DevOps Metrics, Objectives and Key Results

Seyedhamidreza Ziaei

A thesis submitted to

the Auckland University of Technology

in partial fulfilment of the degree of

Master of Computer and Information Sciences (MCIS)

2022

School of Engineering, Computer, and Mathematical Sciences

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning."

_____ Signature
25 Apr 2022

Abstract

The 21st century has been characterized by advancements in technology, one of them being the development and growth of DevOps as set of practices that combines software development and IT operations. DevOps is concerned with combining both the Development and Operations teams to increase cross-functional communication and the speed at which software solutions are delivered to the market. Management thinker, Peter Drucker, is often quoted as saying that 'you cannot manage what you cannot measure'. By this, Drucker meant that an organisation cannot know if it is successful unless it defines and tracks success. With a clearly established metric for success, an organization can quantify progress and adjust their process to produce more desired outcomes. DevOps metrics are therefore used to measure the efficiency of DevOps pipeline, identify, and remove defects in the process and aid in measuring the success organization. Despite the rise of DevOps as a set of practices, tools, and philosophical cultures that aid in delivering software to the market faster, the literature shows a little empirical research in the area of DevOps metrics, and their capabilities, and this warranted further study to identify opportunities for improvements.

In pursuit of identifying opportunities for improvements in DevOps metrics, the thesis has used a systematic literature review in analysing and coming up with crucial DevOps metrics to be considered for specific improvement suggestions. The review also looked at the maturity models used to measure the growth of the DevOps and examined the existing relationship between the business Objectives and Key Results (OKRs), DevOps capabilities, and their metrics. The thesis emphasizes on how these capabilities can be improved to accelerate the maturity of the DevOps pipeline and make teams rise in effectiveness. The suggestions for improvements are realized by presenting findings that outline the critical ideas collected during research and then discussion, which places this study within the existing literature and then offers recommendations for improving the state of DevOps. The discussion section includes several maturity models and identifies costs for measuring and monitoring DevOps.

Through the measuring and monitoring of DevOps metrics, the organization is able to foresee the cost of growing and developing their DevOps teams alongside their benefits. Finally, the thesis concludes with an outline and explanation of the main ideas and recommendations on how to post excellent results on DevOps practices.

Keywords: DevOps, Maturity Models, DevOps metrics, software development, deployment, state of DevOps, Capabilities, Dev, Ops.

1. Chapter 1 - Introduction

DevOps combines the tools, practices, and cultural philosophies that improve an organization's ability to deliver software solutions and services at a higher speed (Lwakatare et al., 2016). DevOps metrics are data points that directly reveal the performance of a DevOps software development pipeline and aid in quickly identifying the bottleneck in the development process and removing them (Sharma, 2017). DevOps metrics are used to track both team processes and the technical capabilities of a team enabling them to track progress, assess and measure collaborative workflows for achieving high level goals such as improved application performance, faster release cycles and increased quality (Atlassian & Tom Hall, 2016). Despite the performance turnover ratio. Organizations are able to deliver software systems to the markets at higher speeds ranging from weeks to months depending on team levels seen in software solution delivery when using DevOps (Humble et al., 2018), there is still a gap to attain optimal results and to help in DevOps maturity. The research presented in this thesis aims to evaluate the DevOps metrics concerning business objectives and software developments to suggest improvements that will help improve the state of DevOps and their capabilities in organizations. Chapter 1 introduces the overall study undertaken, followed by the research topic and background discussion, the purpose of the research is presented, the problem statement, the objectives, and the research questions are introduced. Then follows the scope, contributions, limitations of the study, and the structure of the thesis.

1.1. Background

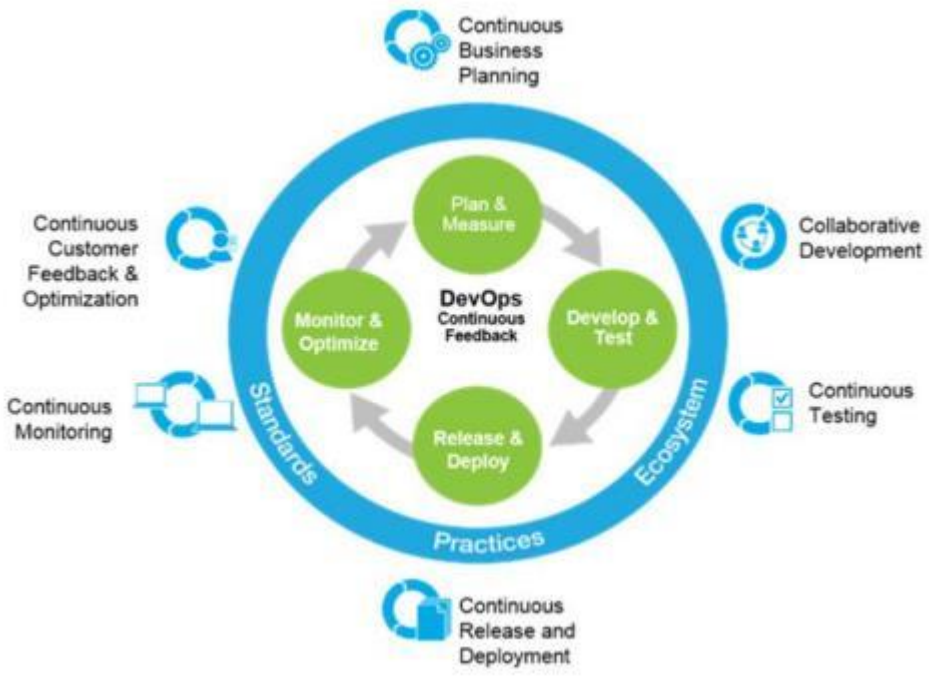
Various DevOps metrics have been recognized and approved to indicate the performance of the DevOps pipeline (Khan et al., 2020). These metrics include the mean lead time for a change, the nature of customer ticket, change volume, mean time to recovery, mean time to detection, deployment frequency, change failure rate, application performance, and defect escape rate (Elliot, 2015). The performance of a DevOps team which is measured using DevOps metric is affected by the DevOps capabilities which are based on the qualities of a team and therefore worthy looking at DevOps capabilities when discussing DevOps metrics. DevOps capabilities can be defined as the ability of the team to perform a specific task in a particular manner or the DevOps pipelines having some specified abilities in performing a particular task (Smeds et al., 2015). Various DevOps capabilities include rate of continuous delivery, amount of continuous deployment, effectiveness of continuous monitoring, effectiveness of automation, rate of continuous testing, the amount of continuous integration, and effectiveness of DevOps team collaboration (Senapathi et al., 2018). DevOps teams can be

classified in terms of their growth using the most common four speculated metrics: the mean lead time for a change, the mean time to recovery, the change failure rate, and the deployment frequency (Forsgren & Kersten, 2018). Accordingly, there are low teams, medium teams, high teams, and elite teams in ascending order of their performance based on Forsgren & Kersten's (2018) four speculated metrics. Also, various maturity models are used to quantify the growth of DevOps teams, including Feijter, Eficode, Buena, Hewlett Packard Enterprise, Mohamed, and IBM DevOps maturity models (Zarour et al., 2019).

Structure affects performance and there are various approaches to DevOps which also affect their performance and the results posted. These approaches include developer–outsourced operations mode, developer – developer operations mode, and developer–operations mode (MaCathy & Bass, 2020). A combination of these metrics, capabilities, and a chosen DevOps mode of approach, and using methods such as Agile Kanban, and Agile Scrum impact the quality and performance of each DevOps pipeline (Lwakatare et al., 2016). In addition, DevOps examines the entire Software Development Lifecycle, breaking it down into four phases and extending coverage to six areas.

DevOps scope across SDLC is shown in Figure 1, (Gokarna, 2021).

Figure 1
DevOps Scope Across SDLC

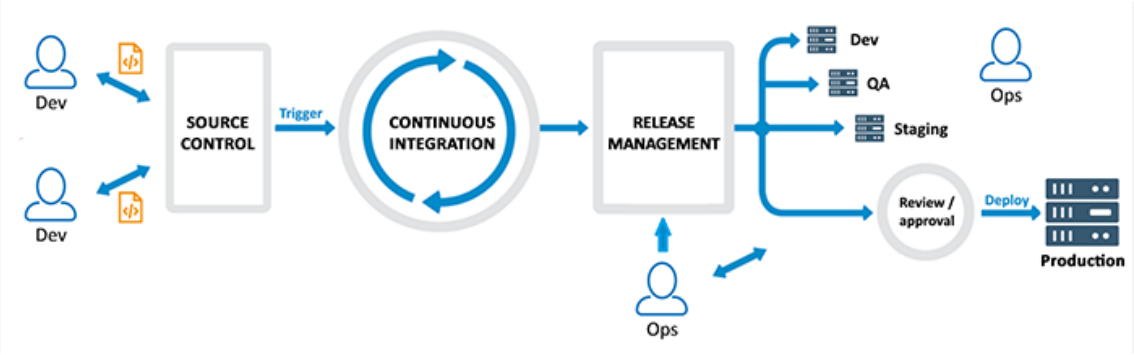


Note. This DevOps lifecycle model produced by Gakarna in 2021, summarizing of four main sections of SDLC and DevOps continuous feedback. From "DevOps phases across Software Development Lifecycle" by M. Gokarna, 2021 Retrieved from

(https://www.techrxiv.org/articles/preprint/DevOps_phases_across_Software_Development_Lifecycle/13207796/2)

DevOps evolved from various efforts to merge the activities of the developers and operations teams (Ghantous & Gill, 2017). In the early 2000s DevOps was spearheaded by teams maintaining popular websites such as Google, which invoked the use of Software Reliability Engineers (SREs) to work closely with the operations team to keep the website running in production. Flicker engineers presented a DevOps-like model in 2009, spearheading the emergence of DevOps as a popular practice. The DevOps-like model was further enhanced by the release of open-source tools such as GitHub, and frameworks such as SonarQube to further the goals of DevOps. Continued development and enhancements in DevOps resulted to emergence of DevOps metrics and capabilities which are helping the field advance each day. The Figure 2 shows the formation and the structure of the DevOps development pipeline (Ankur, 2018).

Figure 2
DevOps Working



Note. This diagram represents the DevOps pipeline by Ankur, summarizing of DevOps pipeline implementation. From "What is DevOps, It's Working, Benefits, Tools in Detail" By M. Aknur, 2018. Retrieved from https://www.dotnettricks.com/learn/devops/what-is-devops-and-devops-advantages

DevOps is becoming popular each day by enabling the teams to overcome various problems such as high amounts risk and uncertainty, incremental delivery for long term projects, and high cost of code generation; prevalent with other modes of software development approaches such as agile, waterfall and rapid application development methodology (Dviwedi et al., 2022). This is a result of using microservices instead of using the traditional means. For example, DevOps enables the teams to identify bottlenecks in the early stages of software development and, sometimes, the ones associated with misalignment of the development and the operations team. One of the key takeaways in using DevOps as the mode of delivering software solutions is aligning to the increased demand by the users for rapid and immediate fulfilment of their demands and needs with consistency and quality software products. As

teams are interested in adopting the concepts of DevOps, it is a learning process. Therefore, the teams will be required to move through a series of steps in different levels of maturity while learning and experimenting. For the team to draw a plan for their growth in the DevOps practice to improve their maturity, it is typical that the teams be able to quantify and track their capabilities over time to identify trends and diagnose issues. As in the other disciplines requiring specialized capabilities and skills, DevOps metrics can characterize the maturity of DevOps teams in various areas and levels. These metrics can also relate to the business OKRs, which are also associated with DevOps capabilities. With several metrics being proposed as outlined at the start of this subsection, there is no clear relationship between business objectives and key results OKRs (OKRs) – a goal setting framework that helps a team to set measurable goals (Al Thinyan et al., nd) and the DevOps capabilities.

Moreover, the details of the cost of measurement and monitoring of those metrics are also not well looked at. Also, the complexity of measuring DevOps metrics and various maturity models is not well discussed. Therefore, this thesis investigates the metrics and their relationship to business OKRs and DevOps capabilities while presenting the effort and cost of measuring each DevOps metric. Also, it outlines various maturity models that can be used to monitor the team's growth. The metrics are discussed in the literature review and the findings section. This will otherwise hint the practitioners on which DevOps metrics to focus on monitoring to improve specific business OKRs and achieve certain maturity goals.

1.2. Problem Statement

DevOps metrics and their relationship to business OKRs and DevOps capabilities are critically important to every DevOps team, whether a start-up or an advanced one. This is because, business OKRs which is shared across the organization is achieved through a well capable team and the capability of a DevOps metrics. Therefore, the effect of one lead to impact of the other. Numerous studies have investigated various DevOps metrics and how they are related to the DevOps pipeline, for example, DORA (Forsgren et al., 2017). Puppet on their review on state of DevOps (Puppet, 2021) and Atlassian (Hall & Atlassian, 2016). These studies have also outlined various DevOps capabilities and their significance in attaining business goals.

However, these studies have only focused on metrics and what they require, how to measure them and their role in DevOps pipeline. Also, the studies have focused on the capabilities and what they entail for example, their importance in achieving business OKRs, not digging deep into the relationships and the mutual benefits of individual components of metrics and capabilities. This only benefits a few industries which have already reached the peak

performance of DevOps and therefore need minor improvements for their teams. For example, Amazon Web Services, is at elite level of DevOps growth, and this has been influenced by its rigid management and the motivated, professional individuals who constitute their DevOps team.

As a result, the prevailing research is essentially inadequate for the industries to wholly and entirely figure out the best practices towards enhancing their DevOps practices. The inadequacy ranges from the approach to the problem, research on the problem field, and design of the solution to the problem using DevOps practices to deliver faster solutions to the market. Overall, there is a need for research and presentation of possible ways of collaborating DevOps metrics, capabilities, and Business OKRs while evaluating the cost and efforts of measuring and monitoring these aspects to realize better goals and make a more informed decision which is valuable to the company in achieving its set goals and improving the user experience.

1.3. Aim, Objectives, Goals, and Research Questions

1.3.1. Aim

Bearing in mind that there is limited research concerning the interconnections of DevOps metrics, DevOps capabilities, and the related business OKRs alongside the cost and effort of measuring and monitoring DevOps metrics, this study will aim at evaluating how DevOps metrics and capabilities of a team can be assessed to recommend the subsequent actions to be taken to continue improving the team's capabilities.

1.3.2.Objectives

RO1. To identify the common DevOps metrics that contribute to the growth of DevOps.

RO2. To evaluate the ways of measuring and monitoring the common DevOps metrics.

RO3. To identify the cost and effort of measuring DevOps metrics.

RO4. To compare various levels of DevOps teams in maturity.

RO5. To identify DevOps capabilities and their associated metrics.

RO6. To evaluate the relationship between DevOps metrics, capabilities, and business OKRs.

RO7. To evaluate how DevOps metrics, capabilities, and business OKRs can be used to recommend changes for DevOps team practices improvement.

1.3.3.Research questions

RQ1. What are the main DevOps metrics currently in use by practitioners that could be useful for characterizing the state of a DevOps team?

RQ2. How to measure each metric reliably and consistently?

- How to monitor each metric?
- When to measure it and its frequency?
- In which SDLC stage is the metric measured?
- Who measures and monitors?

RQ3. What are the current challenges and issues in measuring and monitoring each metric?

- What is the cost or effort of measuring?
- What is the cost or effort of monitoring?

RQ4. What is the relationship between the common DevOps metrics, capabilities, and business OKRs?

- How can DevOps metrics be used to improve the team's capabilities?
- How can business OKRs be used to recommend changes to improve DevOps team practices?
- What are key DevOps metrics for suggesting changes to improve DevOps metrics?

1.3.4.Goals

This thesis aims to gather DevOps techniques to assist enterprises in enhancing DevOps core values while adopting DevOps and have a rigid framework for improvement. The description of DevOps metrics and their adoption will aid other academics and enterprises to address and recognize the current status of DevOps in the market, allowing for reduced time utilization and avoiding possible hurdles. The fundamental goal of this thesis is to address DevOps metrics, measurement approaches, reliability, and associated difficulties and countermeasures and improve its actual values by integrating information from many typefaces. Also, the research will ensure that the relationship between common metrics, capabilities, and business OKRs is unveiled to enable teams to make the right judgment on where and what to improve to expand on the team's capabilities.

The first research goal relates to ambiguity in the concept of DevOps and the subsequent need to understand its underlying concepts that would be beneficial to assess its practices to identify the key performance indicators of DevOps. The first research issue is uncertainty in the idea of DevOps and the requirement to comprehend its underlying principles to analyse its techniques and identify DevOps key performance indicators. This research study is also crucial for sparking a debate over whether the idea of DevOps and its underlying assumptions are innovative or precursors to the known knowledge base of software development processes. The second research goal is to measure and monitor metrics related to software development, deployment, and delivery in DevOps, identifying the advantages and disadvantages of different measuring methods and identifying the roles and responsibilities of various personnel related to each of the common metrics in various organizations.

The third research goal is to identify the benefits and obstacles software-intensive firms are likely to face while implementing and using DevOps strategies. Research has been reported on software development project cooperation to show the extra influence of careful coordination between numerous personnel throughout the various phases of the software development process on software project success. The fourth research goal is concerned with how DevOps is used in practice. Since there is no universally accepted DevOps adoption and execution paradigm, the DevOps idea manifests differently in each firm. DevOps practitioners advocate abandoning software development techniques that favour soliciting software development and operations teams from conceptual to microservices. Organizational and architectural approach in software development where software is composed of small software components that communicates through APIs. This goal will identify the commonly used microservices and the supporting tools such as cloud computing that aid in enhancements and growth of DevOps.

1.4. Purpose

This study will contribute to the section of DevOps metrics and capabilities by evaluating and presenting possible methods of measuring and monitoring DevOps metrics to suggest improvements to the DevOps team capabilities. This will incorporate outlining the most critical frameworks for measuring and improving these metrics and their associated capabilities. This will otherwise help software development organizations deliver software solutions faster and hence create a competitive advantage. This will be of much importance to the students, researchers or any other party which might be interested in research and use of DevOps in their field of specialization.

1.5. Scope

DevOps involves continuous integration, deployment, and testing. The DevOps teams need to integrate, deploy, and test the code continuously, which might be challenging for the DevOps teams. It can help identify the teams' problems during continuous integration, deployment, and testing. DevOps metrics are concerned with accessing the performance of the DevOps. DevOps metrics measure the teams' progress in terms of commitment to providing timely, better, and faster software solutions to the customer. DevOps capabilities are tasked with displaying what DevOps can be able to accomplish. It is imperative to identify the challenges to the DevOps adoption process to identify the risks involved with DevOps adoption and the solutions to overcome them. Also, the cost or effort of measuring and monitoring DevOps metrics to increase performance is of utmost importance for better teams. The thesis also investigates how DevOps factors affect business outcomes to understand how DevOps can represent an impactful change in the software development and delivery paradigm. In addition, the study will also explore various DevOps metrics and capabilities alongside their contribution to the software development process and customer satisfaction in general.

In particular the research will cover DevOps metrics, DevOps capabilities, Business OKRs, challenges of measuring DevOps and how DevOps metrics and their factors affect business outcomes. The research will not consider the DevOps methods, and DevOps practices.

1.6. Limitations of the Research Work

Given the enormous volume of work on DevOps metrics and capabilities, an extensive search of inaccessible sources was not applicable. I did not include academic workshops, and I only cited a few books on the topic. This in return challenged me in producing the most valid and reliable DevOps data for forward oriented processes that would be required by any research project. This is considering that workshops give a hands-on experience and hence document finding in terms of what was found, and the difficulties faced in the process and also giving real time reports. It would be impractical to have all existing DevOps jobs and consolidate them into a single survey. However, it sought to cover the significant literature on the topic by focusing on journal and conference articles. It chose not to broaden the query phrase to prevent artificially preferring particular DevOps principles over others. However, the inclusion of the term "DevOps" was not required in the snowballing process, allowing us to identify critical work tackling highly relevant themes such as continuous delivery utilizing complex search engines of research platforms. Like any other SLR research methodology, there was

suspicion of bias on the information published online in various platforms. Another challenge was limited information sources since all the platform seemed to contain almost the same content approached in a slightly different manner.

