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.

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,

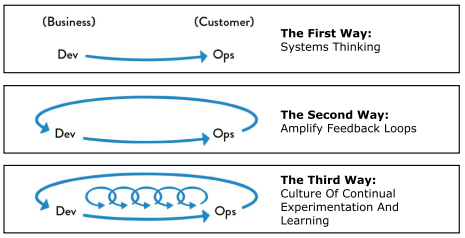

Fig. 1. The Three Ways: The principles underpinning DevOps (adapted) [65].

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.

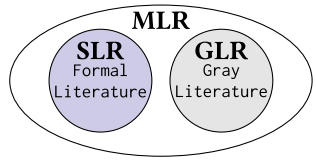

Fig. 2. Relationship of SLR, GLR, and MLR [63].

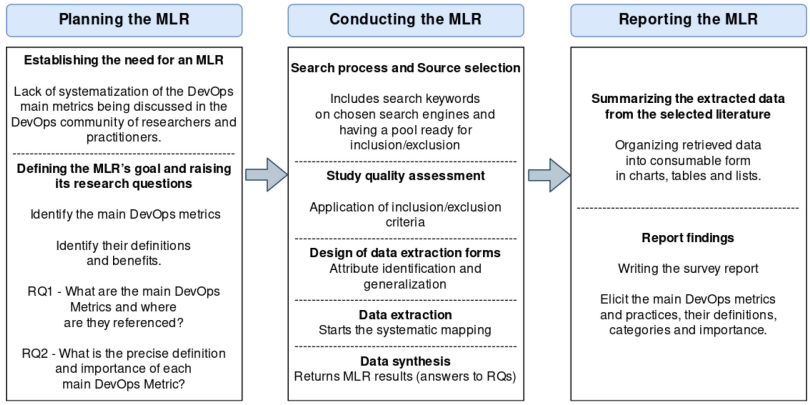

Fig. 3. MLR steps based on Garousi et al. [63].

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

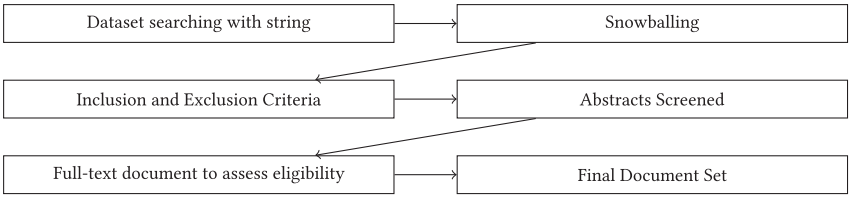

Fig. 4. Review protocol performed in this research.

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.

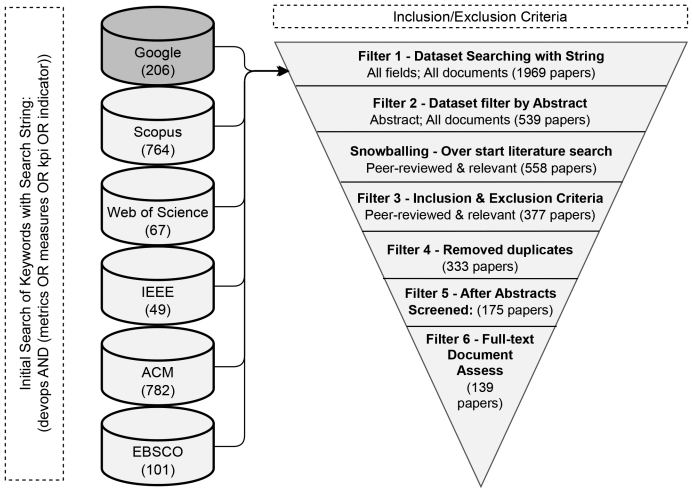

Fig. 5. Followed multivocal literature review process (adapted) [63].

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	+19	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	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.

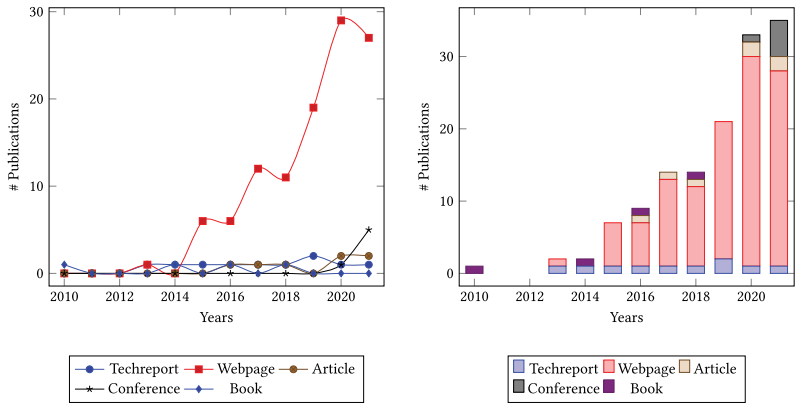

Fig. 6. Distribution of publications per type over the years.

