrella of "Measurement capabilities", defined as those focusing on collecting and analyzing data about software development and delivery. Key aspects of measurement capabilities include proactive monitoring, observability, autoscaling, emergency response, visual management (dashboards), real-time analytics for tracking application health and usage metrics. DevOps metrics are important for a variety of processes: acquisition, supply, infrastructure management, portfolio management, resource management, and quality management. The study states that real-time feedback and measurement-driven design should be practiced for these processes. Measures relevant to this process included automated checks for Service level Indicators (SLIs) and tests that ensure that the service is ready to deliver.

In Article 6 (Chapter 6) DevOps metrics are approached as crucial factors for adoption, categorized into change, operational, business and cultural metrics. The detailed categorization of metrics is seen Figure 7.2, and in Table 7.12 interview results related to metrics are synthesized, and Figure 9 (graphs showing improvements in version control system capabilities).2 Metrics provide tangible evidence of benefits such as increased release frequency, decreased escaped defects, improved operational efficiency, and better software quality. The study highlights specific metrics such as DF, MTTR, CFR, and Automated Test Code Coverage among others. In this case study, metrics help identify the bottlenecks to address first. Promoting a culture of shared responsibility, teamwork and departmental efficiency, speeding up features and bug fixes.

Key DevOps processes In Article 1 (Chapter 2) DevOps Life Cycle Processes are approached indirectly through a framework that emphasizes collaboration, continuous improvement, and visual management. Key figures and tables about Life Cycle Processes include Figures 2.10 and 2.12. Life Cycle Processes can be used to face DevOps adoption challenges by enhancing team skills, improving continuous integration, and using visual management to assess maturity levels. The main takeaways from this reading are that collaboration, continuous improvement, and specific capabilities help in the achievement of the desired outcomes.

In Article 2 (Chapter 3), monitoring process is referred to by the authors who present the process through an eight-step process forming a continuous feedback loop. In the particular case provided in the article, the monitoring process is reflected in the DevOps infinity loop in Figure 3.9. Process is considered a way to face the DevOps adoption challenges by applying a serious procedure providing a mechanism to measure, a systematic approach to measuring the performance, identifying needs, automating to decrease human error, and ensuring continuous feedback and collaboration.

The Article 3 (Chapter 4) approaches process by evaluating and improving organizational practices through an improvement roadmap, while not explicitly mentioning LCPs. Provides a understanding of the planning process of using DevOps capabilities and metrics, for the success of DevOps adoption.

In Article 4 (Chapter 5) processes focus on improving collaboration between development and operations teams through culture, automation, lean methodologies, measurement, and sharing. Using CI/CD, operations teams are involved early on, automation, continuous monitoring, and infrastructure-as-code. These factors help solve challenges by reducing manual errors, ensuring stability, detecting issues early, and standardizing configurations.

In Article 5 (Chapter 6) Life Cycle Processes (LCPs) are introduced by mapping 37 identified DevOps capabilities to the 30 Life Cycle Processes (LCPs) defined in the IEEE 2675-2021 standard [23]. The mappings of DevOps capabilities to LCPs in Tables 6.3 to 6.6, the impact scale classifications in Table 6.12, and the conceptual maps showing relationships between capabilities and processes Figures 6.1 and 6.9 reveal that ways that LCPs together with the positively impactful capabilities can be used to face DevOps adoption challenges. In other words, by determining which capabilities enhance specific LCPs, organizations can commit to the implementation of these capabilities to enhance processes and reap the benefits of software delivery, quality, and reliability. Nonetheless, the anticipated challenges in terms of cultural, technical, measurement, and process capabilities suggest the significance of guidelines, ongoing reflection and adaptation.

Article 6 (Chapter 7) is a case study research using a three-vector approach including DevOps LCPs, capabilities, and metrics. It builds upon the IEEE Standard 2675-2021 standard [23] to organize LCPs and stresses the value of LCPs in defining, modeling, and assessing software processes, underscoring the conclusions of the previous research in this thesis about the incremental and iterative approach of DevOps. The case study provides in-depth knowledge about the benefits and challenges of a real world adoption of DevOps, as well as the factors, such as the implementation of a full scale standard CI/CD pipeline, Configuration Management, or Automated Test Code Coverage. Additionally, it demonstrates how DevOps capabilities affect LCPs. The challenges, such as resistance to change, integration complexity, or prioritization, necessitate the implementation of capabilities. These include transformational leadership, CI/CD or visual management capabilities for improved SDLC efficiency and quality. The study concludes that DevOps capabilities affect metrics and LCPs, which in turn influence benefits and challenges.

8.1.2 RQ2: How can organizations effectively apply key DevOps capabilities, metrics, and processes to overcome adoption challenges?

The studies have shown that successful DevOps adoption in organizations is possible through team effort, best practices, technical and cultural aspects, team ownership, tool chain usage, management commitment, and process improvement models [87, 92–94]. However, the challenges, according to new studies in adopting DevOps, mention them as related to the technology, people and, most importantly, the inability to communicate in the organization [17, 38, 52, 67, 95, 96]. In addition, the analysis from the studies in Chapters 2 to 7 reveal that various challenges are regularly faced in DevOps adoption, like resistance to change, cultural transformation, skill & knowledge gaps, integration complexity, tooling & automation prioritization, security concerns, communication & collaboration, cross dependency of teams, and measurement & monitoring.