Numerous research, including my own, have experienced the influence of environmental and socioeconomic factors on the propagation of the COVID-19 pandemic. This was due to the regulations passed to curve the spread of the pandemic, which had already been declared as a world threat. The requirements to work from home and also the social distance regulation put in place by the Covid-19 regulation requirements prohibited researchers like me from getting in touch with the field experts and get first-hand information on the state of the field physically. This therefore led to overreliance on already published information in carrying out research and hence limiting the information diversity.

1.7. Thesis Structure

1.7.1.Chapter 1

the context of the thesis has been introduced and discussed. The research objectives, aim, and research questions have been identified. Also, the limitations of the study have been presented. Generally, the motivation behind the thesis, the research questions with particular objectives, the contribution of research, and the thesis structure have been discussed.

1.7.2.Chapter 2

the existing study and research on DevOps to identify various DevOps metrics and capabilities. Also, multiple maturity models and DevOps teams' growth levels will be determined and discussed. Generally, the research literature review is presented in this chapter. It describes the topic in detail and adequately deals with the work that has been done previously on the research topic. This chapter will address the literature review to classify the approaches, tools, metrics of DevOps, maturity models, challenges, and mitigations in adopting DevOps in the literature.

1.7.3.Chapter 3

the research design and implementation will be discussed. The chapter will describe the research design methodology to answer the research questions and presents a detailed description of the research method, including the reasons behind the chosen research methodology. Also, the chapter will discuss how the data will be collected and analysed while justifying the collection methods and analysing techniques.

1.7.4.Chapter 4

The essential findings and their analysis are discussed. The chapter will describe the key findings based on the research questions related in chapter 1. The chapter will also incorporate the research questions, which will be in chapter 3, which presents the key findings from both grey and academic literature reviews. It contains the descriptive analysis of the DevOps metrics, objectives, and critical results. It also presents various study results such as DevOps tools and process taxonomy, maturity models in DevOps, DevOps challenges, capabilities, and achievements, which aid in improving the state of the DevOps report.

1.7.5.Chapter 5

The discussion of the main findings and the gap between the existing literature and the research recommendations will be presented. The chapter will describe the in-depth evaluation of the main findings alongside various suggestions on how the state of DevOps can be improved. It will also emphasize various metrics that need to be keenly looked at and how they can be enhanced to realize improved DevOps performance and capabilities. The DevOps metrics to be highlighted include; the lead time for a change, the deployment frequency, the meantime to recovery, and the change in failure rate.

1.7.6.Chapter 6

The overall ideas of the study will be summarized. This will include both the arguments from the literature review and the research suggestions. The chapter will describe the brief of previous chapters. It will explain the overall thesis, including the research aim and achievements, by evaluating and understanding the shared objectives and critical results for DevOps metrics and capabilities.

1.8 Justification

Very few publications and research exist based on DevOps. There exist no studies comparing and exploring the existing relationship between the DevOps metrics, team capabilities and business objectives and key results. Also, there are very few learning materials when it comes to addressing the cost of measuring and monitoring DevOps metrics. Therefore, this explains why there is necessity for further research geared towards DevOps metrics, team's capabilities and business objectives and key results. For idea to be addressed, there is also needed to fully explore the DevOps niche to ensure that it is favourable and therefore answer the question whether move to microservices is ideal in the process of software development life cycle or not.

2. Chapter 2 - Literature review

DevOps adoption and usage at various software development domains is increasing hence triggering many researchers to focus on their implementation. At the verge of implementing DevOps principles and practices (Senapathi et al., 2018), there is need to put into consideration DevOps metrics and capabilities which can be used to gauge what can be achieved and the improvements needed. There exists a relationship between DevOps metrics, capabilities, and business objectives and key results. However, many publications have mainly focused on DevOps metrics and capabilities without putting an emphasis on the existing relationship and how it can be used to foster change and improve the state of DevOps report. In the light of DevOps metrics, (Sallin et al., 2021) talks about the four main metrics which can be used to measure and monitor software delivery process. In addition, DevOps capabilities being processes which define what a DevOps can do and how they can do it (Smeds et al., 2015), should be increasingly improved in order help practitioners deliver more to their customers. Also, there are various DevOps maturity models which are used to measure the rate and level of maturity of DevOps. In contrast, there are no frameworks for accessing the cost or effort of measuring DevOps metrics, a formula for improving the DevOps capabilities, and suggestions for improving on individual metrics. Therefore, this literature review section will focus on the discussion of the common and critical DevOps metrics, Capabilities and the related business objectives and key results (OKRs). Also, the chapter focuses on the existing DevOps metrics monitoring and measuring techniques already in existence. Finally, the chapter presents various DevOps maturity evaluation models which are already in use hence setting a phase on getting an understanding on how practitioners and academic research should approach the frameworks for accessing and improving the state of DevOps reporting.

DevOps

DevOps is a set of practices that enables organisations to deliver applications and services at high velocity: evolving and improving products at a faster pace than what is traditionally considered possible. The key concepts of DevOps include automation, continuous delivery and continuous integration.

How can DevOps help organisations reduce IT costs? One key way that DevOps can help reduce IT costs is by automating the delivery of applications and services. Automation enables organisations to deliver applications and services faster, at a lower cost than traditional

methods. This also ensures that changes are made quickly and without human error, which can lead to improved reliability and customer service.

2.1. Various definitions of DevOps from literature

The term DevOps first appeared in (2008), when Patrick Debois underlined the necessity for an agile architecture and collaboration between development and operations teams at the Agile 2008 Conference. With the start of DevOps Days in 2009, DevOps became widely known (Erich et al., 2014). Over the last several years, DevOps has become a vital aspect of the software business, concentrating on developers and operations to interact effectively and produce efficient software and services (Perera et al., 2017). DevOps is a collection of methodologies for communicating and collaborating between developers and operations to provide software and services more quickly, reliably, and with more outstanding quality. From development through deployment and support (Perera et al., 2017), DevOps is the distribute of duties and responsibilities among a team entrusted with complete accountability for their service and its underlying technological stack.

According to (Riungu-Kalliosaari et al., 2016) DevOps is a collection of methods to shorten the time between making a system change and putting it into production while maintaining high quality. Organization mainly fails to deliver in software projects due to product development and delivery difficulties. The failures can either be as a result of unmanaged cost, time or the resources used during new product development or existing product upgrade. This inefficiency makes many business organizations to lose open opportunities and hence losing changes to expand their market share and continue building their competitive advantage. Therefore, this definition perceives DevOps as a framework which can aid in reducing software project failures through shorting the time for delivery and also maintaining quality of the delivered product.

According to Makosi (2017) DevOps is the combination of cultural philosophies, practices and tools that increase an organization's ability to deliver applications and services to its customers at a high velocity. According to this definition which is also the take for amazon web services (AWS), DevOps is composed of cultural philosophies, tools and practices. In light of this definition, cultural philosophy involves change in culture and mindset. This means DevOps advocate merging of the development and operations team and sorting out the differences in order to deliver the best products and services to its customers. DevOps practices focus on the key practices such as automation and streamlining of software development and infrastructure management processes which help organizations to innovate faster. These practices and processes can be achieved by having the right tooling for carrying out those activities and

these include automation tools, orchestration tools among others. One of the key takeaways of DevOps according to this definition is organizational automation. Organizational automation is one of the essential qualities and dependability indicators. Although there is a considerable body of research that examines this critical feature, little has been done in the area of DevOps. This has prompted a search for a link between software quality and DevOps in organizational automation.

2.2. Relationship Between DevOps Metrics and Business OKRs

Objectives and Key Results is part of project management framework which is useful in ensuring that organizations develop products which are aligned to their goals and mode of working. This framework gives the reason why every project has to align to the organizational goals and objectives (Breyter, 2022). Therefore, these objectives are as well used in measuring the project success and hence vital in the process of ensuring that state of DevOps is improved, which is associated with making the delivery of product to the market.

According to most researchers, DevOps refers to the software development team's ownership and responsibility for designing, implementing, testing, deploying, and maintaining online applications and services in production. The primary purpose is to quickly make software updates visible in circulation and provide immediate feedback on those changes. Operations workers assist development teams with tasks such as automating deployment pipelines, sharing knowledge about security, scalability, and performance concerns, and working together to resolve crises. Most interactions between software development and operations workers were informal (Lwakatare et al., 2019). In a study, DevOps is understood in different cases as follows:

- DevOps means the team has complete control over the infrastructure and the deployment and, every possible thing that can be automated is automated.
- In DevOps, the persons developing the applications are the same ones to maintain them in production. If something goes wrong, then the same persons investigate the problems and fix it accordingly.
- DevOps investigate the problems and fix software you are developing in production.
- It means giving the team more responsibility and control that it does not need hand over to operation team the deployment. It is gaining confidence in the deployment procedure and the ways of working.

Objectives and Key Results (OKR) is a goal-setting framework. It allows departments or organizations to note the priority items or activities and offer focus and alignment for teams and companies (Shah & Dubaria, 2019). Therefore, with OKRs, the focus is on the objectives and the final product. The priorities of a business are identifiable from the OKRs. The use of OKRs is necessitated by focusing on the end product. Notably, the main objective of DevOps is to shorten development cycles, increase deployment frequency, and add quality with dependable releases, all in close alignment with business objectives (Trihinas et al., 2018).

In order to gauge the performance of DevOps through DevOps metrics, there must be some OKRs that capture the priorities of the company or organization. The DevOps OKRs go together with the DevOps capabilities. Key DevOps capabilities include version control for all production artifacts, automating the deployment process, implementing continuous integration, and using trunk-based development methods (Lopez-Fern ´ andez et al., 2021). To successfully evaluate the capabilities of a team to recommend the subsequent actions, DevOps OKRs must be defined within the parameters of the set by the organization. These are some of the common OKRs for DevOps and how their performance is affected by DevOps metrics.

2.2.1. Product Development

Product development in DevOps starts from idea conceptualization to when the product is deployed to the production to be used by the end user. Depending on the software or, say, business idea, the objective, and the key results to be targeted can be generated from the product development (Hemon et al., 2020). The objective should be well stated in product development and the critical results expected to be outlined to the team (Linke, 2019). The capability of the DevOps team should fall within the set objective and critical results expected to ensure product development. DevOps success is most closely related to modular architectures, which favour operational requirements, formal verification, maintainability, and customizability. We hope that the architectural decisions taken by the teams to support DevOps will be helpful to other enterprises (Mishra & Otaiwi, 2020). Therefore, in this regard, DevOps help in shortening the life cycle of the software development while delivering updates, features and fixes in alignment to business goals continuously and hence help building a positive market reputation for the organization involved. DevOps metrics such as defect escape rate and customer ticket are used in determining the quality of the developed product. Also, change failure rate, lead time for change and deployment frequency quantify the ability of the team to provide timely products to their customers (Breyter, 2022). Exemplary performance of these metrics ensures that the organization builds and maintains a constant

market advantage for delivering. Therefore, organizations are continuously adopting and upgrading their processes to accommodate DevOps and enjoy its benefit in the market share.

2.2.2.Continuous Testing

In the process of Development of new software product, there are various test such as unit tests, integration tests, defects tests and validation tests which are carried out to ensure that the final product is up to standard. Therefore, automating testing using testing tools and making it continuous during development is a huge turnover provided by DevOps. The business world has changed and now we are an era where, each organization needs a software in order to run their activities well and sustain their business in the market. This makes the business order to a software from the organization which can deliver it at the shortest time possible. In order to work best under pressure, continuous testing is one of things that will as assure the DevOps team that they are following the right track. Also, continuous assessment helps the team avoid what could be termed as expensive mistakes (Lwakatare et al., 2019). The objective of continuous testing may be to improve the testing process on an annual basis, and the key expected results may include increased functionality (Shah & Dubaria, 2019). The OKRs may vary for continuous testing, but the objective is to free the process from bugs and glitches. There are various metrics which are in line with continuous testing, and they include an increase in defect escape rate, a more positive customer ticket, rapid deployments, and a low change failure rate. This is as a result of a well-established testing framework which ensures that, what is required is revisited each day and done correctly before it is too late. Therefore, as one of the businesses OKRs into consideration, continuous testing helps in delivering bug free software products, delivery of quality software products and timely deliveries.

2.2.3.DevOps Monitoring

Monitoring DevOps covers every section of the development process. therefore, all the processes, including planning, development, integration & testing, deployment, and operations, are monitored in DevOps (Callanan & Spillane, 2016). Monitoring ensures that all vulnerabilities are identified at the earliest time possible. The objective of monitoring is mainly to observe the server's health (Bezemer et al., 2019). Key monitoring results include reducing the probability of errors and optimizing page load. Generally, a DevOps team needs monitoring to ensure necessary changes are implemented during the development phases. A well-developed monitoring framework leads to some improvements in the meantime to detection of an error in case a system running in production fails and also the mean time to recovery for

such system is minimal to a point that the customers does not notice. Therefore, this leads to improved customer ticket and hence helping the organization to guard its reputation at the market.

2.2.4.Release Management

Software development goes through a systematic cycle. The release management is a process of managing, planning, scheduling, and controlling software development through different stages and environments. Also, release management involves testing and deploying software releases (Artač et al., 2016). The phases of release management are critical in guaranteeing the correct release of components. Planning release, building release, user acceptance testing, preparing the release, and deploying release are the five main steps of successful release management (Callanan & Spillane, 2016). The objectives and key results of release management may vary depending on the expectations of the DevOps team. Increasing the frequency of release and updates may be an objective for the DevOps team, and the key results may be to reduce the number of patches from the current figure to a lower figure. Release management is an essential aspect of OKRs.

2.2.5.Change Management

Change is a permanent phenomenon in every aspect of life, technology included. Change management should be incorporated in DevOps teams skilfully and professionally. Therefore, change management is a systematic approach to handling transitions, transformations, or modifications of a team's or organization's goal, procedures, or technologies (Luz et al., 2018). Notably, change is not an event but a sequence of steps that move change from begin to actualization. Change Management Plans are usually developed to support a project to deliver a change (Kersten M. , 2018). Involving the team through communication is one of the initial steps to effect change in a DevOps team. The objectives and key results of change management include strengthening the change management process and reducing the number of failed changes. Change management is crucial for a DevOps team in ensuring continual innovation and better applications. The related metrics for change management include lead time for change which should be as minimum as possible, Deployment frequency which should range at multiple times per day and change failure rate which should be as minimal as no changes failure rate within a single project and hence additional in the quality of the software products and services delivered to the end user.

2.2.6.Configuration Management

Processes and activities in IT are typically configured. Configuration management is thus a core aspect of IT. Simply, it is a system engineering process for establishing consistency of products attributes throughout its life (Senapathi et al., 2018). The role of the DevOps team is critical to ensuring configuration management effectiveness. The additional functions may involve tracking and monitoring the hardware and software owned by the organization. As known, configuration management is a core OKR in IT, more so in software development (Yin & Filkov, 2020). The objective and the key results have to align with the organization's expectations and goals. Configuration management dals with maintaining consistency in software product and therefore concerned with the software upgrade and customer service support. A well-formed configuration management ensures that the Mean time to recovery and mean time to detection are minimal to an extent that the customers does not notice that there was a downtime in the system functioning. This can be achieved by employing the right tools and resources within the DevOps pipeline.

2.2.7.Deployment

Deployment is an essential element for DevOps professionals. It is the final stage of the project. The objectives and key results set by the organization are measured at the deployment stage. Therefore, the OKRs set should be in a position to improve deployment. The objective should improve the deployment process, recovery, and failure rate (Batra & Jatain, 2020). Consequently, the key results may reduce the frequency of failed deployments. Though deployment depends on the type of project, the utter goal is to make the deployment process better.

2.3. DevOps Metrics

Generally, DevOps metrics refers to measure of the performance of the DevOps pipeline and team functioning. DevOps metrics give data for the performance of the DevOps team periodically and hence making it easy to monitor the trends in the performance of the DevOps team (Roche, 2019). This allows an organization to get the insight of the performance of the team and be informed whether the team performance is improving or degrading over time. This would otherwise be used in suggesting changes to the DevOps team on the changes which need to be introduced in order to keep the team activities growing (Hussaini, 2014). This is therefore essential for improving the state of the DevOps report.

DevOps pipeline performance is one of the key aspects which is used in determining the business value of a DevOps team. Therefore, in order to track the performance of a DevOps

software development pipeline, we use a set of datapoints called DevOps metrics. The DevOps metrics can be used to track both technical capabilities and the processes of a DevOps team. Tracking of the technical capabilities is necessitated by assessing the efficiency of the DevOps in delivering the product to the Hence market. For example, lead time for change and change failure rate enable one to assess the skillset of the team and how it can respond to the demands in the market. Hence, to effectively measure the performance of DevOps, there is a specific combination of DevOps metrics which is used. These metrics are monitored and each of them dictate which level of DevOps belongs to. Therefore, metrics in DevOps are data points that shows the performance of a DevOps software development pipeline through identifying and removing glitches in the entire process (Perera et al., 2016). The capabilities of the DevOps team are easily traceable through the use of DevOps metrics. The main objective of DevOps metrics is to reduce the gap between development and operations. This is especially done through ensuring that the team that does development is also concerned with operations of the software product and therefore making it easy for bug fixing and quality improvement to be faster (Katal et al., 2019). Essentially, DevOps allows betters collaboration between developers and system administrators (Shah & Dubaria, 2019). Metrics are necessary elements to the DevOps team because they are used to measure and assess the collaborative workflows and track the progress of attaining the top-level goals which aide in maintaining competitive advantage for the organization, faster releases and customer satisfaction mainly associated with product development. The high-level plans in software development include improved quality and application performance and better frequency in release cycles. The number of DevOps metrics utilized by the team is plenty enough to provide an insight on the team performance, and their application depends on the organizational goals. Therefore, from the list of the DevOps metrics available, the literature focused on the ten common data points that a DevOps team can observe. These 10 metrics were selected based on how they are measure and what they really quantify. Performance is critical to each company and one of the performance measures is the output, therefore, these metrics are output oriented and hence aid in gauging and suggesting improvements to a team. They are as discussed below.

2.3.1.Lead Time for Change

Lead time for change can also be referred to as lead time. It is described as "the time required from committing code to having it effectively perform in production." Faster feedback, course correction, and faster delivery of a problem cure are all advantages of a shorter lead time for change. It is a vital data point for any DevOps team. Lead time is the total duration of time taken to complete a specific task (Smeds et al., 2015). In essence, it is the duration between

the onset of a task and its deployment time. The data point specifies the level of performance of the DevOps team. For example, the DevOps team in the bracket of high-performance measures lead time in terms of hours (Lopez-Fern ´ andez et al., 2021). However, this may not be the case for every piece of the task assigned to a DevOps team.

In some cases, tasks may be overwhelming and demand more time. The best way to improve the lead time is to work in small batches (Maroukian & Gulliver, 2020). Hence, the practices, including the lead time, ensure that developers get quick feedback and respond to the input by making necessary changes or maintaining development.

According to a survey by Sallin in (2021), 9 out of every 16 articles define change lead time. The deployment frequency does not offer much room for interpretation. All recommendations based on the original FKM definition quantify the time it takes for a commit to reach production. Using a version control system for source code is now standard practice. A developer must change source code and place it under version control to make changes to the software system. As a result, the commit is referred to as the "change." As a result, the lead time is determined by the time interval between the commit and deployment timestamps (Sallin et al., 2021).

2.3.2.Change Failure Rate

The effectiveness of software can quickly determine the success of DevOps teams to effect change with little or no probability of failure. It is defined as the proportion of changes to an application or service that cause service degradation or remediation (e.g., cause service impairment or outage, necessitate a hot repair, rollback, fix forward, or patch). The team classified under the emblem of high performer has to change failure rates in 0-15 percent (Callanan & Spillane, 2016). Moreover, the metric ensures that defects in the process are easily identifiable and fixed at the earliest stage possible. A DevOps team with the lowest probability of change failure rate is characterized by test automation, trunk-based development, and work in small batches (Wiedemann et al., 2019). Therefore, it is essential to track the rate of failure change to be able to identify and fix any bugs.

According to a study, by Sallin, Kropp, Anslow, Quilty, & Meier (2021) 9 out of 16 articles contain a definition for change failure rate (Sallin et al., 2021).

The following list is a different suggestion:

- The amount of releases (percentage) that were pursued by a "fix release".
- Count of hotfixes in commit messages.

- Failures can be identified through the use of monitoring data that are split by deployments.
- Carefully, indicate if a deployment was productive or otherwise.
- Count the number of reversals divided by the number of deployments.