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

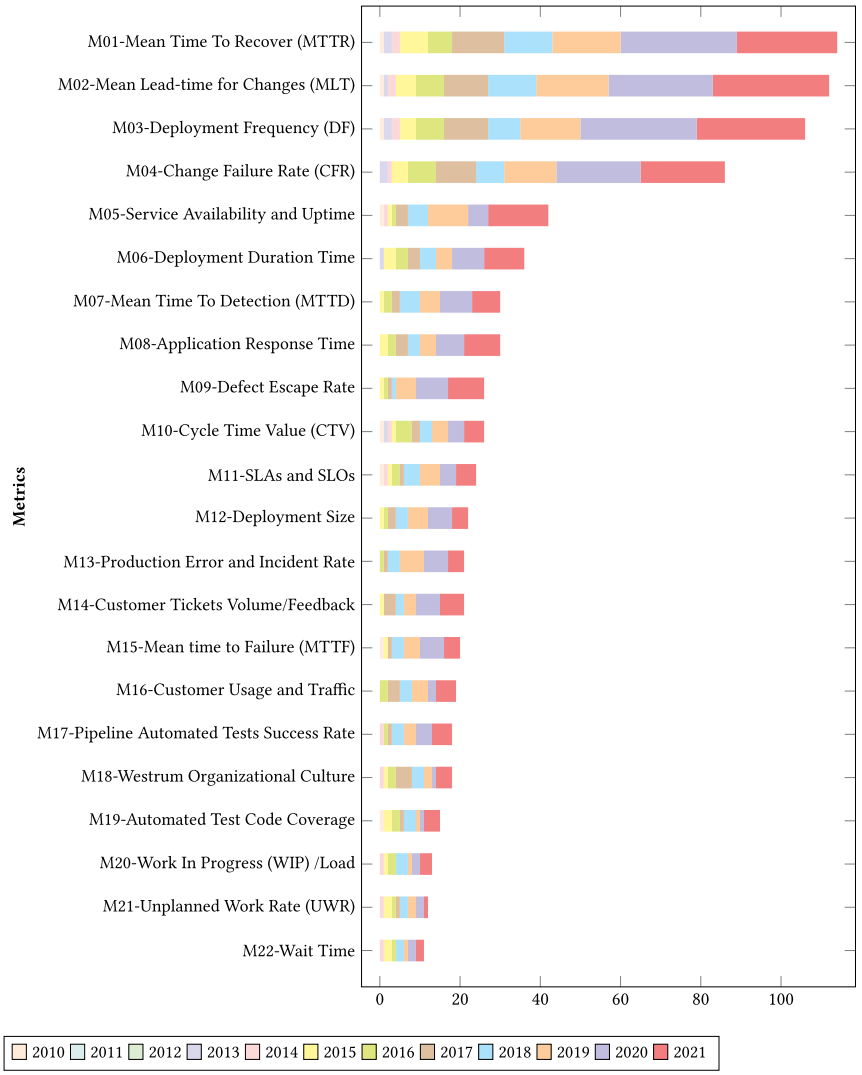

Fig. 7. Top main metrics mentioned in publications over the years.

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.

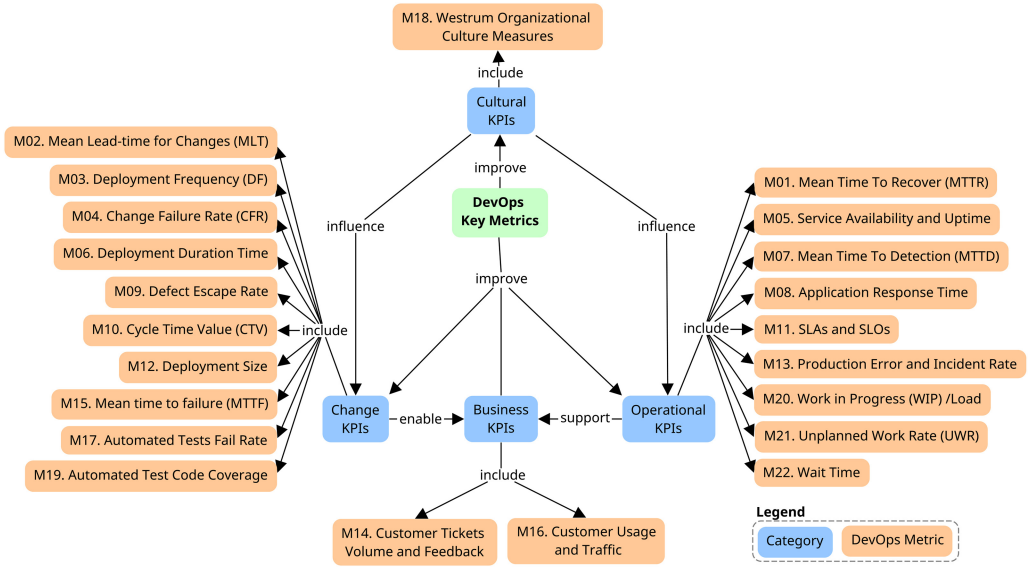

Fig. 8. DevOps metrics categorization and relation of concepts.

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

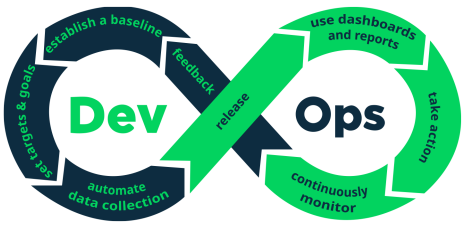

Fig. 9. DevOps metrics in practice infinity loop.

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

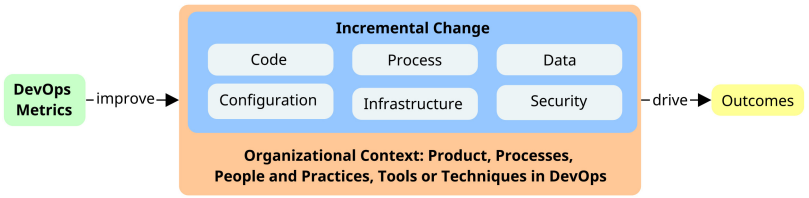

Fig. 10. DevOps metrics and incremental changes in the DevOps context.

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