However, it is also observed, mainly from Chapters 4 and 7, that drawbacks can be controlled or even overcome with the right strategies in place, like the **implementation of CI/CD pipeline** for establishing a continuous integration and continuous delivery to automate the software delivery process. An **internal catalog and maturity assessment** for improving ownership, communication, and conducting maturity assessments to identify areas for growth and enable continuous improvement. A **common Cloud Native platform** to promote microservices, scalability, and agility between teams. **Monitoring and observability** practices are used to gain insight into system performance and encourage continuous improvement. Ensuring **automated test code coverage** to maintain software quality and reliability. A **tool chain automation and integration** to enhance collaboration and promote continuous development. Emphasizing **security and compliance** to build a robust DevOps culture. Gaining strong **leadership support** to drive the DevOps transformation. Establishing **Service level Objectives (SLOs)** in pair with **scaling Site Reliability Engineering (SRE) practice** to support growth, ensuring operational excellence and reliability. Implementing **Configuration as Code** to manage infrastructure and application configurations efficiently.

Improving DevOps Adoption Success The broader answer to this fundamental question lies within the findings and proposal across the research done in the several articles presented. The most important concepts from the research done in the previous chapters revolve around DevOps capabilities, metrics, life cycle processes, benefits, challenges and strategies, which are relationships captured in of Figure 1.2. Therefore, this thesis proposes a Framework for Improving DevOps Adoption Success in Figure 8.1 based on these concepts and substantiated by the research presented.

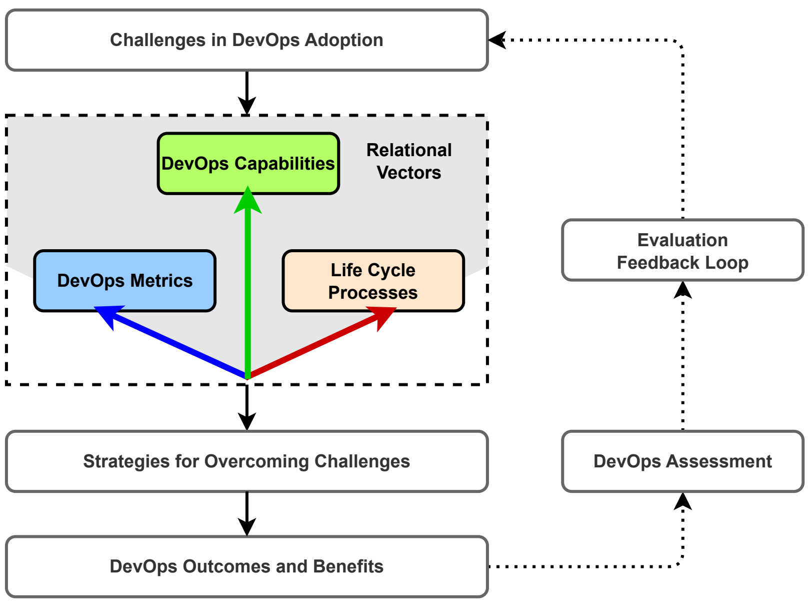


Figure 8.1: Proposed Framework for Improving DevOps Adoption Success

A framework typically offers a systematic way to understand and address complex issues like the DevOps adoption challenges. This framework has the specific purpose of gaining an overview of these challenges, translating them into implementable strategies, while justifying actions based on their impact. In accordance to the nature of DevOps, the framework is iterative and includes a DevOps assessment on each iteration, leading to an evaluation feedback loop. In each iteration, one or more vectors should be utilized to address a specific challenge, depending on the objective and resources available, while keeping it to a low number of vectors in order to reduce the task complexity.

The most important and related key takeaways from what has been studied can now be synthesized using the proposed framework of DevOps adoption. The resulting instantiation of the framework is shown in Table 8.1 aiming to improve successful DevOps adoption success. It provides a structured approach to addressing specific challenges found in DevOps adoption by linking them to relevant vectors of capabilities, metrics, or life cycle processes, supported by empirical evidence from the studies in Chapters 2 to 7.

The framework's strategies and justifications are based on a careful examination of results