To calculate the failure rate, first define what a change is. A deployment signifies a change in all the recognized articles. Therefore, the change failure rate is calculated as the proportion of change failures to the number of changes that are deployed. The following step is to identify a failure and establish a relationship between it and a change. However, unlike the timeframe to re-establish service, management services cannot be used to discover failures; a change failure encompasses all occurrences where immediate repair is necessary. The team will be accountable for the deployment, and any following errors will be rectified quickly without the need for an incident, especially for development teams with an excellent software delivery performance.

2.3.3.Deployment Frequency

Like any other metric, deployment frequency is a vital data point in determining the performance of a DevOps team. It is concerned with lowering the batch size of a project “reducing it is a central element of the lean paradigm”. They used the frequency of software deployment to production as a proxy because this is difficult to evaluate in software. The good deployment frequency can deploy changes on-demand and often with prevalence (Haindl & Plosch, 2019). The strategy of deployment by the DevOps team is crucial in determining their performance. Erich and co-authors acknowledged the effectiveness of automated systems when dealing with deployments. They denoted that automated testing and feedback mechanisms minimize the need for human intervention (Erich et al., 2017). Hence, deployment cuts across production and non-production. Having data points for production deployments and non-production deployments is an effective strategy for improving the performance of a DevOps team (Trihinas et al., 2018). Therefore, quick deployment reflects higher performance from the DevOps team. Consequently, delayed or deployment at lower frequency points to the low-performance capabilities of a DevOps team.

According to a study, 7 out of 16 articles offer deployment frequency definition. The definitions of deployment frequency do not differ significantly because it is already clearly defined as the deployment of software to production. The "number of deployments/releases in a given time" is tallied in all of them. Some say they track successful deployments (albeit not defined), while others say they count deployments to production. Deployment is defined as a fresh release for automated measurement because this is a speed measure (Sallin et al., 2021).

2.3.4. Mean Time to Recovery (MTTR)

Usually, systems are accustomed to either systematic or operational failures. The ability of a team to recover from such failure and resume smooth operations is identified as the mean recovery time. It is described as a failure in fast-changing complex systems is unavoidable; the crucial question for stability is how long it takes to restore service following an event (e.g., unexpected outage, service degradation) from the moment the occurrence happens. DevOps teams exhibiting characteristics of high performance recover from system failures with ease (McCarthy et al., 2015). The main objective of the metric is reducing the frequency of failure, and if in case they occur, they can be fixed easily. Undoubtedly, continuous monitoring or, say, having good application monitoring tools is necessary to identify system failures at the earliest stage, thereby affecting solutions quickly. The metric is measured in hours and reflects the performance of a DevOps team. For a quick mean time to recovery to be efficient, the operational team should be notified with ease in case of system failure. In addition, the team should be equipped with the methods, tools, and privileges necessary to remedy the system problems (Kersten M. , 2018). MTTR is an effective metric when determining a DevOps team's speed, quality, and performance.

According to a survey, 8 out of 16 items define time to restore service. Five of them describe time to restore service as the average time it takes to close an incident within a given time frame. One proposal is to use chaotic engineering (i.e., design a failure and track how long it takes for it to be detected and addressed), and another is to poll the "status" regularly and record how long it takes for the degradation to be restored when the status shows degradation. Two articles imply that the time to restore service should be assessed for unsuccessful releases, so advises designating "fix releases" and monitoring the time between one release and the subsequent "fix release". The causes of failure are many, and many rely on human interpretations of what constitutes "failure" and "repair", making it impossible to automate this statistic fully. The calculation via incidents is an intriguing option if a team has incident management. This technique allows for a combination of manual and automatic failure identification because incident generation may be automated (Sallin et al., 2021). In majority of publications, time to restore is defined as the period from the start of an event to the end of the occurrence.

2.3.5. Customer Ticket

Customer satisfaction is perhaps the most crucial objective of any department within an organization. Customer ticket metric is a guided data point that gives the system developers

feedback on how smooth or complex the system operates. Occasionally, some bugs and glitches may bypass the DevOps team. The frequency at which the team receives customer tickets embedded with notification of errors or bugs explain the team's performance (John et al., 2017). The volume of tickets indicating robustness or quality issues in the system points out the performance of the DevOps team. The team should be in a position to work on the complaints aired by the customer tickets to ensure that the tickets received lie the system's robustness. Customer ticket metric deals with the end-user, and for that reason, it presents the best and the worst opportunity to gauge the performance of a DevOps team (AppDynamics, 2021).

2.3.6. Defect Escape Rate

The number of defects detected in software between production and deployment represents the defect rate. The ability to find a timely solution to the defect falls under the metric of defect escape rate. Some of the initiatives that can be put in place to identify these defects during the development phase within the pipeline are highlighted by Šćekić. He outlines the importance of continuous testing and monitoring of the development process to detect any defects (Šćekić et al., 2018). The numbers of defects that make it to production determine the level of performance of the DevOps team. Therefore, a team that detects defects at the early stages of development and fixes them presents high performance. It is worth noting that at the time, these defects may be bypassed during testing and monitoring phases. Nonetheless, it is essential to identify the defects before reaching the quality assurance and the production departments.

2.3.7. Automated Test Percentage

Velocity and high performance are among the main objective of DevOps metrics. To achieve these objectives, the DevOps team should adapt and employ technology to monitor and test the whole production pipeline (Maroukian & Gulliver, 2020). The metric, in this case, focuses on the effectiveness of the automated test. Also, tracking how long the automated tests can last owing to the frequency at which codes are changed is an essential metric to any DevOps team (Gunja, 2021). Speed should be a priority to the whole team when identifying bugs and errors within the system. Notably, teams with high performance monitor the percentages of the automated test (Gunja, 2021).

2.3.8. Deployment Size

The total number of deployments is another important metric that the DevOps team can follow. Though, there are other metrics, like deployment frequency and failed deployments,

deployment size gives the team the volume of work it can manage within a specific time frame (Torble, 2019). Also, the metric can be used to identify the types and numbers of deployments that are actualized by the team each day. The effectiveness of the DevOps team can be determined by analysing deployment size metrics (Singh, 2021). Nonetheless, deployment size depends on variables like frequency, complexity, and number of deployments.

2.3.9.Error Rates

Software and applications are prone to errors. Therefore, tracking error rates is vital throughout the development process. Errors during development to deployment need to be minimal to reflect the team's high performance (Artač et al., 2016). Also, resolving any arising errors within the pipeline is an effective strategy for dealing with bugs and production issues. The use of automated testing and monitoring tools is critical in the early identification of errors. Notably, every system will have a few errors due to the noisy system (Lwakatare et al., 2019). Nonetheless, the DevOps team should be able to identify and solve most of the errors.

2.3.10. Mean Time to Detection

Meantime to detection metrics is an essential data point that denotes the ease of identifying such failures within the system (Bezemer et al., 2019). Having system failure or outage at the latter stages of development should not be the last thing to be expected when working on any project. The metrics, therefore, determine the team's effectiveness in identifying errors and failures as soon as they occur. Effective DevOps team has a minimum mean time to detection. Like in other metrics, robust monitoring techniques and automated testing will lower the mean time of detection (Gunja, 2021).

2.3.11. Application Performance

Before deploying the system or application, the performance should be tested for any problems, errors, bugs, or other issues. The application's performance should be beyond good and should operate at the top level. High levels of performance before and after deployment reflect the effectiveness of a DevOps team (Bezemer et al., 2019). Continual monitoring and testing are essential in enhancing the performance of the application. Testing the performance of applications before deployment is essential at identifying errors and burgs that the end-user would have discovered.

2.4. Maturity Models in DevOps

Organizations are increasingly interested in measuring and improving their software processes using well-established maturity models such as CMM, CMMI, and ISO/IEC 15504. It has been

noticed that maturity assessments are expensive and time-consuming for enterprises, requiring more effort to automate this process (Simões Teixeira, 2019).

2.4.1.IBM DevOps Maturity Model

A detailed examination of the IBM DevOps strategy can aid in promoting continuous software delivery. Four aspects in adopting or implementing continuous software growth inside a company are highlighted in literature. Planning and measuring, designing, and evaluating, delivering, and deploying and analysing and optimising are some of the factors to consider. The IBM DevOps assessment system is a training approach that reflects a larger context inside an organization's adoption framework. It focuses on iteratively defining best practices to be used in the adoption of new software solutions (Bahrs, 2013).

IBM DevOps Maturity Model is a well-articulated method for evaluating an organization's current DevOps practices. It also aids in the creation of a clear DevOps implementation strategy. Furthermore, the research effort mentioned earlier provided readers with a high-quality method for assessing an organization's progress in using the IBM DevOps strategy. Most importantly, this DevOps maturity lays out a step-by-step process for planning, piloting, and deploying system enhancements within a company (Zarour et al., 2019).

It is worth noting that this model does not include the IBM DevOps approach's applicability to other software products that do not run IBM software. Another flaw is that it lacks a clear rationale for the capital approach for obtaining DevOps maturity (Zarour et al., 2019).

2.4.2.Mohamed DevOps Maturity Model

Mohamed presented a new DevOps maturity model and then evaluated how the model would affect emerging multilateral software development techniques and procedures. The suggested DevOps maturity system is predicated on the Capability Maturity Model Integration (CMMI) and includes five maturity levels across four components: quality, automation, interaction, and accountability. According to Mohamed, implementing the CMMI-based DevOps maturity model improves operational efficiency, increases visibility, and mitigates essential risks such as downtime during adoption. The benefit of this method is that it uses CMMI model approaches to determine the capabilities of the DevOps model at each level of maturity. It also presents a comprehensive transition structure as the DevOps model progresses from one stage to the next (Mohamed, 2015).

2.4.3. Hewlett Packard Enterprise DevOps Maturity Model

Hewlett Packard Enterprise (HPE) developed a new maturity model that is aligned with the CMMI maturity model in order to assess DevOps implementation. This strategy is intended for large businesses to span the complete lifecycle of an application. It is used to assess the process, automation, and cooperation. It is done on five maturity levels. The initial level has poor, Adhoc communication and co-ordination, no automation, and unpredictable, uncontrolled reactive processes. The managed level has managed communication, shared decision making, solid communication, no central infrastructure, processes are managed but not standardized. The defined level has collaboration shared decision making and accountability, central automated and standardized process across organization. The measured level helps to aid in identification and bottlenecks in process, analyse the metrics of automated process and measures are done against business goals, process is visible and predictable for quality and performance. The optimized level includes effective knowledge sharing and individual empowerment, self-automation, learning and remediation and most importantly includes risk and cost optimization (Inbar et al., 2013).

2.4.4. Bucena DevOps Maturity Model

Bucena DevOps maturity framework is based on the CMM framework, with five maturity levels. Each of these tiers comprises four facets: technology, process, people, and culture. In the initial level, environments are provisioned manually involving manual tests and minimal automation and data migration is unversioned and performed manually. In the repeatable level, environments are externalized and versioned involving functional test automation and changes to database are done with automated scripts versioned with application. In the defined level, virtualization is used involving triggered automated tests and data base changes are performed automatically as part of deployment process. In managed level, all environments are managed effectively with smoked tests and shared dashboards and database upgrades and rollbacks are tested with every deployment. At an optimized level, provisioned environments are fully automated involving chaos monkey, and feedbacks are taken from database performance after each release (Bucena & Kirikova, 2017).

2.4.5. Eficode Maturity Model

The structure and mindset, contexts and distribution, building iterative development, quality assurance, and visibility and reporting are the five dimensions of Eficode's maturity model. Four levels of maturity are defined under the paradigm. DevOps practices are not employed at the primary level. Organizations in the second and third levels have begun to implement some

DevOps concepts, although they are in their early phases and have the potential for improvement. The final level of Eficode's architecture focuses on using measurements for continuous improvement to attain efficiency and quality, at which point DevOps approaches become appropriate (Eficode Oy, 2015).

2.4.6.Feijter Maturity Model

The Feijter DevOps maturity model offers strategic priorities that allow implementation and execution organizations to evolve finely. The concept is considered for software product organizations that create software for multiple clients, but not bespoke products for a single customer. The Feijter model has 63 capabilities distributed throughout ten capability levels. Feijter model is a non-traditional model because of its focus area architecture. In this context, staged and continuous representations are traditional. Focus area architecture is different for two reasons. First of all, there can be unlimited quantity of maturity and capability levels. Secondly, each focus area has different evaluation intervals. This model describes sixteen focus areas which must be taken into consideration while trying to adopt or improve DevOps practices in company. All focus areas are logically grouped into three groups (De Feijter et al., 2018):

1) culture and communication

2) product, process, and quality

3) foundation

2.5. Challenges Hindering DevOps Growth and Their Mitigations

2.5.1.DevOps Values in Software Industry

These five potentially disruptive technologies and trends were identified by the 182 IT professionals who took part in the Computerworld Forecast 2016 survey: the rise of DevOps, virtualization 2.0, carbon-reduction technologies, the evolution of IT-marketing alignment, and sharpening of IT focuses on customer experience (Stackpole, 2015). Organizations practicing DevOps consistently report more frequent deployments, shorter lead time to change, decrease in failure rates, and faster mean time to recovery (Kersten M. , 2018).

DevOps can assist in avoiding or mitigating the issues confronting our software industry today. These issues are the following (Kim et al., 2015; Wahaballa et al., 2015):

- Consternation substitute: Once an application has been delivered, the stakeholder is frequently terrified by the prospect of change. As a result of the predicament, bureaucratic

management systems are in place; making any changes to the application takes an inordinate amount of time to implement.

- Deployment risks: The term "software deployment" refers to all the processes that take place to make a software system accessible for use. Because no one is entirely convinced that the software will function effectively in its real-world setting, developers and operations teams are becoming increasingly concerned during this phase. It simply places it and wait to see if it topples over.

- Blame game: Typically, system administrators or end-users are the ones that notice the problem. It is determined that there is an issue, and it is reported to the developers. This situation may develop into a classic scenario of finger-pointing upon each other.

- Isolation: In conventional project management settings, the project team is organized into two groups: the project managers and team members. The first is the technical team, which comprises programmers, testers, and quality assurance, and the second is the operational team, which comprises a database and network administrators and operations personnel. Because they are independent of one another, this division frequently behaves as a barrier in their operations.

- In terms of time and human/financial resources, the actual management of external networks results in a high operating cost in terms of both time and resources.

- Lack of visibility into network and service statuses makes it challenging to ensure a high quality of experience for customers.

- The failure to identify and easily locate the cause and precise location of issues (troubleshooting) and to debug software problems.

2.5.2.Challenges of Improving DevOps Capabilities and Adopting DevOps

Even though DevOps effectively addresses most of these issues, several challenges have been identified in the literature by various authors. Organizations have ignored or misunderstood the cultural, organizational, and procedural adjustments needed to embrace a new way of working with technology. They have invested in automation but not in the organizational processes and mismatched incentives that led to the DevOps movement (Kersten M. , 2018). This is made even more necessary because all technical capabilities are being designed and deployed on a centralized channel with no formal requirement processes and no analysis of the overall enterprise competencies and infrastructure in place. Though on the surface, the

method appears to lack the customary rigor that most engineers are accustomed to, the process can bring capabilities to operations far more quickly than is now the case (Farroha & Farroha, 2014). DevOps addresses collective interests and benefits and shared processes and tools as part of a collaborative effort. Because of the inherent tensions that exist between diverse groups, it is possible that common objectives and incentives are not always feasible. In any case, they should at the very least be in sync with one another (Wettinger et al., 2015).

DevOps framework was developed in response to growing evidence that there is a 4C (communications, co-operation, culture, and collaboration) gap between what is traditionally considered the IT development function and what is generally recognized as an organization's IT operations function. This disparity frequently increases and presents itself in various ways, including unfair competition, team conflict, excessive delays, and inefficiency of business applications and information technology systems (Hussaini, 2014). Even while the notion of DevOps looks to be widely accepted theoretically, it is proving challenging to implement completely.

To put it another way, it is challenging because of the requirement of preserving a runtime environment, a lack of commitment from top management (particularly in the software development community), a rigid organizational structure, and internal resistance from the workforce (Jones et al., 2016). DevOps implementation issues have slowed it down by limiting DevOps enablers or raising the risk of not reaching DevOps goals. This problem involves finding new employees with the necessary technical abilities, developing skills, and keeping current employees. Because the required competencies are lacking at the moment of demand, a lack of suitably qualified people might cause the DevOps adoption path. To overcome reluctance to this long-term shift, effort and the uncertainty of how this change would affect them in the future, the movement to the DevOps method of working requires some incentive.

Transitioning the commodity to the cloud and implementing a micro-services architecture was a vital facilitator of the company's adoption of DevOps and continuous deployment practices. According to the literature, it was a difficult and challenging phase of the DevOps journey for the product re-architecting team. The progressive change in duties associated with DevOps adoption has occasionally resulted in misconceptions about who is accountable for what tasks inside an organization (Senapathi et al., 2018).

Most application problems, such as crashes, poor performance, or improper application behaviour, may be traced down to a small number of root causes. For example, non-optimized database access may result in memory leaks, deployment problems, or wasteful code.

Companies who believe that Continuous Delivery and DevOps would cure all these difficulties are doomed to failure since they just run into these issues more quickly than their competitors (Gottesheim, 2015). Development and information technology companies are under growing pressure to build and deploy applications at ever-increasing speeds to assist their enterprises in driving top-line growth.

Defining DevOps as a theoretical foundation for reintegrating the development and operations of information systems can remove the traditional boundary between developers and operations experts. DevOps is a set of ideas and practices that help enhance work quality by fostering tight cooperation between development and operations employees in a project. On the other hand, both sides have paid little attention to the problems that each other is experiencing. Communication breakdowns are a recurring issue in agile teams, and they are particularly noticeable in the interaction between developers and operations (Diel et al., 2016).

Three different but feasible methods to bringing Dev and Ops closer together are presented, each focusing on specific activities (Nybom et al., 2016):

- Merge Responsibilities: All engineers should be responsible for both development and operations.
- Merge Personnel: Increase communication and cooperation between Dev and Ops while keeping current responsibilities distinct.
- Bridge Team: Form a distinct DevOps team to act as a link between Dev and Ops. The challenge is to identify & decide on which approach to use.

In many organizations today, DevOps (development and operations) is a new paradigm that aims to remove the gap and barrier between developers and operations employees. As previously said, the primary promise of DevOps is to allow for continuous delivery of software to allow for rapid and frequent release cycles. Customer needs change often and responding quickly to these changes may be a critical competitive advantage in some instances (Wettinger et al., 2014).

A frequent issue with DevOps is that it is more cultural than technological. Most of the time in today's world, organizations and their processes must be broken down into components that adhere to only certain critical design principles, such as the minimization of functions with the least number of dependencies, portability, shared knowledge, predictable contracts, and maximization of human value, among others. The last three principles provide a summary of the definition of DevOps. The idea of improved integration between Development and

Operations is a worthwhile goal to strive forward. The aim is to promote quantifiable incremental cultural change to extract the total value from the confederation of people, processes, and technology. However, cultural problems, incentive structures, and risk distribution are significant roadblocks to achieving those objectives (McCarthy et al., 2015).

Although both practitioner and academic research circles have recognized DevOps as a critical component of the software delivery architecture, insufficient documentation has been provided to define and clarify what it is. Without this knowledge, the phenomena cannot be adequately conveyed, and its influence on communities is unknown (Lwakatare et al., 2016). Software development companies must provide software in a timely, dependable, and predictable manner. Despite this, they often struggle to put appropriate methods and processes in places, such as release engineering and DevOps. For one thing, the absence of consistently defined words in both cases makes it harder to comprehend the meaning and adds to the confusion. Another factor is the lack of consistent meanings for each phrase. As a result, these words are often misunderstood, misconstrued, or used interchangeably as synonyms (Dyck et al., 2015).