Table 8.1: Instantiation of the Framework for Improving DevOps Adoption Success

Challenge	Vector	Strategy	Justification	Source
Resistance to change	Capability	Transformational Leadership (C04)	Promotes empowerment according to Westrum [97], fostering a supportive and innovative work environment for talent retention.	A1, A3, A4, A6
	Metric	Team Happiness (M21)	A culture that encourages learning and experimentation contributes to an engaging and fulfilling work environment.	A3, A4, A6
	Life Cycle Process	Project Planning (L09)	Helps manage and mitigate resistance to change by setting clear expectations and providing structured guidance.	A5, A6
Integration complexity	Capability	Continuous Integration and Delivery (CI/CD) (C20,C21)	Building and testing software components are automated, reducing the complexities of integration of code changes by many contributors and delivering them.	A1, A3, A6
	Metric	Mean Lead-time for Changes (MLT) (M02)	Lead time for changes is measured and reduced highlighting and remedying integration bottleneck	A2, A6
	Life Cycle Process	Configuration Management (L13)	Configuration Management ensures that all elements are controlled and managed across the life cycle, reducing complexity.	A5, A6
Skills and knowledge gaps	Capability	Support Learning Culture and Experimentation (C02)	Encouraging a learning culture and experimentation helps close skills and knowledge gaps by promoting continuous learning and improvement.	A1, A3, A4, A6
	Metric	Talent Retention (M22)	A supportive learning culture can help retain talent by providing opportunities for growth and development.	A3, A6
	Life Cycle Process	Knowledge Management Process (L08)	Ensures team members have access to the information and training they need.	A5, A6
Cultural transformation	Capability	Cross-team Collaboration and Communication (C01)	Effective collaboration and communication lead to a more satisfied and cohesive team, essential for cultural transformation.	A1, A3, A6
	Metric	Team Happiness (M21)	Measuring and improving team happiness can indicate a positive cultural transformation.	A2, A6
	Life Cycle Process	Quality Management Process (L07)	Ensures products, services, and implementations meet quality goals and satisfy customers, which accelerates cultural change.	A5, A6
Tooling and automation challenges	Capability	Cloud infrastructure and cloud native (C26)	Ensures highly automated microservices platform, reducing manual effort and improving reliability.	A3, A4, A6
	Capability	Proactive Monitoring, Observability, and Autoscaling (C08)	Help to identify and resolve issues earlier, for effective tooling and automation.	A1, A3, A6
	Metric	Automated Test Code Coverage (M10)	Automated test code coverage ensures that code is automatically tested reducing any manual test effort, increasing quaility and reliability.	A3, A6
	Life Cycle Process	Configuration Management (L13)	Ensures tools and environments are consistent and reliable.	A5, A6
Prioritization and visibility issues	Capability	Visual Management Capabilities (C37)	Visual management helps the organization communicate goals, change, and prioritize improvements.	A1, A3, A4, A6
	Metric	Work in Progress (WIP) Limits (M20)	Setting and monitoring Work in Progress (WIP) limits helps teams focus on key work and avoid overload.	A2, A6
	Life Cycle Process	Project Planning (L09)	The principle of project planning and creating a workable plan is important to prioritize and for visibility.	A5, A6
Security concerns	Capability	Shift Left on Security (C20)	Security is best integrated earlier reducing vulnerabilities and problems.	A1, A3, A4, A6
	Metric	Change Failure Rate (CFR) (M04)	Security vulnerabilities can be known if the change failure rate is continuously monitored.	A2, A3, A6
	Life Cycle Process	Risk Management (L12)	Knowing how to consistently handle development and operational risks, ensures system security can be addressed.	A5, A6
Communication and collaboration	Capability	Cross-team Collaboration and Communication (C01)	Enhances collaboration and communication, while breaking down the silos.	A1, A3, A4, A6
	Metric	Team Happiness (M21)	Improved communication and collaboration lead to higher team happiness and job satisfaction.	A2, A3, A6
	Life Cycle Process	Stakeholder Needs and Requirements Definition (L18)	Ensures that all stakeholders' needs and requirements are clearly defined and communicated, facilitating better collaboration.	A5, A6
Measurement and monitoring	Capability	Proactive Monitoring, Observability, and Autoscaling (C08)	Ensures continuous monitoring and observability of systems to detect and resolve issues proactively.	A1, A3, A4, A6
	Metric	Mean Time To Detection (MTTD) (M07)	Reducing the time to detect issues helps in quicker resolution and maintaining system reliability.	A2, A3, A6
	Life Cycle Process	Information Management (L14)	Generates, obtains, confirms, transforms, retains, retrieves, disseminates, and disposes of relevant information for stakeholders.	A5, A6

and findings, in order to ensure its solidity and credibility. Its main purpose is to tackle the research issue through providing applicability advice for organizations that want to adopt DevOps effectively, which serves as the most central feature of this framework.

This instantiation uses the data obtained from the research to provide a structured approach to addressing the identified challenges in the context of DevOps adoption. As such, it includes the relational vectors that are capabilities, metrics, and life cycle processes, and uses strategies that allow addressing the identified DevOps challenges as well as justification and sources for each proposed strategy.

In order to mitigate the discovered challenges of **resistance to change**, the development of transformational leadership capability (C04) that promotes empowerment and work friendly atmosphere should become a key focus area as noted by Westrum (2004) [97]. Additionally, measuring team happiness metric (M21) shows how a culture of learning and experimentation can affect someone, while project planning (L09) helps mitigate resistance to change through setting reasonable expectations. The challenge of **integration complexity** is addressed through CI/CD (C20,C21) that is related to integration issues caused by building and testing automation while focusing on faster delivery. The MLT metric is vital in addressing the identified challenge since it highlights the bottlenecks for remediation, while configuration management (L13) ensures control of what is involved in the life cycle in all phases. The gap in **skills and knowledge** is addressed through supporting a learning culture and experimentation (C02) by leveraging a knowledge management (L08) life cycle process that ensures access to the required training and information and can be measured by the talent retention (M22) metric. The challenge of **cultural transformation** is addressed through the life cycle process of cross-team collaboration and communication (C01) that is followed by the team happiness (M21) metric, while quality management (L07) ensures the product meets quality goals.

The challenges of **tooling and automation** are addressed through a common Cloud Native platform (C26), proactive monitoring (C08) and automated test code coverage (M10). Notably, in this case based on the findings of Chapter 7, we start by using two capabilities in the first iteration of the framework to tackle this challenge, which can be re-evaluated in the next framework's feedback loop. The use of tooling configuration management (L13) is essential for ensuring the reliability of the tools and environments to reduce variability. **Prioritization and visibility** are addressed through visual management capabilities (C37) while WIP limits (M20) are used to allow the developers to focus on the work ahead. Project planning (L09) because it is essential to prioritize and visibility. **Security concerns** are accounted for by security left (C20) and monitoring CFR (M04), where risk management (L12) is responsible for treating development risks as operational risks also in a unified way. Improving on **Communication and collaboration** though cross-team collaboration (C01) as well as defining stakeholders needs and requirements definitions (L18) increases team happiness (M21).

Finally, **measurement and monitoring** are addressed through proactive monitoring (C08) and reduced Mean Time To Detection (MTTD) (M07) by following the information management

(L14) life cycle process to properly manage the information required by the stakeholders. This framework instantiation demonstrates various ways in which specific related capabilities, metrics, and life cycle processes are used to incorporate specific strategies for successful DevOps successful strategies.

8.2 Closing Remarks

At the end of this research, It can be stated that it is possible to improve DevOps adoption by using the implementation strategies that are supported by the 37 capabilities [90] and 24 metrics [28], along with the 30 life cycle processes [23]. The relationships explored and analyzed, increase the theoretical and practical knowledge in the field of this discipline and clearly point out drivers that make it easier for organizations to adopt DevOps more effectively, quickly, and in a way that can be consolidated.

A few important factors are transversal to the research, like **visual management communication** [90], together with a continuous **feedback loop** is an effective mean for monitoring and evaluating the maturity and adoption of DevOps, making it possible to identify processes within the software delivery life cycle that are suitable or in need of improvement. The importance of **planning** with careful preparation and strategy are needed before starting to adopt DevOps. Good planning requires a good knowledge of the organization and its challenges, as seen in Chapter 7. The idea is that a thorough understanding of the organization's own capabilities when compared to the full scope of available DevOps capabilities [90] is essential for success. The organization must be **flexible** and able to adapt itself and its teams to changes in the environment, security, and market challenges. Hence, it is imperative for leaders to exhibit transformative qualities, have appreciation for generative culture, as exemplified by Westrum (2004) [97], and exhibit fluidity in their decisions and strategies. The quality of **leadership** is considered a critical factor for success. A good leader is described as wise, sincere, benevolent, courageous, and determined. On the other hand, organizations adopting DevOps should keep a check on **resource optimization** to avoid unnecessary expenditure of resources. Go straight to the supporting capabilities based on existing metrics and processes has discussed in Chapters 3 and 6. In this sense, knowledgeable and strategic use of **automation** is essential to optimize time and investment. Finally, it is seen in Chapter 7 that **cross team collaboration** is a basis for **performance and quality**, where speed of development and delivery is a significant advantage. While keeping quality factors at high standards avoids situations such as failures in production, critical vulnerabilities and to minimize the need for **incident response** which has a considerable negative impact on **job satisfaction** leading to a fall in **talent retention**. In the end, this essential collaboration may require a strong **cultural shift** undertaking. However, maintaining a generative organizational culture and collaboration between teams is decisive for successful DevOps adoption. This thesis explores three fundamental vectors of DevOps: capabilities, metrics and processes, while providing new insights and proposing key strategies for adopting

DevOps successfully in organizations.

8.3 Limitations

Most of the identified limitations of this study are explained at the end of each article, since each one has its own particular limitations. The case study is based on a single case, which could not be generalized to other organizations or other industries in particular. In order to overcome this limitation, triangulation is performed and future research on other case studies should be conducted in a number of other industries with diverse background and locations to validate and generalize the findings even more. Two of the literature review studies are based on gray literature, which is less rigorous compared to peer-reviewed research. For that, the guidelines by Garousi et al. (2019) [98] were used, and more empirical research was done to validate the capabilities and metrics identified in the MLRs. The study may have limitations because of the inability to reflect the dynamic nature of DevOps capabilities and metrics and the rapid pace of development of the latter. To address this limitation, future research should conduct longitudinal studies that would imply regular tracking of DevOps capabilities and metrics and constant updating of results. Another study limitation is the generalizability of the framework for improving the success of DevOps adoption. In the future, the research should adapt the framework to various organizational types on the basis of pilot implementation and feedback collection for further adjustments. Finally, the inclusion of sources in English could impact the outcomes because relevant research in other languages was not considered.

8.4 Future work

In this section, are mentioned several possible areas for further research. Future work ideas for DevOps research can encompass the following:

Integration of microservices and DevOps: Future research might explore how DevOps could be incorporated into microservices architectures. Microservices currently increase scalability and enhance performance, and hence they help to reduce time taken in continuous integration and delivery process. Case studies could be done in many industries, whether the microservices bring explicit benefits, restrictions, and limitations.

Cloud-Native DevOps practices: Future research may investigate cloud-native technologies' adoption in a DevOps environment. The study may focus on determining how platforms and tools like Kubernetes, Docker, Istio, ArgoCD etc., that are cloud-native can facilitate automation of infrastructure provisioning, resource utilization as well as operational reliability.

Artificial intelligence application in DevOps: Another area to look into is how AI/ML is used in DevOps. Such studies may focus on the utilization of AI/ML algorithms for resource allocation optimizations, anticipating future system failures, or automating tasks. Furthermore, research may be conducted into the application of AI-based monitoring and alert systems that detect anomalies and make notifications about them.

Cross-country case studies: Case studies on DevOps acceptance in various countries are possible areas for future investigation. In this light, the role played by cultural, legal and financial factors within the jurisdiction has to be given priority in making such considerations. It is important to note that comparative analyses can also be done amongst developed as well as developing nations.

Industry-specific DevOps practices: The studies in industry-related practices such as DevOps in healthcare, DevOps in finance, DevOps in manufacturing, and others may also be an interesting research topic for future work. The development of Devops framework for each of such industries and the case studies conducted at enterprise, SME and startup levels may give a vast picture of DevOps performance in these environments

Longitudinal Studies of DevOps Maturity: It may be interesting and beneficial to conduct a longitudinal study of DevOps maturity with periodic interval time comparing research results. Future research could be focused on the stages that the various DevOps maturity has in an organization and the movement of the organization from stage to stage. The study could analyze the impact of the continued improvements and the effect of the leadership of the organization on the organizational culture.