DevOps is a new revolution that seeks to connect developers with those in charge of operations closely. This is needed to allow rapid and frequent releases to provide software continually. When issues or feature requests arise, users and customers of today's web applications and mobile applications hosted in the cloud expect to get prompt responses. As a result, reacting rapidly is a critical competitive advantage. In addition to the cultural and organizational changes that are needed to put DevOps into effect, technology is required to automate the whole deployment process from start to finish. Collaboration and integration between development and operations are made possible via automated processes. The DevOps community is constantly pushing for new methods, tools, and open-source artifacts that may be used to automate these types of operations in the enterprise. However, since all these patented and diversified DevOps automation methods are so dissimilar from one another, it is difficult to integrate and mix them in term of deploy applications on the Cloud (Wettinger et al., 2014). When coupled with a micro-service-style architecture, emerging container platforms such as Docker provide unparalleled agility in building and operating applications in a cloud environment. However, it is often difficult to utilize containers to administer cloud infrastructure without preceding many of the advantages that containers provide in the process (Kang et al., 2016). It is simple to utilize the most recent tools and libraries while developing new web application projects. However, as the project gets older, its

code and the libraries on which it relies become legacy code, making it more challenging to maintain the infrastructure. The complexity may be significant at times; nevertheless, even with decreased complexity, creating isolated, single-tenant virtual grids might be challenging for researchers who do not have prior expertise with cloud-based system administration. Another significant distinction between scientific research and web development is the problem of testing. Continuous integration based on unit tests in agile web development is an essential building block. Tools like Jenkins streamline the practice of bringing the most recent version of the program, compiling it, running all regression tests, and even uploading authorized releases to the production servers on a schedule that the user sets. The outcome of an independent regression analysis must be known in advance to create the test itself. On the other hand, in a scientific project, no one can predict precisely the outcome of a research prototype until the project is underway. Sometimes it may even be challenging to test a prototype with complete real-world input data, as has been the case in certain instances.

On-demand access to a shared pool of customizable computer resources may be provided and released quickly with little administrative effort or service provider involvement using the cloud computing paradigm. In order to profit and enhance the environment, an organization must have a thorough understanding of Cloud Computing's essential features, service models, and deployment patterns (De Bayser et al., 2015). Literature also shows that there are also some challenges for DevOps adoption in the embedded systems domain, including the culture of continuous improvement, configuration management of test environments, deployment process automation, monitoring in the production environment (Lwakatare et al., 2016).

2.5.3. Mitigations to Challenges in DevOps Capabilities Improvement Adoption

It may be costly and time-consuming to convert a conventional product organization to a DevOps one. Although this transition is costly, many fast-growing companies justify the investment since the anticipated advantages outweigh the work and change required to embark on the DevOps implementation path. Some security risks and compatibility problems, along with other issues, may arise due to DevOps, but still, there are ways to tackle them.

There are several strategies that an organization can implement to foster a self-organizing culture, including the formation of cross-functional teams that include functional teams from each area of the software delivery process, the establishment of a blameless post-mortems environment by removing the sense of fear among teams, the establishment of a sense of shared responsibilities, and the introduction of a concept of introducing new ideas with research and testing times. Strengthen employee communication by allowing them to

understand their team and company better, educating them on the advantages of change, affirming the importance of new responsibilities, and allowing each employee to participate in the decision-making process (Olszewska & Waldén, 2015). Additionally, offering a visual CD pipeline architecture that depicts a whole perspective of the CD channel may inspire the team members. Then management, human resources, and administrative departments must rally to their cause with enthusiasm (Chen, 2017). DevOps cultural shift may be difficult for specific individuals in a company, especially when it comes to large organizations. In order to address this problem, studies have suggested that Human Resource Management programs be implemented along with DevOps transformation initiatives (Carturan & Goya, 2019).

For DevOps interpretations to be successful, it is necessary to first confirm DevOps' direction before beginning to ensure the framework and goal clarity. It is necessary to organise the corresponding lectures and training related to team building and develop a sense of DevOps among team members and familiarisation with team building and organisation (Virtanen et al., 2017).

DevOps techniques necessitate the use of the right technology. There are several DevOps tools for different purposes such as Codeship (Continuous Integration, Continuous Testing), Git (Code, Build), Gradle (Build), Selenium (Test), Jenkins (Build, Test, Deploy), HipChat (DevOps team Communication), Puppet (Deploy, Operate), Docker (Build, Deploy, Operate) and eG Enterprise (Monitor) and many others. Organizations will be able to make educated decisions regarding the many forms of DevOps technology and their local demands after categorizing and classifying such tools and their use (Ghantous & Gill, 2017).

A substantial number of tool-related obstacles can be reduced if developers struggling with tool issues work together to establish tool standards (Tomas et al., 2019). Creating a solution, providing plugins linked to well-known goods, and then gathering specific criteria for tools and procedures to be utilized in a DevOps environment are some of the best practices to be implemented (Snyder & Curtis, 2017; Elberzhager et al., 2017).The complexity of existing DevOps tools is also a cited issue, typically due to a lack of comprehensive documentation. As a result, improved documentation on tool usage needs to be encouraged. This would address resource management issues and security settings for tools (Rahman et al., 2019). As a result, best practices for tools such as personal customization must be followed. As an alternative to using the software's default settings, which might lead to problems like insufficient (or excessive) security levels, the best approach here is to customize the tool based on the situation. Scanning tools used in the CD pipeline include a variety of best practices, including

being explicit about the kind of checks to conduct and excluding non-production code from scanning, and correctly setting tools according to corporate rules and not running duplicate tests (Zampetti et al., 2017).

In the current environment, switching to cloud-based services is a modern trend. Customers who use such services can avoid some of the difficulties of utilizing standalone solutions. The introduction of Cloud Computing has not only reduced the cost of IT operations but has also facilitated the concept of continuous delivery, which has the potential to significantly decrease resistance in DevOps processes while simultaneously speeding up the product delivery cycle (Kupsch et al., 2017).

In order to address the difficulties associated with the adoption of practices and standards, researchers typically propose that procedures such as compliance testing, risk management, security, and privacy by design be fully or partially automated, depending on the situation (Steffens et al., 2018). The fast pace of software releases in the DevOps paradigm means that practitioners do not prioritize documentation and log-keeping. However, these are some of the best practice methods in software development, and they must be followed regardless of how quickly a product is released. For example, if automated deployment and testing results are documented, they may be used as evidence in auditing (Mohan et al., 2018). Therefore, studies have discussed using suitable tools and maintaining the required data.

An important DevOps advice is to make it possible for many team members from various teams to collaborate on the same pipeline. The process includes granting authorization and various access permissions to these members in practice. However, strong access management measures are recommended if the output is released into a compassionate and controlled environment. Additionally, modifications to the manufacturing and testing environment must be automated, resulting in developers no longer having access to the production environment (Zheng et al., 2018).

When developing infrastructure-related solutions, it is advised to understand the model first and then code it accordingly. If the infrastructure is inadequate, it may begin with a fundamental but critical component and then actively offer additional hardware resources for the product's CI server so that the software product can constantly incorporate as required so that software products can be delivered at any time (Olszewska & Waldén, 2015).

Infrastructure as Code (IaC) is a method of controlling host, archive, and networking infrastructure in cloud services. IaC is highly recommended in complicated and controlled

infrastructures. IaC allows infrastructure to be evaluated, developed, and delivered, making it a viable option for managing complex systems with privacy and security issues (Zheng et al., 2018). Setting up simulation environments for testing in complicated infrastructures is a suggested approach (Morales et al., 2018). Intrinsic testing may be carried out by developers in these infrastructures thanks to the establishment of everyday simulation environments that everyone shares. This may be extremely helpful for doing vulnerability testing and collecting end-user feedback. The literature has also described a model-driven approach to addressing IoT infrastructure issues (Ferry et al., 2020). Additionally, it gives users the ability to define their security and privacy requirements while supporting automated deployment and other related processes.

One related recommendation is the formation of multidisciplinary teams in an organization. As a result, cooperation between the engineering, security, and functional departments would be very high from the beginning of the process forward. Two additional suggestions are to design security tools to make the feedback loop with other teams shorter and boost developer participation in security activities (Jaatun, 2018). Additionally, a clear and established method of communication between employees should be in place. Manual communication techniques such as emails should be avoided since they increase the work required for communication (Mohan et al., 2018). Automated approaches should be used instead of using manual techniques to notify team members of actions in the processes. For example, rather than having an administrator transfer a message to each individual about testing efforts, successful installations, and so on, the appropriate stakeholders should be automatically informed from a system.

DevOps adoption costs may be reduced by keeping track of team performance, and risk clarification can be achieved by improving communication between developers and management (Perera et al., 2016). More frequent software deployments that do not adversely impact consumers will help gather interactive data on customers and services. Continuous deployment can rapidly repair problems and roll back software to a prior version if any code changes leave it less than entirely functional (Claps et al., 2015). Organizational reluctance to change is a problem for low-evolution teams, including legacy architecture, a skills deficit, a lack of automation, and ambiguous objectives. Highly developed teams overcame organizational buy-in blockers and cultural obstacles. They have developed a technological stack that makes heavy use of automation and have made substantial investments in internal platform infrastructures. Consequently, their biggest roadblocks are legacy architecture and a

lack of skilled workers. In other words, culture is no longer a hindrance for highly developed companies (Cardenas, 2018).

2.6. DevOps Capabilities

By definition, capabilities refer to processes that an organization should be able to carry out (MJÖLL & HELÉN , 2018). Therefore, DevOps capabilities refer to capability enablers, cultural enablers, and technological enablers. In order to improve the state of DevOps report, their processes are supposed to be improved. In pursuit to improving and adding the capabilities of DevOps, many researchers suggest that one should be able to understand the organization in terms of Performance, which is indicated by the DevOps metrics, goals of the organization which is represent by business OKRs and the management of organization which is the champion for change. Therefore, in this subsection we explore the capabilities which if properly propagated, can aid in improvement of the state of DevOps. In pursuit to achieve this, we give consider the following categories of DevOps capabilities:

2.6.1.Continue Delivery Capabilities

Continuous delivery capabilities are DevOps capabilities which aid in building the chain of continuously delivering solutions to the customer. These capabilities are mainly concerned with automating the DevOps pipeline and ensuring that the development and operations teams are working together to achieve a common goal. Also, these metrics aid in ensuring that there is security of the DevOps development pipeline from corruption. They deal with things such as testing, deployment, delivery, integration, and development. Some of these capabilities include:

- Continuous delivery
- Version control
- Shift left on security
- Test data management
- Test automation
- Trunk-based development
- Continuous integration
- Deployment automation.

2.6.2.Architecture Capabilities

Architecture capabilities are concerned with providing a common phrase or language and framework to describe an organization using business terms. Its key focus is to unite an organization through functional boundaries, simple statements that together link various activities, and outcomes and approaches (Mishra & Otaiwi, 2020). In DevOps, this is achieved by ensuring that there is cohesiveness in the team and that they work together and having a rigid DevOps pipeline which supports activities for both teams. The microservices idea makes it possible for the teams to be able to carry out development, testing, and integration continuously through automation of various processes and collaboration between teams. The already stated business DevOps capabilities include:

- Empowered teams
- Loosely coupled architecture

2.6.3.Product and Process Capabilities

The capabilities under this category ensure that the customer's expectations are met, and that the user experience is improved. These capabilities depend on the quality of the DevOps pipeline and the skills of the team. Therefore, in order to improve on this category of DevOps capabilities, the organization has to channel the required resources for professionalism in the technical sector and also invest in the right tools to deliver quality to their customers. Some of the DevOps capabilities in this category include:

- Team experimentation
- Working in small batches
- Value stream
- Customer feedback.

2.6.4.Cultural Capabilities

Cultural capabilities refer to the skills, knowledge behaviours and systems that are required to plan, support, improve and deliver services in a culturally respective and appropriate manner (Wilson et al., 2017). In DevOps this category of DevOps capabilities deals with the likelihoods and the flexibility of the team to change and adopt new practices and cultures. In order to service and build reputation in the business world, you must be businesswise flexible to changes. This involves having like-minded team members and a DevOps pipeline which is flexible to changes. Some of these capabilities include:

- Transformational leadership
- Job satisfaction
- Collaboration among teams
- Supporting learning
- Performance oriented culture

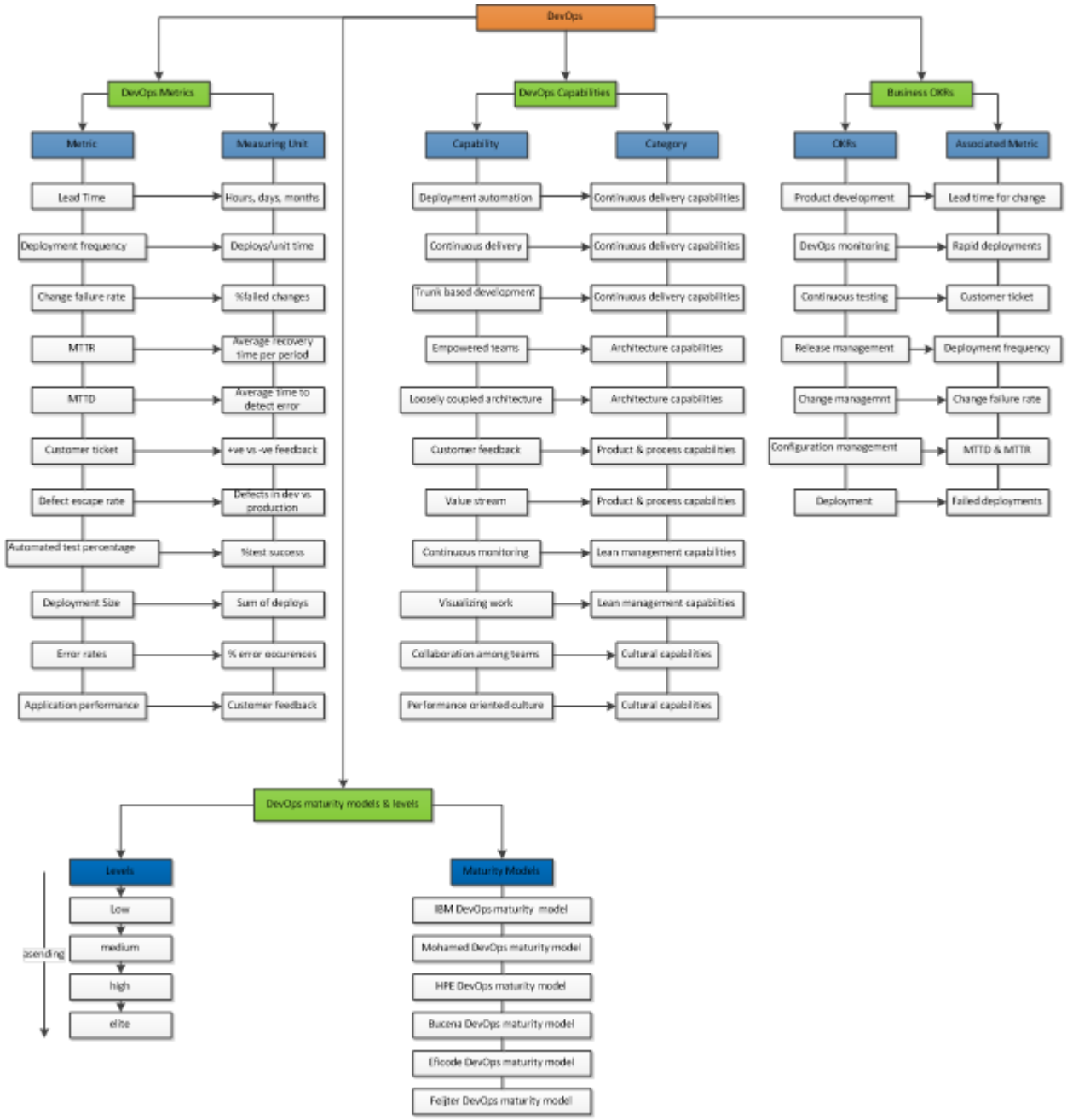
2.6.5.Lean Management Capabilities

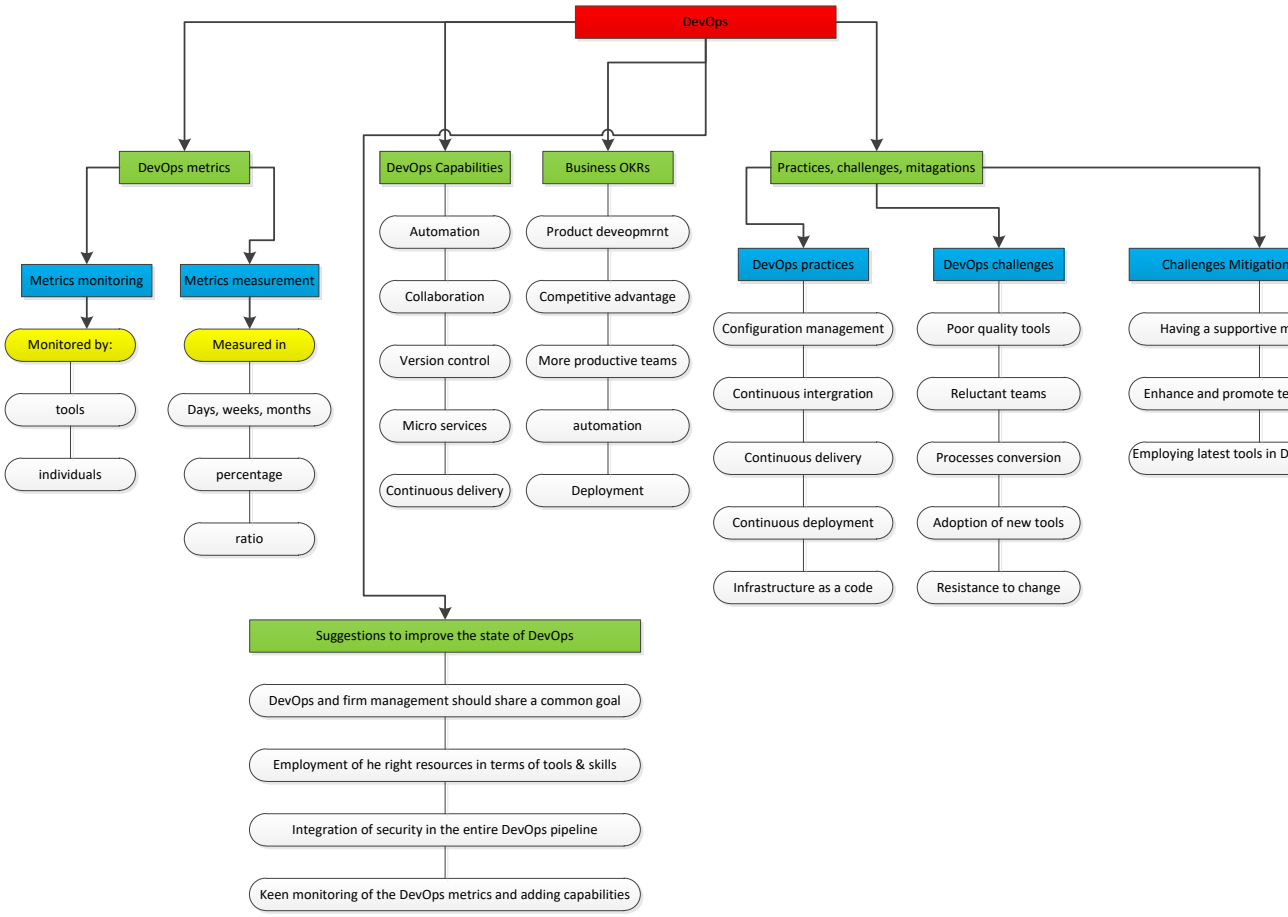
Lean management is an approach to managing an organization which supports the concept of continuous improvement, a long-term approach to work that systematically seeks t achieve small incremental changes in process in order to improve efficiency and quality. Lean management usually allows the organizations to record small but significant changes in their development pipeline (Ahmed & Rashdi, 2020). Lean management capabilities also explain the benefits of having a well aligned to the changing environment conditions and technology. Therefore, the organization supports the DevOps teams in terms of financing and motivation into introducing and embracing positive change in the development pipeline and the DevOps team's environment. Therefore, this category of DevOps capabilities is key to change and growth in DevOps from one level to another. Some of these management capabilities include:

- Visualizing work
- Work in progress (WIP) limits
- Proactive notifications
- Monitoring
- Change approval process

2.7. Related Literature Conceptual diagram

The diagram shows a summary of the related literature. This is based on already discussed building blocks of the thesis which includes the DevOps metrics, team's capabilities and the business objectives and key result. The diagram digs deep and represent the related literature graphically.