Lastly, **Security and compliance in DevOps:** May be a future research field to explore, such as security protocols in which are sometimes called DevSecOps. The focus of studies may be placed on the shifted-left security practices, automated commission control strategies, and secure data handling and management in cloud-native environment. Case studies focusing on DevSecOps implementations may highlight both successful cases and almost nonexistent adoption of such practices in highly regulated industries.

References

[1] P. Debois, “Agile infrastructure and operations: How infra-gile are you?” in *Proceedings - Agile 2008 Conference*, 2008, pp. 202–207.

[2] A. Mishra and Z. Otaiwi, “Devops and Software Quality: A Systematic Mapping,” *Computer Science Review*, vol. 38, no. 1, p. 14, Nov. 2020.

[3] J. P. L. . K. C. Laudon, *Management Information Systems: Managing the Digital Firm, Global Edition*. USA: Pearson Education, 2017.

[4] M. Senapathi, J. Buchan, and 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*, ser. EASE’18, no. June, ACM. New York, USA: Association for Computing Machinery, Jun. 2018, pp. 57–67.

[5] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. USA: Addison-Wesley Professional, Dec. 2010.

[6] D. N. Blank-edelman, *Seeking SRE: Conversations About Running Production Systems at Scale*. USA: O’Reilly Media, Inc., 2018.

[7] P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen, and P. Kuvaja, “Advances in Using Agile and Lean Processes for Software Development,” in *Advances in Computers*, M. A.M., Ed. Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland: Academic Press Inc., 2019, vol. 113, pp. 135–224.

[8] P. Perera, M. Bandara, I. Perera, and IEEE, “Evaluating the Impact of DevOps Practice in Sri Lankan Software Development Organizations,” in *2016 SIXTEENTH INTERNATIONAL CONFERENCE ON ADVANCES IN ICT FOR EMERGING REGIONS (ICTER) - 2016*, no. 16th International Conference on Advances in ICT for Emerging Regions (ICTer). Institute of Electrical and Electronics Engineers Inc., 2016, pp. 281–287.

[9] E. Diel, S. Marczak, and D. S. Cruzes, “Communication challenges and strategies in distributed DevOps,” in *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*. Computer Science School, PUCRS, Porto Alegre, RS, Brazil: Institute of Electrical and Electronics Engineers Inc., Aug. 2016, pp. 24–28.

[10] G. Kim, K. Behr, K. Spafford, and G. Spafford, *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*. USA: IT Revolution, 2014. [Online]. Available: https://books.google.pt/books?id=H6x-DwAAQBA

[11] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “An Exploratory Study of DevOps Extending the Dimensions of DevOps with Practices,” in *ICSEA 2016 : The Eleventh International Conference on Software Engineering Advances*, 2016.

[12] J. Smeds, K. Nybom, and I. Porres, “DevOps: A Definition and Perceived Adoption Impediments,” in *Lecture Notes in Business Information Processing*. USA: Springer, 2015, vol. 212, pp. 166–177.

[13] I. Bucena and M. Kirikova, “Simplifying the DevOps Adoption Process Challenges of DevOps Adoption. In BIR Workshoops.” *Simplifying the DevOps Adoption Process*, 2017.

[14] D. Teixeira, R. Pereira, T. A. Henriques, M. Silva, and J. Faustino, “A Systematic Literature Review on DevOps Capabilities and Areas,” *International Journal of Human Capital and Information Technology Professionals*, vol. 11, no. 2, p. 22, Apr. 2020.

[15] M. A. Akbar, S. Mahmood, M. Shafiq, A. Alsanad, A. A. A. Alsanad, and A. Gumaei, “Identification and prioritization of DevOps success factors using fuzzy-AHP approach,” *Soft Computing*, 2020.

[16] A. Qumer Gill, A. Loumish, I. Riyat, and S. Han, “DevOps for information management systems,” *VINE Journal of Information and Knowledge Management Systems*, vol. 48, no. 1, pp. 122–139, Jan. 2018.

[17] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A survey of DevOps concepts and challenges,” *ACM Computing Surveys*, vol. 52, no. 6, p. 35, Nov. 2019.

[18] L. E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo, and C. Lassenius, “DevOps in practice: A multiple case study of five companies,” *Information and Software Technology*, vol. 114, no. March 2017, pp. 217–230, 2019.

[19] J. Díaz, D. López-Fernández, J. Pérez, and Á. González-Prieto, “Why are many businesses installing a DevOps culture into their organization?” *Empirical Software Engineering*, vol. 26, no. 2, p. 50, 2021.

[20] M. Muñoz and M. N. Rodríguez, “A guidance to implement or reinforce a DevOps approach in organizations: A case study,” *Journal of Software: Evolution and Process*, vol. 1, p. 21, 2021.

[21] N. Forsgren, M. C. Tremblay, D. VanderMeer, and J. Humble, “DORA Platform: DevOps Assessment and Benchmarking,” in *Designing the Digital Transformation*, A. Maedche, J. vom Brocke, and A. Hevner, Eds. Cham: Springer International Publishing, 2017, pp. 436–440.

[22] N. Forsgren and M. Kersten, “DevOps Metrics,” *Communications of the ACM*, vol. 61, no. 4, pp. 44–48, Dec. 2018.

[23] IEEE, “IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021,” *IEEE Std 2675-2021*, vol. 1, no. 16 Apr 2021, p. 91, 2021.