DevOps
DevOps metrics
DevOps Capabilities
Business OKRs
Practices, challenges, mitagations
Metrics monitoring
Metrics measurement
Monitored by:
Measured in
tools
individuals
Days, weeks, months
percentage
ratio
Automation
Collaboration
Version control
Micro services
Continuous delivery
Product deveopmrnt
Competitive advantage
More productive teams
automation
Deployment
DevOps practices
DevOps challenges
Challenges Mitigation
Configuration management
Continuous intergration
Continuous delivery
Continuous deployment
Infrastructure as a code
Poor quality tools
Reluctant teams
Processes conversion
Adoption of new tools
Resistance to change
Having a supportive m
Enhance and promote te
Employing latest tools in D
Suggestions to improve the state of DevOps
DevOps and firm management should share a common goal
Employment of he right resources in terms of tools & skills
Integration of security in the entire DevOps pipeline
Keen monitoring of the DevOps metrics and adding capabilities

3. Chapter 3 - Research design and implementation

3.1. Introduction

This chapter describes the research aim and research questions and justifies a suitable research process to answer these questions. This includes the use of a systematic literature Review to gather the data and thematic analysis of this data to answer the research questions.

3.2. Research Aim and Questions

The aim of the research is to identify the metrics useful to characterise the state of DevOps team as well as confirm the team's improvement over time so the team can mature in its implementation of DevOps. This will provide practitioners with a mechanism for making evidence-based decisions on DevOps improvement. The research aim will be achieved by answering the following research questions.

Table 1

Research Questions Expected Outcome

Research Questions	Expected Outcome
RQ1. What are the main DevOps metrics currently in use by practitioners that could be useful for characterising the state of a DevOps team?	It will identify the key performance indicators that will demonstrate to management, why and which DevOps metrics matters the most and how it affects business results.
RQ2. How to measure each metric reliably and consistently? How to monitor each metric & when to measure it (which part of SDLC) and frequency & who measures and monitors?	A table recording the methods to measure and monitor the metrics related to software development, deployment, and delivery, and identify the advantages and disadvantages of different measuring methods and identify the roles and responsibilities of different personnel related to each metrics in different organizations.
RQ3. What are current challenges and issues in measuring and monitoring each metric? What is the cost/effort of measuring/ monitoring?	A table listing the problems that DevOps faces during software development, deployment, and delivery, as well as the remedies and mitigation measures that have been found in the literature.
RQ4. Given a DevOps metrics profile what is a reasonable target profile for improvement and what changes will be likely to achieve this improvement?	The research problem findings will aid in creating insights for State of DevOps with performance, productivity, scaling, and improvements.

The systematic literature review is proposed to gather and synthesise what is currently known about these areas of DevOps metrics and improvement. While there are several studies related to metrics in DevOps and their measurement and some research on DevOps maturity models, the information is quite scattered and needs more profound analysis. A systematic literature review will gather this research together and provide a basis for deeper analysis to achieve the research aim. The following section describes how to conduct an SLR systematically step-by-step.

3.3. Systematic Literature Review

A systematic review is a research technique and procedure for discovering and critically evaluating relevant research, as well as for gathering and analysing data from that study (Liberati et al., 2009). Systematic reviews conduct a thorough search to assess and consolidate all pertinent empirical data in order to give a comprehensive interpretation of study findings. By reading pertinent literature, it may gain a better understanding of the comprehensiveness of the current body of work. The test can be done by a specific theory to generate new ideas by summarising, interpreting, and synthesising a collection of relevant material. Additionally, it is necessary to ensure the credibility and quality of existing literature employing a criterion to identify flaws, anomalies, and absurdities (Xiao & Watson, 2019).

Numerous issues arise as a result of the huge amount of scholarly literature. One is how to fully document and evaluate the status of knowledge on a certain subject. Systematic review is an effective tool for accomplishing the ongoing increase of research, along with the need to synthesise current knowledge in a methodical manner.

Numerous advantages accrue from systematic reviews. First, they provide a concise and comprehensive assessment of the available data on a particular subject. Additionally, systematic reviews contribute to identifying research gaps in our current understanding of an area. Finally, systematic reviews may be used to identify clear answers based on known data and hence do not require more investigation (Tanveer & Peričić, 2019).

A most common systematic review misunderstandings is that they are synonymous with meta-analyses. The phrase "systematic review" refers to research that compile existing data in response to a specific query. A meta-analysis is a review in which quantitative data from primary research are collated using statistical methods (Charrois, 2015). When the objective of the review is to analyse and synthesise information regarding the influence of a certain component, an integrative review technique is unreliable; instead, a systematic review approach should be employed. The remainder of the evaluation should then be guided by the stated aim (Snyder H. , 2019).

A systematic review may be conducted to confirm or reject whether present practise is supported by relevant evidence, to assess the quality of that evidence, and to resolve any lingering uncertainty or variance in practise. These discrepancies in practise may be the result of contradictory evidence, which a systematic review should (presumably) address. Conducting

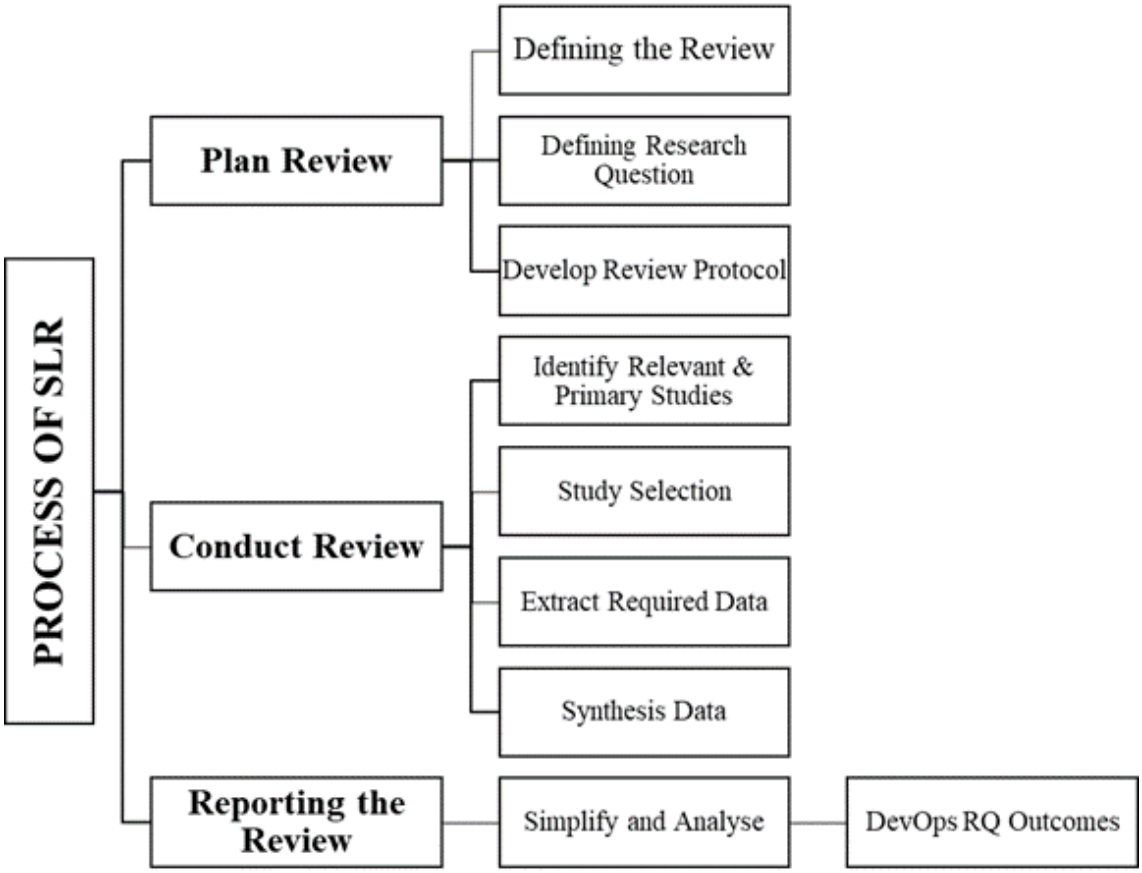
a systematic review may also reveal gaps, inadequacies, and trends in the available evidence, so bolstering and informing future study in the field (Munn et al., 2018).

The nature of this research is conclusive, meaning that the information contained here is useful in decision making or making consultation (Neelankavil, 2015), so, due to the nature of the research issues, types of our research questions, and desire to examine the available literature, SLR is the most acceptable or suitable research approach. The data collected by SLR enables identifying its continuous delivery related problems for DevOps deployment and practises. SLR is utilised to analyse data relating to DevOps and identify research gaps to lay the groundwork for the study. SLR is the ideal approach for collecting research data and synthesising the whole body of knowledge on the subject based on all the facts mentioned above and its benefits to achieve a high degree of accuracy and precision.

3.4. SLR Research Process

The research work follows the steps of “The Challenges and Mitigation Strategies of Using DevOps during Software Development” (Liu & Zhou, 2017). The research work follows a pattern by first introducing the basic concepts and the reasons to adopt the study. Through identifying and filtering information on objectives and key results for DevOps, the metrics playing the central role describes the state of team's DevOps functions, and the primary challenges in different phases to improve DevOps practice. Afterwards, the literature was studied to develop the understandings of the work done previously and to identify the gap in the literature.

Figure 3
Process of Systematic Literature Review



Note. This diagram shows the summaries process of systematic literature review, through identifying and filtering information. From "Procedures for Performing Systematic Reviews" by B. Kitchenham, 2004 and "Lessons from applying the systematic literature review process within the software engineering domain" by Brereton et al, 2007

The research work further describes the research methods including systematic literature review carried out based on Search Strategy, Search String, Resource, Quality Assessment Criteria and Data Synthesis Strategy. The research further has the discussion section to discuss how the DevOps has evolved in the past few years and how different metrics are contributing to its evolution and how are they linked to each other. The research will further focus on the factors that influence the reliability and precision of the research results to identify the validity of the study. The research finally concludes by answering the research questions. Flow of research conduction adopted is shown in Figure 3 (Kitchenham & Stuart, 2007) (Brereton et al., 2007). The detail of Figure 3 is discussed below.

3.4.1.Plan Review

A systematic review begins with the specification of a review protocol that describes the research questions to be addressed the technique of reviewing, which in our case is SLR and the research questions as stated in the introduction of this chapter.

3.4.1.1. Defining the Review

For this stage, it is necessary to summarise all current material regarding the DevOps phenomenon thoroughly and fairly to make more general conclusions about the phenomena than can be drawn from individual studies or as a precursor to subsequent research efforts.

This step was executed through doing keyword research which identified all materials related to DevOps for review. After recording and identification of the materials, I gone through them while summarizing the main ideas related to my research questions. I considered the DevOps metrics, team capabilities, business objectives and key results, monitoring tools, maturity models and common practices that define a successful DevOps team in an organization.

3.4.1.2. Development of Review Protocol

When conducting a systematic review, a review protocol often describes the methodologies employed, such as the context for our investigation, research selection criteria and processes, study quality evaluation checklists, data exploration strategies, and timeframe parameters.

In this subsection there was consideration of the current focus in the DevOps and what is mainly driving the software development methodologies. The research selection involved checking at the relevant processes that are geared towards enhancing DevOps and therefore explored the data to ensure that we have first-hand information. This included taking into

consideration the grey literature and what it had in store for growth and enhancement of the state of DevOps.

3.4.1.3. Research Questions

The formulation of research questions is the most significant activity, and this is done following the introductory section provided in chapter 1. The criteria used in the selection and formulation of the research question were based on the objectives and the research question. It looked at the research problem to ensure that the objectives and research questions fully address the issue in question for the achievement of the research aims and goals.

3.4.2.Conducting the review

Once the protocol has been agreed upon, the actual review may begin, which will include:

- Research identification
- Study selection
- Study quality evaluation
- Progress monitoring with data extraction
- Data synthesis

3.4.2.1. Identify Relevant & Primary Studies

The aim was to use an unbiased search technique and locate as many primary studies as feasible related to the study topics. For example, language prejudice must be avoided, but initial search included all biasing conditions which were later excluded based on selection criteria. Therefore, to identify relevant and primary studies, I did a search through google but this gave both biased and unbiased information. For separating biased sources from unbiased sources, keywords and specific materials such as publications, journals, and company reports were encouraged.

3.4.2.2. Study Selection

Once the possibly relevant primary studies have been identified, it is necessary to determine whether or not they were indeed relevant.

It is generally considered essential to assess the quality of primary studies to provide detailed inclusion and exclusion criteria, investigate whether quality differences are an explanation for differences in study results, guide the interpretation of findings, determine the strength of inferences, and guide recommendations for further investigation. This is accomplished by using Table 2 that will be discussed later in this chapter. Hence, It used methods such as taking the reviewed studies to ensure that they are relevant to the topic. In addition, it considered how the study was referred to by other researchers to check and determine its

validity. This therefore, resulted to a table of the relevant studies as discussed later in this chapter.

3.4.3. Data Extraction

In this step, the researchers' goal is to construct data extraction forms that will correctly capture the information they get from the primary studies they conducted. Table 2 contains the specifics of the data extraction process. I created a data extraction form which helped me in keying in the collected data and ranking it in terms of its relevance. In this, I recorded the details of the source and also the relevant concepts that are described in the source based on my research topic.

3.4.4. Data Synthesis

When it comes to data synthesis, it is the process of gathering and summarising results, which in our case were primarily descriptive and non-quantitative. I used conceptual analysis in generating various concepts available in the sources used. The research was mainly qualitative and did not entail figures which would require use of formulas. Therefore, I categorized the data in terms of themes including the maturity models, metrics, objectives and key results and the team capability's themes related to DevOps.

3.5. Data extraction

The data extraction form presented in Table 2 was used to extract the data. Primary studies were examined using predefined research questions. The most relevant terms for each research topic were collected from each primary study. One researcher extracted the data, which may have influenced its validity. The form was the mostly preferred simply because it assisted in filtering data based on the thesis themes, the year of publication, the research question it is related to and tools and techniques that it used in gathering the information presented. This aided in ensuring that there is no biasness in the information presented in this chapter. Many studies were excluded from review by the use of the information presented on the table.

The form used various factors in considering the information in order to ensure that each aspect of the research questions is well catered for explicitly.

Table 2

Data Extraction Form

Author	Determine name of the author/authors.
Paper Title	Determine name of the article.
Year	Year of publication.
Type of the article	Determine article.
Research Question	Determine if specified.
Research Method	Determine if mentioned.
Data Analysis Type	Qualitative, Quantitative or Mixed Method Analysis.
Research Type	Validation, Evaluation or Solution.
Application Domain	Application for reporting challenges and validation of tools and techniques.
Tools and Techniques	Identify the tools and techniques used to facilitate the DevOps metrics.
Challenges & Practices	Challenges and barriers in software development and practices adopted to overcome barriers.

3.6. Data Collection

3.6.1. Search Strategy

Search strings were created based on research questions and objectives by combining the keywords using the Boolean operators "AND" and "OR." Keywords identified for this thesis are Continuous, Development, Integration, Delivery, Deployment, Monitoring and DevOps.

3.6.2. Database Selection

Investigators and information specialists who explore relevant references for a systematic review (SR) advise finding several databases and trying more ways to identify all literature related to the topic of interest (Bramer et al., 2017). A thorough search strategy should involve multiple databases, registries, sources of grey literature (Tanveer & Peričić, 2019). A combination of multiple databases would yield more articles than Medline alone, which can help us to make accurate decisions (Zhao, 2014).

During the initial stage of following the search string, filtering is not necessary. The first four databases are selected for primary search related to relevant studies as they are all accepted academic research databases and provide most of the data required on DevOps. However, there was not much literature available for DevOps monitoring, delivery and maturity models so some other journals databases were referred for literature of few studies based on AUT library search results without particularly specifying the search strings as there were very few articles adopted from all these databases one or two from each including:

- ScienceDirect
- ACM Digital Library
- IEEE Explore
- Springer Link
- Emerald insight
- Wiley Online Library
- Google Scholar

Grey literature can provide a systematic review with an additional source of rich information, depending on the topic and nature of the research (Institue of Work & Health (IWH), 2008). Grey literature is a significant source of information for large-scale review syntheses since it contains a wealth of information (Godin et al., 2015). Some grey literature review from Amazon, Puppet and Atlassian is also used.

3.6.3.Search Strings

A search string is a combination of keywords, truncation symbols, and Boolean operators entered in the search box of a library database or search engine. Search strings are used to find files and their content, database information and web pages. A search string may include keywords, numeric data, and operators. It is generally considered necessary to optimize the search results based on the research needs.

The search strings defined in Table 3 are based on the research questions defined and described in chapter 1 and 3. The difference in search strings are due to the limitations in the Boolean search strings design allowed by each database. The thesis is using 4 main databases as main databases for initial search to conduct SLR. This initial search is based on the criteria defined. Moreover, it uses some articles and papers that do not include these databases and belong to the other databases or are considered grey literature.

Table *3*
Search Strings Used in Databases

Database	Search String
ScienceDirect	(Development AND Deployment AND Delivery in DevOps) AND (DevOps Maturity) AND (Metrics) AND (Monitoring) OR (Challenges) AND (Solutions) AND (Software)
ACM Digital Library	(Development AND Deployment AND Delivery in DevOps) AND (Metrics) OR (DevOps Maturity) AND (Monitoring OR Measure) AND (Challenge OR Problem) AND (strategies OR Mitigations OR Solutions) AND (Software)
SpringerLink	(Development AND Deployment AND Delivery in DevOps) AND (Metrics) OR (DevOps Maturity) AND (Monitoring OR Measure) AND (Challenge OR Problem) AND (strategies OR Mitigations OR Solutions) AND (Software)
IEEE Explore	(DevOps) AND (Development) AND (Deployment) AND (Delivery) AND (Monitoring) OR (Measure) AND (Challenges) AND (Mitigation) OR (Solution) AND (Metrics) AND (Maturity)

3.7. Inclusion and Exclusion Criteria

In order to find the relevant research, the following inclusion/exclusion criterion is utilised. The search results helped us to decide whether or not to add the article.

3.7.1. Inclusion Criteria

- Published studies from 2015 through 2021 as the development of DevOps actually started from 2010 and most of the previous reviews undertaken are being updated in the recent time span studies.
- Literature available in the English language.
- Full text literature.
- High quality literature, it recognized based on the amount of the citation or download.
- If the title or abstract describes relevancy in DevOps.
- Literature that focuses on the deployment and development in DevOps organization.
- Literature that focuses on the organizational aspect related to organisation strategies and production environment in DevOps.

3.7.2.Exclusion Criteria

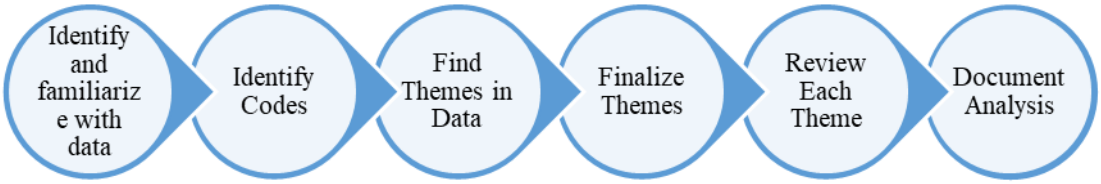
- Literature that does not give any information on the subject.
- Literature that is not available in the English language
- Literature that is not available in full text
- Literature that is duplicated and ambiguous
- Only abstracts

3.8. Thematic Analysis

Thematic analysis is the search for themes that emerge as crucial to the description of a phenomena. The approach entails identifying themes by "careful reading and re-reading of the data." It is a type of data pattern recognition in which emergent themes become the categories for analysis (Fereday et al., 2016).

The purpose of thematic analysis, as it is typically used to systematic reviews of qualitative research, is to identify common themes that emerge from the findings of the many studies. Because the Results sections of qualitative investigations are usually arranged around themes (or their equivalents), thematic analysis serves as a mechanism for collecting and comparing the data from the individual studies.

Figure 4
Thematic Analysis of Qualitative Research Data



Note. This model demonstrates the thematic analysis of qualitative research data.

Thematic Analysis begins with identification and checking the integrity of overall data. Afterwards, the elements of interests are identified. Then, themes are identified and data analyzable under each theme is located. Then, specific identity of each theme is verified and finally the patterns are analyzed, and inferences are drawn (Figure 4).