[24] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, “DevOps adoption benefits and challenges in practice: A case study,” in *Product-Focused Software Process Improvement*, vol. 10027 LNCS. Department of Computer Science, University of Helsinki, Gustaf Hällströmin katu 2b, P.O. Box 68, Helsinki, 00014, Finland: Springer International Publishing, 2016, pp. 590–597.

[25] W. P. Luz, G. Pinto, and R. Bonifácio, “Adopting DevOps in the real world: A theory, a model, and a case study,” *Journal of Systems and Software*, vol. 157, no. July, p. 110384, Nov. 2019.

[26] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, “Towards a benefits dependency network for DevOps based on a systematic literature review,” *Journal of Software: Evolution and Process*, vol. 30, no. 11, p. 26, Nov. 2018.

[27] J. Faustino, R. Amaro, D. Adriano, Daniel, R. Pereira, and M. M. da Silva, “DevOps benefits: A systematic literature review,” *Software: Practice and Experience*, vol. 52, no. 9, pp. 1905–1926, 2022.

[28] R. Amaro, R. Pereira, and M. M. da Silva, “Capabilities and Metrics in DevOps: A Design Science Study,” *Information & Management*, p. 32, May 2023.

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

[30] J. Davis and R. Daniels, *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. USA: "O’Reilly Media, Inc.", May 2016.

[31] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, “What is DevOps? A Systematic Mapping Study on Definitions and Practices,” in *Proceedings of the Scientific Workshop Proceedings of XP2016*. New York, NY, USA: ACM, May 2016, p. 11.

[32] R. Kneuper, *Software Processes and Life Cycle Models: An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes*. Cham: Springer International Publishing, 2018.

[33] F. M. A. Erich, C. Amrit, and M. Daneva, “A Qualitative Study of Devops Usage in Practice,” *Journal of Software: Evolution and Process*, vol. 29, no. 6, p. e1885, 2017.

[34] M. Rodriguez, L. J. P. De Araújo, and M. Mazzara, “Good practices for the adoption of DataOps in the software industry,” in *Journal of Physics: Conference Series*, A. K. Kruglov A., Ed., vol. 1694. IOP Publishing Ltd, 2020.

[35] R. W. Macarthy and J. M. Bass, “An Empirical Taxonomy of DevOps in Practice,” in *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, M. A., W. M., and S. A., Eds., IEEE. University of Salford, School of Science, Engineering and Environment, Manchester, United Kingdom: Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 221–228.

[36] K. Maroukian, “Towards practice and principle adoption through continuous DevOps leadership.” *Journal of Software. Ruanjian Xuebao*, vol. 16, no. 1, p. 13, 2021.

[37] D. Teixeira, R. Pereira, T. Henriques, M. M. D. Silva, and J. Faustino, “A maturity model for DevOps,” *International Journal of Agile Systems and Management*, vol. 13, no. 4, p. 464, 2020.

[38] S. Rafi, W. Yu, M. A. Akbar, A. Alsanad, and A. Gumaei, “Multicriteria based decision making of DevOps data quality assessment challenges using fuzzy TOPSIS,” *IEEE Access*, vol. 8, no. 1, pp. 46 958–46 980, 2020.

[39] K. Maroukian and S. R. Gulliver, "Leading DevOps Practice and Principle Adoption," in *9th International Conference on Information Technology Convergence and Services (ITCSE 2020)*, vol. 13. AIRCC Publishing Corporation, May 2020, pp. 41–56.

[40] M. F. Lie, M. Sanchez-Gordon, and R. Colomo-Palacios, "DevOps in an ISO 13485 regulated environment: A multivocal literature review," in *International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: ACM, Oct. 2020, p. 11.

[41] L. Yin and V. Filkov, "Team Discussions and Dynamics during DevOps Tool Adoptions in OSS Projects," in *Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 697–708.

[42] M. Zarour, N. Alhammad, M. Alenezi, and K. Alsarayrah, "A research on DevOps maturity models," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 4854–4862, Sep. 2019.

[43] J. Perez, A. Gonzalez-Prieto, J. Diaz, D. Lopez-Fernandez, J. Garcia-Martin, and A. Yague, "DevOps Research-based Teaching Using Qualitative Research and Inter-Coder Agreement," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3378–3393, Sep. 2022.

[44] R. Kumar and R. Goyal, "Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC)," *Computers and Security*, vol. 97, p. 101967, Oct. 2020.

[45] R. Cherinka, S. Foote, and J. Prezzama, "Lessons learned in adopting agile software development at enterprise scale," in *WMSCI 2020 - 24th World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings*, S. B. S. M. Baracho R. Callaos N.C., Ed., vol. 1. International Institute of Informatics and Systemics, IIIS, 2020, pp. 68–73. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85096545508&partnerID=40&md5=2774e8455ae5129ec7d46edcaf8c1022

[46] U. Zdun, E. Wittern, and P. Leitner, "Emerging Trends, Challenges, and Experiences in DevOps and Microservice APIs," *IEEE Software*, vol. 37, no. 1, pp. 87–91, 2020.

[47] DevOps Research and Assessment (DORA), "State of DevOps 2019 - DORA," DORA, Tech. Rep. DORA2019, 2019. [Online]. Available: https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

[48] Puppet Labs, "2019 State of DevOps Report," Puppet Labs, Tech. Rep. 2019, 2019. [Online]. Available: https://puppet.com/resources/report/2019-state-of-devops-report

[49] ——, "2020 State of DevOps Report," Puppet Labs, Tech. Rep. 2020, 2020. [Online]. Available: https://puppet.com/resources/report/2020-state-of-devops-report/

[50] C. Bansal, S. Renganathan, A. Asudani, O. Midy, and M. Janakiraman, "DeCaf: Diagnosing and triaging performance issues in large-scale cloud services," in *Proceedings - International Conference on Software Engineering*. Microsoft Research, Redmond, WA, United States: IEEE Computer Society, 2020, pp. 201–210.

[51] E. Bernard, F. Ambert, and B. Legeard, “Supporting efficient test automation using lightweight MBT,” in *Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2020*. 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE; IEEE Comp Soc; Farfetech; Facebook; VMWare; Msg Life; Google; New Work Se, 2020, pp. 84–94.

[52] M. A. Akbar, W. Naveed, S. Mahmood, A. A. Alsanad, A. Alsanad, A. Gumaei, and A. Mateen, “Prioritization Based Taxonomy of DevOps Challenges Using Fuzzy AHP Analysis,” *IEEE Access*, vol. 8, pp. 202 487–202 507, 2020.

[53] S. Li, Q. Xu, P. Hou, X. Chen, Y. Wang, H. Zhang, and G. Rong, “Exploring the Challenges of Developing and Operating Consortium Blockchains: A Case Study,” in *ACM International Conference Proceeding Series*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 398–404.

[54] R. Colomo-Palacios, E. Fernandes, P. Soto-Acosta, and X. Larrucea, “A case analysis of enabling continuous software deployment through knowledge management,” *International Journal of Information Management*, vol. 40, pp. 186–189, Jun. 2018.

[55] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect’s Perspective*, ser. SEI Series in Software Engineering. New York: Addison-Wesley, 2015. [Online]. Available: http://my.safaribooksonline.com/9780134049847

[56] A. Alnafessah, A. U. Gias, R. Wang, L. Zhu, G. Casale, and A. Filieri, “Quality-aware DevOps research: Where do we stand?” *IEEE access : practical innovations, open solutions*, vol. 9, pp. 44 476–44 489, 2021.

[57] J. Roche, “Adopting DevOps practices in quality assurance,” *Communications of the ACM*, vol. 56, no. 11, pp. 38–43, Nov. 2013.

[58] B. Snyder and B. Curtis, “Using Analytics to Guide Improvement during an Agile-DevOps Transformation,” *IEEE Software*, vol. 35, no. 1, pp. 78–83, Jan. 2017.

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

[60] A. Ravichandran, K. Taylor, and P. Waterhouse, *DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory*. USA: Springer Nature, 2016.

[61] M. Farshchi, J. G. Schneider, I. Weber, and J. Grundy, “Metric selection and anomaly detection for cloud operations using log and metric correlation analysis,” *Journal of Systems and Software*, vol. 137, pp. 531–549, 2018.

[62] D. Sun, M. Fu, L. Zhu, G. Li, and Q. Lu, “Non-Intrusive Anomaly Detection with Streaming Performance Metrics and Logs for DevOps in Public Clouds: A Case Study in AWS,” *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 278–289, Apr. 2016.

[63] M. A. Akbar, K. Smolander, S. Mahmood, and A. Alsanad, “Toward successful DevSecOps in software development organizations: A decision-making framework,” *Information and Software Technology*, vol. 147, p. 106894, Jul. 2022.

[64] M. Waseem, P. Liang, and M. Shahin, “A Systematic Mapping Study on Microservices Architecture in DevOps,” *Journal of Systems and Software*, vol. 170, 2020.

[65] W. John, G. Marchetto, F. Nemeth, P. Skoldstrom, R. Steinert, C. Meirosu, I. Papafili, and K. Pentikousis, “Service provider DevOps,” *IEEE Communications Magazine*, vol. 55, no. 1, pp. 204–211, Jan. 2017.

[66] T. Laukkarinen, K. Kuusinen, and T. Mikkonen, “DevOps in regulated software development: Case medical devices,” in *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track, ICSE-NIER 2017*. Tampere University of Technology, Tampere, Finland: Institute of Electrical and Electronics Engineers Inc., 2017, pp. 15–18.

[67] R. Anandya, T. Raharjo, and A. Suhanto, “Challenges of DevOps Implementation : A Case Study from Technology Companies in Indonesia,” in *2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS*, Oct. 2021, pp. 108–113.

[68] A. W. Miller, R. E. Giachetti, and D. L. Van Bossuyt, “Challenges of Adopting Devops for the Combat Systems Development Environment,” *Defense Acquisition Research Journal: A Publication of the Defense Acquisition University*, vol. 29, no. 1, pp. 22–49, Jan. 2022.

[69] A. A. Ur Rahman, L. Williams, A. A. U. Rahman, and L. Williams, “Software security in DevOps: Synthesizing practitioners’ perceptions and practices,” in *1st International Workshop on Continuous Software Evolution and Delivery, CSED 2016*. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 70–76.

[70] H. Myrbakken and R. Colomo-Palacios, “DevSecOps: A multivocal literature review,” *Communications in Computer and Information Science*, vol. 770, no. 1, pp. 17–29, 2017.

[71] A. Al-marsy, P. Chaudhary, and J. Rodger, “A model for examining challenges and opportunities in use of cloud computing for health information systems,” *Applied System Innovation*, vol. 4, no. 1, p. 20, 2021.

[72] C. P. Bezemer, S. Eismann, V. Ferme, J. Grohmann, R. Heinrich, P. Jamshidi, W. Shang, A. Van Hoorn, M. Villavicencio, J. Walter, and F. Willnecker, “How is performance addressed in DevOps? A survey on industrial practices,” in *ICPE 2019 - Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 45–50.