3.8.1. Quality Assessment Criteria

The study's quality was evaluated by reviewing whether the previously selected main studies met the minimum quality standards of the current research. The research did not include any papers that did not fulfil the criteria. The primary study's final articles have been peer-reviewed and accepted.

The quality evaluation checklist is given in Table 4. Each statement is evaluated on a three-level numeric scale with yes (2 points), partial (1 point), and no (0 points) each having a corresponding number value. For studies with a maximum of 22 points and a minimum of 0, 11 questions are on the assessment. To make the primary studies, articles required to have minimum of 11 points. Thus, any articles with a rating of 10 or lower were eliminated from the research due to their poor quality. The relevance of the papers for this study was considered as a major aspect of this research. Even if a writing was exemplary, it may have been omitted if it didn't relate to this research. This study's research uses a checklist developed to locate relevant articles, and the goal was not to rank articles by their scored points. The objective was to remove those publications that did not assist the research in a substantial way.

Table 4

Checklist Defining the Quality of the Primary Studies

#	Questions
1	Are the aims clearly stated?
2	Was the study designed to achieve these aims?
3	How credible are the findings?
4	How has knowledge or understanding been extended by the research?
5	How well has diversity of perspective and context been explored?
6	How clear are the links between data, interpretation and conclusions – i.e. how well can the route to any conclusions be seen?
7	Are the statistical techniques used to analyse the data adequately described and their use justified?
8	Do the researchers discuss any problems with the validity/reliability of their results?
9	Is the purpose of the data analysis clear?
10	Is there enough evidence to draw a conclusion?
11	Are the challenges or mitigation strategies in relation to DevOps discussed in the paper?

3.8.1.1. Snowball Sampling

For databases search, it is possible that important information may be missed. So, snowballing approach was utilized with a database search as the strategy. Additionally, the database's search strings may fail to provide all linked research. The various databases employ distinct search techniques and interface types, which are exclusive to certain search strategies. According to (Wohlin), it is difficult to construct a search string that covers all relevant research, as the terms employed are not consistently applied. A lot of irrelevant articles may be found using database searches. Peer review is improved by the snowballing process in the selection of research. It is particularly helpful to compile information for a systematic literature review when using snowball sampling. This graphic depicts the snowballing method, which can be used to take a step backward or a step ahead.

Backward and forward snowballing has been done. Webster and Watson are credited with having invented the idea of forward and backward search. The examination of references discovered in keyword search results is called backward reference search. One can investigate this research due to these origins. An extra search on references is performed for references that reference other references. Keywords that were previously employed are utilized combined with new search results and another search. Conducting a search using the search terms will provide you with the newest research that is useful. The idea of forward reference search is referred to as forward search.

The snowballing should be conducted iteratively both forward and backward. Studies that are recently published or newly created should be incorporated into the first list of studies. New research should be found and included in each successive iteration until the snowballing is complete. Setting the snowballing process to commence relies on the area of research. Wohlin believes that the primary studies of SLR should contain those research studies found in the initial set. Various research from different authors and publishers should be included in the initial batch of papers. Data was acquired by conducting database research and locating studies from that study. Studies are scanned to identify relevant papers. Then, Google Scholar is used to find all citations to those papers. A single stride conducted a backward and forward snowballing. The information was gathered using a total of thirty different main research, each of which had a search string attached. More papers were identified after applying the inclusion/exclusion criteria. The second study was done, and a duplicate paper came out. This group of research did not fulfil the inclusion requirements as specified above, thus only one round of snowballing was completed

3.8.1.2. Selection of Studies

Given the general search term used in the databases, the total number of publications was 10727. However, these publications included duplicates which owe to the fact that different databases were used for the process. Nevertheless, after subtracting the duplicates, the number was still high and at precisely 1063 publications. To reduce the number constructively, a more objective process must be use.

The process involves a series of activities that filter the relevant publications in line with the research question. Also, being more specific on the search string is essential in reducing the number of publications to the most relevant literature. Another activity is eliminating all the search results written in different languages and maintaining those written in English. The titles of the publications and the abstract are relevant when selecting publications that focus on the research question. Choosing materials that look at different aspects of DevOps is essential to ensure that more comprehensive literature research is conducted. Finally, the selected publications are downloaded then skimmed to classify them with their relevance.

Primarily, the selection criteria and process begin from the general search term DevOps. As outlined, reducing publications start by eliminating the number of duplicates, then non-English publications, reviewing titles and abstracts, and inspecting the whole text for relevance.

Table 5

Articles From Each Database

Database	Search Results with Search String
IEEE Explore	7217
ACM Digital Library	2576
SpringerLink	18
ScienceDirect	916

Table 6

Search Results from Different Databases after Inclusion & Exclusion Criteria's

Steps	Count of results in selected databases
Overall Results	10,727
Relevance by tittle	2562
After deleting non-English publications	1803
After deleting duplicates	1063
Relevance by abstract	142
Relevance by full text	86

The Table 6 displays the process through which the relevant publications to the study were selected. The method identifies 86 publications as being relevant to the research questions for relevant SLR. The actual total studies included are 127 among which 41 are a part of introduction and research design chapters. The reason of exceeding references is the large number of data even after applying Boolean search strings. The total count replicates the total number of literatures found from various databases using the search term DevOps on the respective search engines.

4. Chapter 4 - Finding and Analysis

4.1. Introduction

This chapter involves the in-depth evaluation of the data collected through academic literature review and grey literature review. Grey literature review aided in reduction of the publication bias since it is one of the immediate or recent sources of information. It depicts what is going on in the current market and therefore foster a balanced picture of the available evidence while increasing the comprehensiveness and timeliness (Paez, 2017). It considers the publications not yet published and also the ongoing activities in the organizations which are using DevOps in their product development processes and therefore reduces publication biasness. These include journals, publications, case studies, websites such as amazon.com, and reports. It initially sets out the descriptive analysis of DevOps metrics considering the objectives and key results, discussing the characteristics of DevOps, how various software companies use DevOps to elevate their value, how DevOps can be improved, and their capabilities widened. It also discusses the DevOps taxonomy, considering bridging, process taxonomy, and the DevOps tools taxonomy. Lastly, it critically evaluates the field research results and evaluates their linkage to the existing literature on DevOps metrics, Objectives, and Key Results. This otherwise suggests the basis on which DevOps practices can be improved to make them more productive and valuable in a production environment.

4.2. Findings from the Systematic Literature Review

4.2.1.Lead time for change (Time from commit to deployment)

I found more than three definitions of this metric, as depicted in Table 7. They talked about the measure of unit time taken to go from the code committed to the code successfully running in the production (Creasey & Hiatt, 2018). This is after the development has been successfully done, and therefore testing is being done and the deployment hence corresponding to the testing to deployment stages of software development and life cycle as depicted in Figure 5. Therefore, the metric relates to all work done after changes have been suggested to the deployment of the application to the final user. The whole CI/CD process is automated using tools such as GitHub, with monitoring analysis being done by the project manager.

This metric tracks a project's time to go from inception to implementation. This does not necessarily mean that this is the time for the whole project to be implemented but time to develop small features as requirements are being gathered in an agile environment. A shorter

lead time for change shows the team's efficiency in releasing new solutions to customers. This also dictates whether the team can quickly adapt to customers' feedback, give new features, or make new improvements. Lead time can be measured in hours, days, weeks, months, or years. According to the results obtained, lead time measured in hours, days, or weeks implies an efficiency unless in the one which is measured in months or years. This metric is also used to quantify the rate the team can implement changes. Therefore, the mean lead time for change is mainly used in accessing the team's throughput and hence a key consideration in improving the state of DevOps measuring and monitoring.

Table 7
Table Metric: Lead Time for A Change

How Measured	Measure as unit time a team takes to go from code committed to code successfully running in the production.
Part of the SDLC	4 - 6
Frequency of measure	Once per every release.
Who measures/ monitors	DevOps engineer.
Relates to which capabilities	Continuous delivery, continuous monitoring, and continuous testing and deployment.
Typical range at high maturity	For elite teams which depict high maturity, it takes less than one day.

4.2.2.Meantime to recovery (MTTR) – Time to Recover from a Bug

Through analysing various data from my literature review, I found various definitions of this metric, as depicted in the DevOps metrics table in the appendix. They all refer to the time it takes to recover from a system failure (Vialle, 2018). After the system has been deployed for the final user, it is always used to find bugs that may fail. Some tools are purposely for detecting system failure, and the support engineer regularly monitors them. The work of detecting system failure and correcting it happens at the last stages of software development and life cycle, which are deployment and maintenance.

Therefore, for a solid customer base, the meantime to recovery should be as low as possible, most preferably less than one hour. There was no apparent argument on the one who is supposed to monitor and measure this metric, but a support engineer is mentioned as the one to be alerted in case of failure. The capabilities of the team associated with this metric are mentioned above. The measure of the highest performance for this metric in terms of maturity is less than one hour, as depicted by DORA google metrics though it is not a peer-reviewed source.

Failure is considered part of life, and therefore contingency plans have to be made on how to deal with those failures. It is evident that sometimes some software deployments fail, and therefore it is vital to be able to recover quickly. Therefore, MTTR measures the time it takes from detecting a problem to when the services are restored to their normal state. Mainly, recovery is made through rollback to avoid changes that have led to degraded services. MTTR is a vital metric when it comes to customer Service Level Agreements. If MTTR takes longer than a few minutes, it can result in a lousy will and loss of revenue. This can be perfected by ensuring that you have good CI/CD application monitoring tools and solid incident response and rollback procedures. Therefore, to improve the state of DevOps metrics measuring and monitoring, the procedures and tools used should be highly considered, as shown in Table 8.

Table 8

Table metric: Mean Time to Recovery

How Measured	It adds up all the downtime in a specific period and divides it by the number of incidences.
Part of the SDLC	4 - 7
Frequency of measure	A span of time when the system was down.
Who measures/ monitors	It is automated. The support engineer monitors it.
Relates to which capabilities	Continuous monitoring, Continuous integration, and automation.
Typical range at high maturity	Less than one hour for elite teams, which depict high maturity.

4.2.3. Deployment frequency (Frequency of Releases)

I found various resources discussing this metric, among other metrics key to DevOps. According to the sources, deployment frequency tracks the frequency with which code increments are deployed to testing, staging, and production (Lawrence, 2021). This is what measures how productive the DevOps team is.

Under high maturity, there are several deployments per day. The team capabilities demonstrated here include automation, continuous deployment, and continuous monitoring. There were no sources that indicated the part of SDLC where the metric is involved, but since it involves testing, staging, and deployment, it happens at the testing, deployment, and maintenance phases of SDLC. Accordingly, it was evident that this metric is monitored by using specific tools integrated with the CI/CD, and then the data is analysed by the DevOps engineers and other supporting staff to gain more insights on the team performance.

This measures how quickly a team can deliver new features to production. Incomplete features are not relevant since they do not add value to the customers or the organization. If the

software is not delivered, there is no customer feedback or satisfaction, hence the need for smaller but frequent releases. Smaller but more rapid deployments keep feature release minor and decrease the likelihood of the presence of bugs. A well-productive team will deliver a couple of releases per week, sometimes several per day, which indicates the presence of quality CI/CD automation tooling. However, in inefficient teams, this can range from months to years which is not economical. Therefore, in pursuit of improving the state of DevOps, the quality of tools and expertise needed should be considered to ensure that there are more frequent deployments. (Table 9)

Table 9
Table metric: Deployment frequency

How Measured	Surfacing data from continuous integration and continuous deployment tools.
Part of the SDLC	6 -7
Frequency of measure	After each release
Who measures/monitors	It is automated using tools such as Jenkins and CircleCI.
Relates to which capabilities	Automation, continuous deployment, continuous integration, continuous monitoring, and continuous delivery
Typical range at high maturity	Several deployments per day.

4.2.4.Change failure rate (% of Deploys with Bugs)

As depicted in the table about DevOps metrics in the appendix, I found various but related arguments about change failure rate as a metric of concern for the DevOps team. The sources change failure rate measures and compare the changes that led to degraded services (Adam, 2022). Changes might either come from poor application performance or customer feedback. There were no sources giving details of the stages of SDLC where the change failure rate is a factor of consideration but according to the knowledge and considering its activities, and how it is approached, it starts from the development stage up to deployment. The best rates for high maturity teams classified as elites were found to be at a rate of less than 15%. For monitoring, automation tools give results that the technical support engineer otherwise analyses to update on CI/CD pipeline, or the tools and resources being used. Therefore, the research showed that the change failure rate could best be controlled by employing the right resources at the right time. This is the percentage of the changes delivered to production but failed, or either had defects or bugs.

Changes with severe defects mainly lead to rollbacks, affecting other metrics such as revenue and service uptime. Therefore, the delivery of rapid more minor releases is a sign of effective DevOps processes and a healthy culture. Frequent failures can indicate poor code quality, poor

testing, undue pressure on teams to deliver a feature quickly, and other development pipeline defects. In a perfect world, the change failure rate is close to zero. In the world of reality, the digits are a little bit lower. Two-digit change failure rate implies that software has many bugs, or the releases are not small enough to support a better success rate. Therefore, teams should be well equipped to ensure that they reduce the rate of change failure. (Table 10)

Table 10
Table metric: Change failure rate

How Measured	It took a percentage of changes that resulted in failure or system degradation.
Part of the SDLC	4 -7
Frequency of measure	Measured after each change is made.
Who measures/monitors	Automated and monitored by the technical support engineer
Relates to which capabilities	Continuous integration, continuous monitoring, and automation.
Typical range at high maturity	Rate of between 0 – 15%

4.2.5.Defect escape rate

From the literature review, more than five sources identified and described defect escape rates. According to the sources, defect escape rate is the effort of identifying the rate of issues identified after the software is in the production environment (WATSON, 2017). Therefore, this metric relates to the work that id did to make sure that there are minor bugs possible and could have found their way to the production environment. A team's skills and capabilities that can have the best defect escape system have been listed in Table 11. The defect escape rate for a high maturity team is at least 75%. Defect escape rate gives a clear definition of how teams are capable. Therefore, the SDLC stages include testing, integration, and deployment stages.

However, compelling the DevOps pipeline is, there will always be defects. These defects can either be detected during the development or testing stages of the pipeline, with the worst case being experienced when the end-users detect them. It reflects the number of defects found in production during and after software deployment. It helps in identifying cracks in the software development process. Cracks are points where these defects penetrate and signal that the quality process should be optimized and tightened. This is measured by comparing the defects found in the preproduction and production phases.

Table 11

Table Metric: Defect escape rate

How Measured	Comparing the number of defects found in the preproduction phase and the production phase.
Part of the SDLC	3 -7
Frequency of measure	During preproduction and production per each release.
Who measures/monitors	It is monitored by the DevOps engineer, Developers, and system architects.
Relates to which capabilities	Automation, Architecture, Code maintainability, and cloud architecture.
Typical range at high maturity	A rate of 75 – 100% is ideal for elite teams.

4.2.6.Customer Ticket Volume

Through thematic analysis, I found two articles that extensively looked at this metric and how it is used to measure the quality of the work done by the DevOps team. According to various sources, customer ticket volume measures how well the product is released to the end-user market. This metric work is mainly concentrated on the deployment and maintenance SDLC stage. The customers express their opinion on the product developed and suggest changes (Robson, 2019). High maturity teams display high end-user acceptability since they have the experience and the skills required to ensure that the product is acceptable to its users. There are various capabilities concerned with customer ticket volume, including code maintainability and cloud accessibility. Customer ticket volume is automated and is done after the release of a product or after a product upgrade. Therefore, it is frequent after a new product is launched or an upgrade is deployed. Some of the platforms used in the analytics of such data and the application include cloud firebase analytics, google play, where there are reviews, Bitbucket, among others. Therefore, customer volume is a critical metric that requires more attention.

This is concerned with the measure of end-user satisfaction. Bugs and errors can always find their way to the end-user, and therefore the customer can either contact the support or share their feedback. Therefore, the number of customer tickets presented as problems or bugs is an essential indicator of application reliability. For measurement, many reported issues indicate major quality issues, while a smaller number of reported issues indicate the robustness of the application. A support engineer can do this analysis and support the operations team. This helps improve customer or end-user satisfaction and therefore contributes to the improvement of the state of DevOps measuring and monitoring. (Table 12)

Table 12
Table metric: Customer ticket volume

How Measured	By analysing the customer feedback and the number of reported issues.
Part of the SDLC	7
Frequency of measure	After every release for a set of specified periods.
Who measures/monitors	Automated and then analysed by the support engineer
Relates to which capabilities	Continuous delivery, automation, continuous integration, and continuous monitoring.
Typical range at high maturity	The number of issues reported is minimal.

4.2.7 Summary of DevOps Metrics Findings

The table below shows a summary of the finding of the 10 selected DevOps metrics based on their relevance and their ability to help in team growth. The table is arranged in order of the importance of each metric. The table also has inclusion of the search count and sampled article sources. The first four metrics are the commonly used ones in rating and also ranking DevOps team and they all revolve around the product development and the final output.

DevOps Metric	**Count**	**Sample Sources**
Lead time for change (Time from commit to deployment)	3750	(Creasey & Hiatt, 2018), (Sallin et al., 2021), (Forsgren & Humble,, 2016)
Meantime to recovery (MTTR) – Time to Recover from a Bug	138	(Vialle, 2018), (Kljaić, & Šćepanović, 2022), (Davis, 2019)
Deployment frequency (Frequency of Releases)	3130	(Lawrence, 2021), (Snyder & Curtis, 2017), (Melgar, 2021)
Change failure rate (% of Deploys with Bugs)	3520	(Adam, 2022), (Airaj, 2017), (Rzig et al., 2022)
Defect escape rate	1040	(WATSON, 2017), (Mohamed, 2015), (Snyder & Curtis, 2017)
Customer Ticket Volume	1030	(Robson, 2019), (Senapathi et al., 2019), (Cito et al., 2018)
Automated Test Percentage	7320	(Maroukian & Gulliver, 2020), (Gunja, 2021), (Angara et al., 2018)
Deployment Size	7600	(Torble, 2019), (Singh, 2021)
Error Rates	5250	(Artač et al., 2016), (Lwakatare et al., 2019)
Mean Time to Detection	160	(Bezemer et al., 2019), (Gunja, 2021), (Hüttermann, 2012)

4.3. DevOps Capabilities

4.3.1.Collaboration

Initially, the Dev and Ops teams worked independently, and therefore this had many barriers such as malfunctioning applications, extended deadlines, thousands of code errors, and heated discussions among the two teams. However, with the emergence of the DevOps, there was the erosion of internal boundaries and, hence, both teams' biding. Therefore, this led to encompassing everybody involved in the process of application development, quality assurance, and delivery (Mumbarkar & Prasad, 2022). However, the collaboration came along with other requirements such as sharing tools and updating the DevOps pipeline. Therefore, to have the best collaboration, the right tools ought to be installed in the development and operations environments.

Collaboration leads to faster product development where design, development, testing and deployment are taking place simultaneously. Collaboration being a greater tool into microservices needs advanced tools which ensures that both teams work to their best. This includes having super-efficient integration tools, testing tools, and deployment tools.

4.3.2.Automation

The DevOps pipeline is automated. Therefore, various tools are involved in the automation and the connection of the separate teams. As shown in the tool's taxonomy, some of the tools required for automation include orchestration tools, configuration, provisioning tools, integration, and testing tools (Shah, et al., 2018). These tools are costly, but for a team to be classified as elite, it should have the best tools and be an everyday cloud user. Through this, the state of DevOps monitoring and measuring is improved.

From the findings, emergence of cloud computing is one of the enablers of DevOps. Therefore, in order to ensure that the state of DevOps in organizations is improved, the organization has to invest a lot in cloud computing. Tools backed up with expert team members will soon make the DevOps a more improved and efficient one in software delivery. Analysis of the findings show that some of the most efficient teams such as google have fully explored the best tools and employed the best skills in being best at software delivery and support to their customers.

4.3.3.Continuous Integration

Continuous integration expands the agile nature of the DevOps processes. It provides room for continuous testing while fixing through immediate feedback. Therefore, DevOps ensures that everyone remains on track since they allow one to know who made changes and what changes

were made. This otherwise helps in knowing the impact the software has on the end-user. Additionally, it promotes collaboration since the developers and the system support teams know whom to communicate with in case of any changes (Senapathi et al., 2018). Consequently, it helps improve the state of DevOps reporting on measuring and monitoring metrics and, hence, is compelling.