[73] A. Hemon, B. Fitzgerald, B. Lyonnet, and F. Rowe, “Innovative Practices for Knowledge Sharing in Large-Scale DevOps,” *IEEE Software*, vol. 37, no. 3, pp. 30–37, 2020.

[74] I. Kumara, M. Garriga, A. U. Romeu, D. Di Nucci, F. Palomba, D. A. Tamburri, and W.-J. van den Heuvel, “The do’s and don’ts of infrastructure code: A systematic gray literature review,” *Information and Software Technology*, vol. 137, p. 106593, Sep. 2021.

[75] N. Forsgren and J. Humble, “DevOps: Profiles in ITSM Performance and Contributing Factors,” *SSRN Electronic Journal*, p. 25, 2015.

[76] J. Díaz, J. E. Pérez, M. A. Lopez-Peña, G. A. Mena, and A. Yagüe, “Self-service cybersecurity monitoring as enabler for DevSecops,” *IEEE Access*, vol. 7, no. 1, pp. 100 283–100 295, 2019.

[77] E. E. Romero, C. D. Camacho, C. E. Montenegro, Ó. E. Acosta, R. G. Crespo, E. E. Gaona, and M. H. Martínez, "Integration of DevOps Practices on a Noise Monitor System with CircleCI and Terraform," *ACM Transactions on Management Information Systems*, vol. 13, no. 4, pp. 36:1–36:24, Aug. 2022.

[78] F. Erich, C. Amrit, and M. Daneva, "Cooperation between information system development and operations: A literature review," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '14. New York, NY, USA: Association for Computing Machinery, Sep. 2014.

[79] A. Trigo, J. Varajão, and L. Sousa, "DevOps adoption: Insights from a large European Telco," *Cogent Engineering*, vol. 9, no. 1, 2022.

[80] Cambridge, "CULTURE | meaning in the Cambridge English Dictionary," 2021. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/culture

[81] João Faustino, Rúben Pereira, Bráulio Alturas, and Miguel Mira Da Silva, "Agile information technology service management with DevOps: An incident management case study," *International Journal of Agile Systems and Management*, vol. 13, no. 4, p. 339, 2020.

[82] A. Hemon-Hildgen, F. Rowe, and L. Monnier-Senicourt, "Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk and work conditions," *European Journal of Information Systems*, vol. 29, no. 5, pp. 474–499, Sep. 2020.

[83] W. P. Luz, G. Pinto, and 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.

[84] A. A. Khan and M. Shameem, "Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process," *Journal of Software: Evolution and Process*, vol. 32, no. 10, Oct. 2020.

[85] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, Jan. 2017.

[86] J. Angara, S. Gutta, and S. Prasad, "Devops with Continuous Testing Architecture and Its Metrics Model," in *Advances in Intelligent Systems and Computing*, vol. 709. K.L. University, Vijayawada, AP, India: Springer Verlag, 2018, pp. 271–281.

[87] L. Marrero and H. Astudillo, "DevOps-RAF: An assessment framework to measure DevOps readiness in software organizations," in *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*. Chile: IEEE, Nov. 2021, p. 8.

[88] F. Helwani and 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)*, Mar. 2022, pp. 198–205.

[89] K. Kuusinen, V. Balakumar, S. C. Jepsen, S. H. Larsen, T. A. Lemqvist, A. Muric, A. Ø. O. Nielsen, and O. Vestergaard, "A large agile organization on its journey towards DevOps,"

in *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018*, B. T. and A. L., Eds. University of Southern Denmark, Odense, Denmark: Institute of Electrical and Electronics Engineers Inc., Aug. 2018, pp. 60–63.

[90] R. Amaro, R. Pereira, and M. Mira da Silva, "Capabilities and Practices in DevOps: A Multivocal Literature Review," *IEEE Transactions on Software Engineering*, vol. 1, p. 20, 2022.

[91] R. Amaro, R. Pereira, and M. M. da Silva, "DevOps Metrics and KPIs: A Multivocal Literature Review," *ACM Computing Surveys*, Mar. 2024.

[92] S. Martínez-Fernández, A. M. Vollmer, A. Jedlitschka, X. Franch, L. López, P. Ram, P. Rodríguez, S. Aaramaa, A. Bagnato, M. Choraś, and J. Partanen, "Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study," *IEEE Access*, vol. 7, pp. 68 219–68 239, 2019.

[93] S. Rafi, W. Yu, M. A. Akbar, S. Mahmood, A. Alsanad, and A. Gumaei, "Readiness model for DevOps implementation in software organizations," *Journal of Software: Evolution and Process*, vol. 33, no. 4, Oct. 2021.

[94] G. Rong, H. Zhang, and D. Shao, "CMMI guided process improvement for DevOps projects: An exploratory case study," in *Proceedings - International Conference on Software and System Process, ICSSP 2016*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 76–85.

[95] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges," *Journal of Computer Information Systems*, vol. 60, no. 5, pp. 428–436, 2020/09/02/Number 5/September 2020.

[96] J. A. V. M. K. Jayakody and W. Wijayanayake, "Challenges for adopting DevOps in information technology projects," in *2021 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. Colombo, Sri Lanka: IEEE, Sep. 2021, pp. 203–210.

[97] R. Westrum, "A typology of organisational cultures," *Quality and Safety in Health Care*, vol. 13, no. SUPPL. 2, pp. 22–27, 2004.

[98] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, no. September 2018, pp. 101–121, Feb. 2019.