Analysis shows that continuous integration ensures that the software is always in use even when it is under development. There is also a chance for all the teams involved in the development to familiarize with the software early and therefore subsequent improvements take low turnaround time and other processes such as testing and deployment since the teams are familiar with the individual modules from various development segments.

4.3.4.Continuous Testing

Testing was restricted to the concerned team in earlier development pipelines, but now, testing is continuous since developers do unit testing while developing. When the developers pass the correct code to the testing teams, it is passed along with test data, enabling the testing team to be on the same page with the production environment (Badshah et al., 2020). Therefore, this increases the number of deployments and reduces the number of bugs and errors in an application. One of the bottle necks that makes the software deployment lag is the exchange between the development and testing team. Therefore, with continues testing, there is reduction in time spent to do various testing and recommend changes to the software product.

4.3.5.Continues Delivery

Continuous delivery is the ongoing DevOps practice of building, testing, and delivering improvements to software code and user environments with the help of automation tools (Battina, 2021). The key outcome for continuous delivery is that the code is always deployable. In some cases, some developers risk carting their code to the end-users and depend on the end-user feedback in watching the real-time performance of the applications and work to minimize the glitches (Mohamed, 2016). This, therefore, has a significant impact on reducing the change failure rate and increasing the number of deployments per unit time. This acts as an improvement measuring and monitoring the state of DevOps. From the analysis of practices of various teams such as Spotify which is based on business teams called squads which are involved in various functionalities such as searches, players, playlist among others. Therefore, analysis shows that continuous delivery is as a result of microservices which is facilitated with existence of different teams working in different functionalities.

4.3.6.Continuous Monitoring

Continuous monitoring deals with identifying failures and fixing them. This helps the developer enhance the visibility of application performance, compliance, and security threat concerns by the use of automated processes which leverage specialized software tools to empower the DevOps teams (Rafi, et al., 2022). Various tools are used in continuous monitoring, including App dynamics, kami among others. These tools help monitor infrastructure, network, and application in the production environment and hence a vital component in ensuring that the deployment frequency is increased (Plant et al., 2021). This, in turn, improves the state of DevOps metrics monitoring and measuring, leading to a more productive team hence the growth to the elite class of DevOps team performance.

The analysis of the finding shows the relationship between continuous monitoring capability and change failure rate metric. With continuous monitoring, it is easy for the teams to identify the defects and therefore restore the system in case of failure within the shortest time possible. This therefore increases the efficiency of the systems deployed and already running in the production environment.

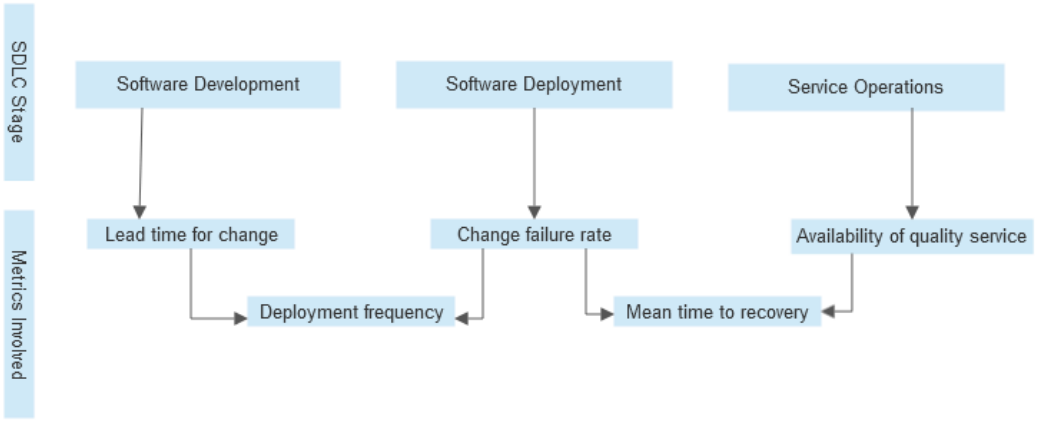
4.4. Descriptive analysis

In the context of this research, descriptive analysis is used for describing the collected data and trying to answer the research questions from the perspective of their validity and frequency of occurrence. In this case, the DevOps metrics and their rate of adoption and the organizations that have adopted them are considered. These organizations are further analysed to give an overview of the performance of DevOps and what is required for excellent performance.

From the data collected, it was evident that usage of DevOps metrics realizes some benefits towards the business's success. In conjunction with this, the DevOps team should be familiar with each metric, where it should be applied, and how that metric can be improved to elevate DevOps capabilities. The following diagram shows a graphical representation of the four key metrics and the stage of software development they should be applied to.

Figure 5

DevOps Metrics and Software Development Stage are Most Valuable



In order to successfully achieve high targets in terms of DevOps metrics success, the collected data showed that the DevOps framework should be well understood and laid down from low to the elite performing DevOps teams. According to the CA technologies research (Ravichandran et al., 2016), out of 100 percent of the enterprises that try to adopt DevOps, only 20 percent of the organizations successfully adopt them. The question remains why is it so impossible to adopt DevOps? According to the moonlight research and findings (Ko, 2017), various roadblocks hinder various enterprises from adopting DevOps. These roadblocks are centered around technology, processes, and people. Measuring the strengths of the challenges and analyzing them shows that the organization can have a better approach to DevOps and maneuver through the tides of adopting innovative technology (Leite et al., 2019).

Also, the existing DevOps can realize their goals through crucial monitoring of the four primary DevOps metrics and objectives alongside the key results. This can be achieved by taking the right path, designing a roadmap, and following a structured implementation procedure. Having an outlined procedure and aligning to the business needs and priorities will be vital to having a successful DevOps with all set goals achieved with precision. DevOps metrics play a crucial role in ensuring that enterprises that adopt them thrive and get the most out of them. This is because usage of DevOps leads to happy and satisfied customers and hence increases the customer value. DevOps metrics are the pillars that measure and show how quality customer services are and should be handled with care. Therefore, proper performance in DevOps metrics means improved capabilities and productivity.

From the analysis of various articles and surveys dating from 2010 to 2021, successful implementation of the four key DevOps metrics resulted in business success in various aspects,

fostering success in the involved enterprises (Rowse & Cohen, 2021). The information in Table 13 represents feedback from critical stakeholders such as managers, project managers, clients, among other key personnel in any enterprise concerning the impacts of withholding DevOps metrics and being in line with objectives and the organization's key results. The data collected involved measuring customer feedback, customer satisfaction, and studying the trends of business returns.

Table 13

Commentators and surveys results analysis

Business Outcome	Commentators	Surveys	Total
Faster Delivery	14	6	20
High Quality	13	7	20
Lower Cost	12	7	19
Higher Revenue	4	5	9
Reliable IT infrastructure	3	8	11
Improved Business Efficiency	3	4	7
Higher Customer Satisfaction	8	5	13

Being one of the critical considerations, DevOps metrics have a particular order of implementation to realize the best out of them, according to the DORA 2018 survey. Among the four-development metrics, two critical metrics need to be given more attention in order to realize massive success, according to the field findings. These metrics are the lead time for change and deployment frequency. Therefore, they are priced and should be given more time than other metrics.

The measure of DevOps metrics success is critical for ensuring that your DevOps are performing and adding value to the business' progress. This can be a proven way to know the efficiency and the effectiveness of your DevOps team and products delivery to the market. The Table 14 shows the metrics calculation and their results according to the results obtained from various sources.

Table 14

DevOps Metrics Success Measure for Various DevOps Team Growth Levels

The aspect of Software Delivery Performance	Elite	High	Medium	Low
Deployment frequency. This discusses how often an organization deploys code to production or releases it to end users.	Multiple deploys per day	Between once per day and once per week	Between once per week and once per month	Between once per month and once in six months
Lead time for a change. This explains the time from code committed to code successfully running in production.	Less than one day	Between one day and one week	Between one week and one month	Between one month and six months
Meantime to recovery. When a defect that affects the users occur, how long does it take to restore service?	Less than one hour	Less than one hour	Less than one day	Between one week to one month
The change failure rate for the core service or application which is worked on, what percentage of the changes made to release to users, or changes to production.	0 – 15%	0 – 15%	0 – 15%	46-60%

Note. The table shows metrics calculation and their results according to the results obtained from various sources based on state of DevOps 2019. From DORA (DevOps research & assessment). (https://services.google.com/fh/files/misc/state-of-devops-2019.pdf)

4.5. Prescriptive Statistics and Analysis (DevOps Objectives and Key Results)

Prescriptive statistics as a method of statistical analysis, utilizes the past performance and trends to determine what needs to be done to achieve future goals. In this context, this is used in checking on the performance of the current organizations using DevOps and the past state of DevOps and recommend changes for future improvements.

According to various sources, many companies are migrating from the old ways of doing production to DevOps, which are geared towards managing and integrating the development team with the operations team to realize robust products development and deployment (Ivanova & Ivanova, 2018). Prescriptive statistics is a build-up from the descriptive and the predictive statistics. Each day, the IT field evolves with new tools deployed and new methods and technologies positioned. DevOps are also deploying new tools, all of this with the DevOps engineer to manage. Prescriptive statistics help in the perspective analysis, which helps

monitor and maintain DevOps systems. These statistics use simulation and optimization algorithms in advising and concluding on the possible outcome. Therefore, this calls for improvement each day and ensures that the skills set, and tools are updated for better performance. This would otherwise lead to a better performance in terms of metrics in various stages.

Working towards enhancements, objectives, and critical results is one of the pillars required to set up a luxurious start. This can be made a success through a well-formed formula for metrics acquisition and hence drives performance in the department that you are planning to enhance. In DevOps, vital objectives and key results need to be concentrated on while considering various key performance indicators and metrics to attain success. Targets and initiatives are better starting points for organizations that want to be far with DevOps. In this regard, there are considerations of factors such as release management, product development, and DevOps monitoring.

The Figure 6 pie chart shows the outcome of setting up successful DevOps OKRs and attaining the efficiency that one has targeted.

Figure 6

Business Success Due to Well-Implemented Metrics and Well-Set OKR.

Note. The pie chart shows the outcome of setting up successful DevOps OKRs and attaining the efficiency that one has targeted.

4.6. Impacts of DevOps metrics success on business

According to the Puppets' annual survey, which was lastly conducted in 2021, various success factors need to be considered to have functional DevOps in the enterprise. One of the factors which was much articulated and emphasized was having a clear framework on how to execute the DevOps metrics, which are the performance measure elements for DevOps. The survey involved various stakeholders, including the DORA google DevOps engineers Nicole, and it addressed the growth and rate of adoption of DevOps (Kersten N. , 2021).

According to the survey, DevOps adoption is increasing, though not yet rooted. The survey found out that the organization leaders play a significant role in ensuring the success of DevOps. This resulted from conducting interviews and considering various enterprises and developed middle- and low-level organizations. With support and cohesion from leaders, teams set Objections and work towards achieving those objectives and attaining key results. Usually, this is made a success by targeting how well the key metrics are performed. Therefore, sharing common goals in the whole organization ensures support and hence aids the DevOps team in growing and posting improvements on their performance.

From the survey, it was evident that proper adherence and setting of the objectives and concentration on the four primary DevOps metrics led to massive success and a significant return on investment for the organization. It is also evident that many enterprises tried to adopt DevOps but could not implement them fully. Among 100% of organizations that wanted to adopt DevOps, only 20% of them reached the peak and enjoyed maximum benefits; primarily, the big organizations such as Google, as discussed above, are successful in DevOps since they embraced unity and hence able to manoeuvre through the waves of growing from grass to grace. According to them, DevOps metrics and a set of objectives and key results was the leading factor contributing to the success (Puppet, 2020). According to various findings, the Table 15 shows a summary of the percentages of the DevOps metrics implementation according to various sources.

Table 15

DevOps metrics implementation success measure

Metric	Performance		
	Low	Medium	High
Change failure rate	Below 15%	Below 15%	Below 5%
Deployment Frequency	Monthly or less often	From daily to weekly	On-demand, whenever needed
Meantime to recovery	Less than one week	Less than one day	Less than one hour
Lead time for change	Between one week and six months	Less than one week	Less than one hour

Cohesion is a crucial thing when it comes to success. If the organization can afford to have the same vision and goals, then there is no doubt that the DevOps team will also succeed. It is wise to involve the entire organization to implement the objectives and key results and make them aware of the DevOps metrics to be keen with until it becomes a habit. Operations team to be aware of what to avoid and maximize on as well as the developers' team and hence success.

4.7. DevOps Taxonomy

In order to have a successful DevOps team and activities, we must have a clear understanding of the processes involved in the DevOps work chain. We will know which the best approach to use is to realize. In the research process, we examined various articles concerning the taxonomy of DevOps both in tools and in activities. This would give a clear idea of the best approach to have the best results and boost return on investment. The findings were as follows, each with its strengths and weaknesses and therefore setting the basis for a roadmap to ensure that challenges are minimized, and DevOps realize remarkable success. The findings are cantered on realizing the possible ways to improve the change failure rate, mean time to recovery, lead time, and deployment frequency metrics, which are essential for measuring DevOps performance.

4.7.1. DevOps Activities Taxonomy

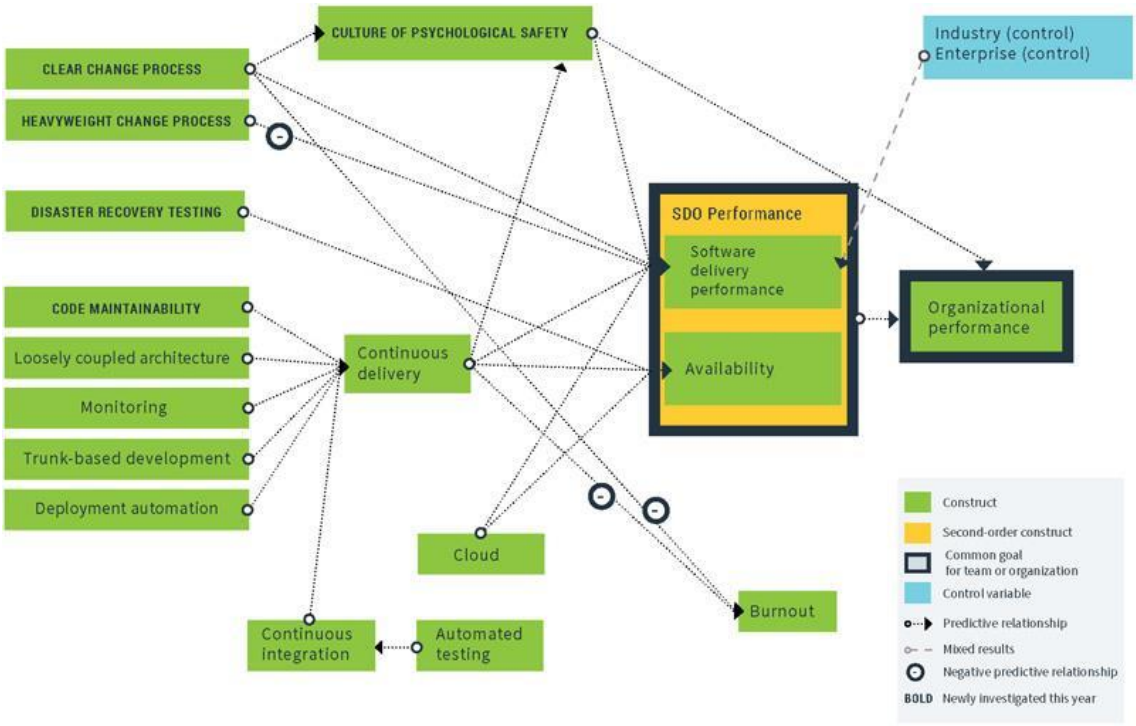
DevOps is a new term in the industry, and therefore, it is underdeveloped and less widespread. This section discusses the DevOps taxonomy finding, considering the developer's interaction with the organization's operations, DevOps teams, outsourced operations, and the DevOps bridging teams. Upon examination and following the case study performed by the University of Salford Manchester and examination of various enterprises on the DevOps (Macarthy & Bass, 2020), there was the identification of three distinct groups of activities which include: provisioning and maintenance of physical systems, function virtualization and creation of

automated pipelines, and development, deployment, and maintenance of applications. This has further led to the development of DevOps as a bridge system.

Given the three categories of DevOps, we have various activities which each group performs. Firstly, the Developers-Operations mode depicts a situation where the senior developers manage infrastructure automation in the hybrid cloud environment and other enterprise activities (Katal et al., 2019). The IT operations team maintains the physical and the deployed applications while the developers code and deploy their codes to the cloud. The senior developers act as the facilitators in the DevOps practices. The Figure 78 depicts the activities of DevOps and the connection between the developers and the operations team.

Figure 7

Improving DevOps Bridging Process.

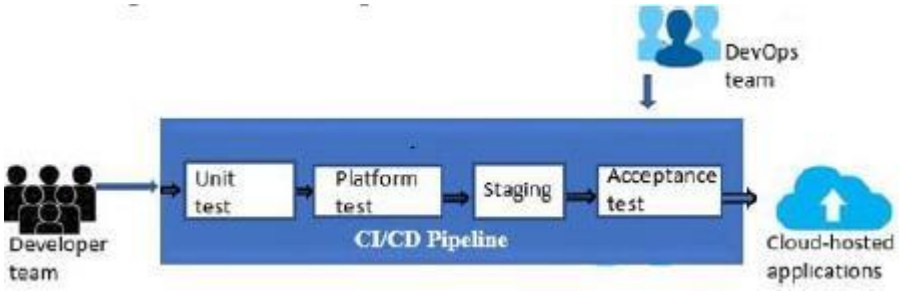


Note. The diagram demonstrates the important finding in DevOps. It shows the DevOps bridge process between each section. From "State of DevOps Report 2019: DevOps Not a Trend, The Standard" by Veritis, 2019. https://www.veritis.com/blog/state-of-devops-report-2019-devops-not-a-trend-the-standard

The second DevOps approach mode is the developer-developer operations mode. The deployment pipelines and the cloud infrastructures are created, managed, and deployed by the operations team. However, the developers' applications are created, deployed, and maintained by the development team. In this case, the developers are concerned with adding value to the business. According to various research, this is the most commonly used mode among the other modes. Instead of development, management, and deployment, the developers' team is further concerned with facilitating the software's working. In this case, the developer's team merges with the IT operations team to carry out continuous delivery and integration, configuration management, deployment, automated testing, metrics collection, and monitoring. In this mode, everyone is responsible for their whereabouts, and therefore developers' teams act as service providers to the IT operations team. This approach revealed the three major activities discussed earlier in this section: provisioning and maintenance of physical systems, function virtualization and creation of automated pipelines, and development, deployment, and maintenance of applications. The lead time is reduced while the deployment frequency increases, hence the best approach for DevOps.

Figure 8

A Pictorial Representation of The Developer-Dev Operations DevOps.



Note. A pictorial representation of the developer-dev operations DevOps. From "An Empirical Taxonomy of DevOps in Practice" by R. Macarthy, 2020, 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020.

The other approach is the developer-outsourced operations mode. The difference between this mode and the latter is that the software deployment is done on the cloud, and therefore there is no need for operations experts. In this case, the senior software developers are tasked with writing infrastructural codes and creating and managing deployment pipelines. The following diagram is a visual representation of this DevOps approach.

4.7.2.DevOps Tools Taxonomy

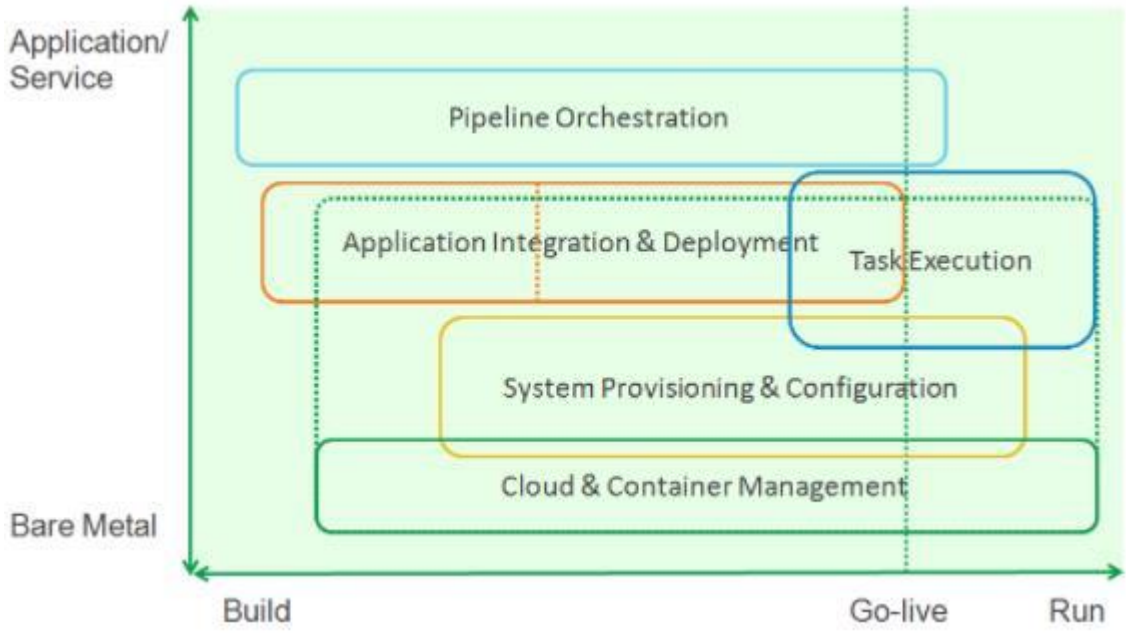
Tools play a crucial role in ensuring that DevOps teams achieve their objectives, and the four key performance measures are in order. Therefore, one must have a deep knowledge of the various tools used and which sections are used in the DevOps environment for exemplary performance (Rafi et al., 2020).

Moreover, one should have a deeper understanding of tips to consider when selecting a particular stage and activity tool. Following this, the research conducted on the best approach to deciding and using these tools showed various considerations to be taken into control while identifying some of the tools. This involved considering the working environment and secondary sources on various articles and DevOps tools reviews.

There were various tools to be used from the research and consider the tools in each category. The categories were as follows: cloud and container management, task execution category, system provisioning and configuration, application integration, application deployment, and pipeline orchestration. The whole process and categories of the tools can be represented visually as follows (Figure 9):

Figure 9

A Pictorial Representation of Various Levels of DevOps System Functioning



Note. The figure represents the various tools to be used from the research and consider the tools in each category. The categories were as follows: cloud and container management, task execution category, system provisioning and configuration, application integration, application deployment, and pipeline orchestration. From "A "taxonomy" for DevOps tools" by Digital.Ai, 2014. https://digital.ai/catalyst-blog/a-taxonomy-for-devops-tools

Examining each category, the research looked at the consideration and tools used in each level. Task execution – these tools allow you to run scripts on one or multiple execution environments or systems. These functions depend on the tool in question. They are best on regular maintenance and management of the DevOps. They include task execution frameworks since they need t to be run on multiple servers.

They are not good at coordinating tools across multiple machines, targeting hybrid systems such as Windows operating systems, and proving the box script. The next category of tools in the cloud and container management tools. These are tools used for cloud management – they aid in managing load balancing, routing, storage, networks, and virtual machines (Khan & Shameem, 2020). They are also used in container management – these are the ones that assist in managing the runtime environments and runtime containers that might be in virtual matches networks, among others. They are best at providing the on-demand environment and resources for runtime environments and delivery pipelines. They are also termed environments or container definition models, instantiated in a developer's computer or a production environment. They are advantageous in providing lightweight containers, scaling, and load balancing. These portable versions are too restrictive while the flexible ones are not

portable enough hence being a hindrance. Also, they are not good at external linkages and the management of persistence data. They also have limited or no support at all.

System configuration and provisioning tools- these are tools that aid in system state definition, which is mainly done in a declarative manner, bringing the target system to the desired state and making sure that it remains there, and also allowing the execution of the ad-hoc server commands. They are best at ensuring that most machines are in known states and remain there, provision out-of-box service installation tasks and system configuration contents, and reporting on the intended and the actual system state depending on the tool. However, these tools are poor at coordination and action sequencing across multiple machines. Application integration tools are version control tools that take the source code and other development artifacts into a deliverable that can be released run code integration and validation tests. They are good at maintaining the releasability of the main branch. They are poor at coding action sequences for multiple machines, so they are not recommended.

Category	**Tools**
Pipeline Orchestration tools	GitHub, CicleCI, Marathon (Matskin et al., 2021)
Application Integration and Deployment tools	Capistrano, Shippable, Juju, Deploybot, Travis CI, AWS CodeDeploy, GoCD, IBM UrbanCode Deploy, Octopus Deploy, Jenkins (Zimmermann, 2017)
Task Execution tools	Cron++, JenKins (Ohtsuki et al., 2016)
System Provisioning and Configuration tools	Puppet, Ansible, Chef, Saltstack, CFEngine (Ebert et al., 2016),
Cloud and Container Management tools	Docker, Kubernetes, Containership, AWS, Microsoft Cloud, GitLab, SUSE Rancher, IBM Turbonomic (Punjabi & Bajaj, 2016)

4.8. DevOps Maturity Models

Organizations have various ways of measuring their performance and growth level. This, in turn, helps find ways of increasing performance over time and hence achieving more critical goals. However, without the right culture, architecture, and best practices, the organization cannot feel the full effects of DevOps in their environments (Zarour et al., 2019). Taking a few case studies, we have the Mohamed DevOps maturity model. HP enterprise maturity model, Eficode maturity model, and IBM DevOps maturity model.

The IBM DevOps maturity model is based on continuous software delivery. It enables enterprises to seize the market and get quick customer feedback. The model is based on uniting people, practices, technology, and information, which play a crucial role in the

software delivery pipeline. The model outlines the four paths towards successful continuous delivery: planning and measuring, developing and testing, releasing and deploying, monitoring, and optimizing.

The maturity model created by Hewlett Packard Enterprise – commonly known as HP, is based on planning for big organizations based on the Capability Maturity Model Integration (CMMI) maturity model. This is done by spanning the complete lifecycle of an application and its development. It accesses the aspects of process automation and cooperation among the teams. Despite the usage of DevOps, the issue of late software submission has not fully been solved. However, there is a need for a quality software deliverable. This model deals with structure and mindset, contexts, and distribution, build and iterative, quality assurance, visibility, and reporting. Hence accommodating both beginners, intermediate and advanced DevOps teams.

The Mohammed DevOps maturity model is based on the CMMI maturity model tool for measuring performance. Mohamed suggests that implementing this model helps improve operations efficiency, increases visibility, and mitigates some common risks such as downtime during adoption (Mohamed, 2016). It presents a comprehensive transition structure and determines DevOps capabilities at each level.

Maturity Model	Explanation
The IBM DevOps maturity model	based on continuous software delivery
The maturity model created by Hewlett Packard Enterprise	based on planning for big organizations based on the Capability Maturity Model Integration (CMMI) maturity model
The Mohammed DevOps maturity model	CMMI maturity model tool for measuring performance
Efficode Maturity Model	Based on four levels with final level being continuous improvement attain efficiency and quality.
Feijter Maturity Model	offers strategic priorities that allow implementation and execution organizations to evolve finely

4.9. DevOps Metrics Monitoring

This oversees the whole development process ranging from planning, development, integration and testing, deployment, and operations (Arundel & Domingus, 2019). This involves complete view and real-time streaming, visualization, and historical replay, the most crucial components in application performance monitoring. DevOps monitoring is critical in

enabling the teams to respond to any poor customer experience quickly and automatically. In general, it helps the team shift back to the earlier stages of production and mend the broken production changes. This includes functionalities such as the ability for software to respond to common errors either through a call to action or automatically in any case possible.

The cost of DevOps monitoring is not fixed. However, it ranges depending on the amount of data ingested, stored, and processed. It also fluctuates depending on the tools being used in the monitoring process. The tools range from cloud automation tools to orchestration tools, vital in the DevOps development pipeline. In this case, we take a case study of AWS DevOps services monitoring which is the organization responsible for Amazon cloud services. In this case study, the researcher looked at the amount of data scanned by the Amazon Athena queries and the number of Amazon QuickSight Readers and authors considering their access time to dashboards. The Table 16 shows the cost of running the service in the United States of North Virginia with an approximate value of $34.20, putting into consideration that CI/CD teams generate approximately 100GB of data monthly in code changes code deployments, and there is one author and ten QuickSight readers.

Table 16

AWS DevOps Metrics Monitoring (Amazon, 2021)

AWS Service	Dimensions/Month	Cost/month
Amazon Athena	100 queries, 10GB data scanned per query	~$5.00
Amazon Kinesis Data Firehose	100GB	~$2.90
Amazon Simple Storage Service (Amazon S3)	100GB	~$2.30
Amazon QuickSight	One author, ten readers, access two times per month for each reader	~$24.00
Total		~$34.20

Note. The table shows the details of the cost of running a solution in a certain region (US East) and it excludes the free tier. From Amazon 2022. (https://docs.aws.amazon.com/solutions/latest/devops-monitoring-dashboard-on-aws/cost.html)

In order to complement the process of successful DevOps monitoring, there are healthy monitoring practices that include using advanced monitoring tools which are compatible with DevOps culture. Therefore, attention is needed in order to identify and implement these monitoring tools in addition to the well-known and used developer tools such as Debuggers, IDEs, defect tracking, code repositories, integration, and deployment tools. A single tool can provide an overall view of production and staging services, applications, and dependencies. This helps the team provision, ingest, tagging, view, and analyse the health of complex

distributed environments. An example of such tools is Atlassian's internal PaaS tool Micros which is used as a single pane of glass for providing information about services in a concise and discoverable manner. Application performance monitoring tools such as NewRelic and SignalFx monitor application-specific performance indicators such as load time, latencies, and transitions in the systems such as CPU and memory utilization. They also Implement different types of monitors such as security in development, capacity, infrastructure, alarms, heartbeats, synthetic, transactions, and errors. These monitors are application-specific, and therefore, they are implemented based on the application's specifications. An alert and incident management system sends critical crash reporting and logs management alerts. Such monitors fit one's development and operations environment and give alerts with low latency. An example of such a tool includes Opsgenie, a product of Google.

DevOps monitoring is also strategical, and its implementation requires attention to the core practices and a set of tools. The key DevOps capabilities to consider include shift left testing and alert and incidence monitoring. Quality, test cycles, and error reduction are vital in shift-left testing. It is comprehensive to extend shift testing practices and ensures that monitoring is implemented early to maintain continuity through production and preserve the quality of monitoring alerts (Airaj, 2017).

Incidences to be managed include network and hardware failures, resource exhaustion, misconfiguration, software bugs, and data inconsistencies. Some of the best practices for responding to incidents include building a culture of collaboration, planning, and building monitors to ensure that dependent services operate as expected.

DevOps monitoring is extended to staging, testing, and development environments due to frequent code changes, demand visibility, automated collaboration, experimentation, change management, and dependent system monitoring. These are the main points that describe why we need DevOps monitoring in our DevOps teams, and they are elaborated as follows:

- Frequent code changes, driven by continuous integration and deployment, have increased the change rate and made the production environment more complex. With the incubation of the micro frontend and microservices to cloud-native, there are considerable workloads in production, each with different operational requirements security, scale, redundancy, and latency. Therefore, this has called for more visibility and the need to detect and respond to poor customer experience in a time-bound manner.

- DevOps ensures increased collaboration between the development, operations, and business functions teams. This can be hindered by inadequate or no integration between tools, leading to less collaboration between teams. This can be solved through automation through practices such as getting a complete view of the development pipeline inside the editor and setting automation rules which listen to pull or commit requests and then update the status of the issues and send messages to the team's slack channel.
- Experimentation is there to optimize products to respond to customer needs (Riungu-Kalliosaari et al., 2016). The production environment may run numerous feature flags and experiments, making it difficult for monitoring systems to report the cause for degraded experiences. Therefore, development teams need to make sure that they define the service level objectives and service level indicators that are monitored and acted upon.
- Change management is vital since most production outages are caused by changes, especially for mission-critical applications for financial and health care sectors. Therefore, approval flows need to be automated, and risks associated with changes must be determined based on the risk of change. This requires a deeper understanding and monitoring strategy alongside flexible, rich, and advanced monitoring tools critical to the development environment.
- Alongside monitoring the developed systems, the teams should also consider the availability and performance of dependent systems. There is a need to ensure that the application built, and the service chosen are compatible. There is also a need for instrumentation and strategies for tracing errors independent systems, mainly distributed systems, and handling failures when they occur.

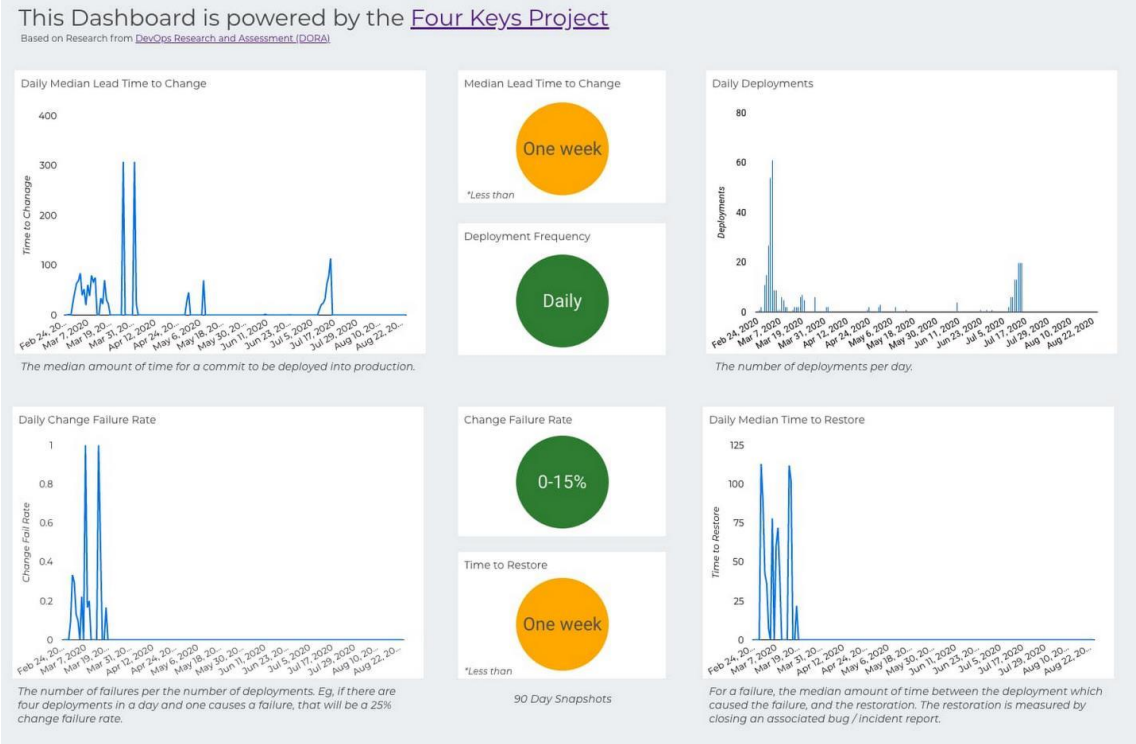
4.10. Critical Analysis of the Results

A few organizations have already adopted DevOps as part of their daily product development and deployment practice. These organizations had to practice and develop the skill set required to be outstanding in DevOps. They had common goals and shared the same organizational vision, both the DevOps team and the management team. Upon studying their working mode, success, and failure, the research recommends improving DevOps metrics implementation and ensuring that are realized.

From the research and findings, it is evident that the DevOps metrics should align with the objectives and key results to realize massive success. On the side of metrics, two vital metrics

need to be prioritized, including lead time for change and the deployment frequency. The lower the lead time for a change, the more efficient the DevOps, and the higher the deployment frequency, the more efficient the DevOps team. This can be evidenced in the following diagram from the DORA Google dashboard on the DevOps performance (Figure 10).

Figure 10
DORA Metrics Effectiveness Measure



Note. This figure shows the dashboard that it is demonstrated to provide a high-level categorization based on the DORA research for the four key metrics. From "Accelerate State of DevOps Report" by D. Graves, 2020. (https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance)

In order to maximize these metrics, various factors need to be considered. A happy man is always said to be a productive man, and therefore, motivation is needed. Cohesion is the critical thing towards success. Therefore, DevOps success requires the management, operations, and development teams. Also, the required personnel and resources should be employed for success. Bot the skills and tools are required. All the teams should share the same vision and strive to attain the same objectives. Finally, the two vital DevOps metrics will be in control hence success.

4.10.1. General Findings Summary

Figure 11
DevOps Metrics Summary (Part 1)

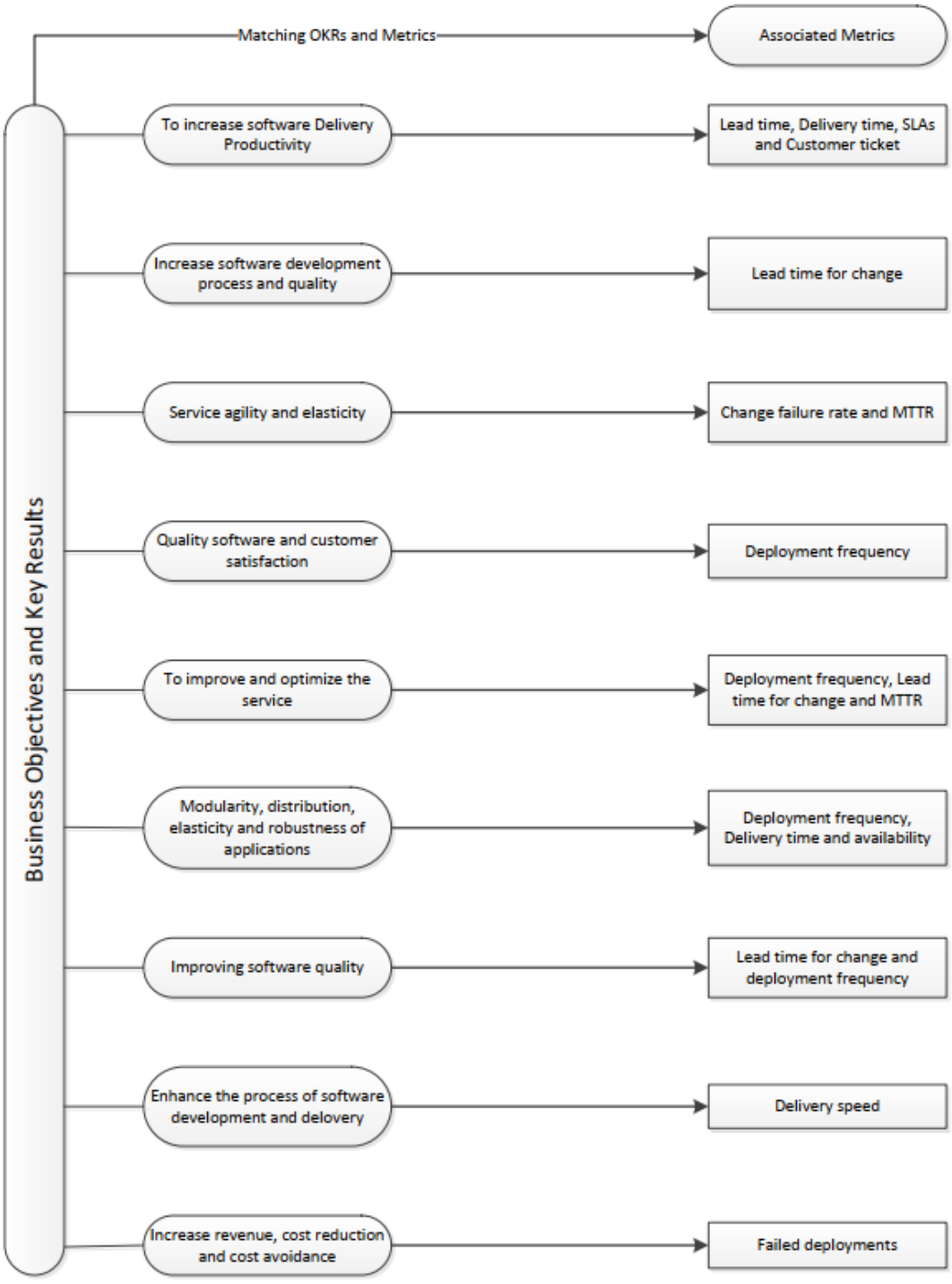


Figure 12
DevOps Metrics Summary (Part 2)

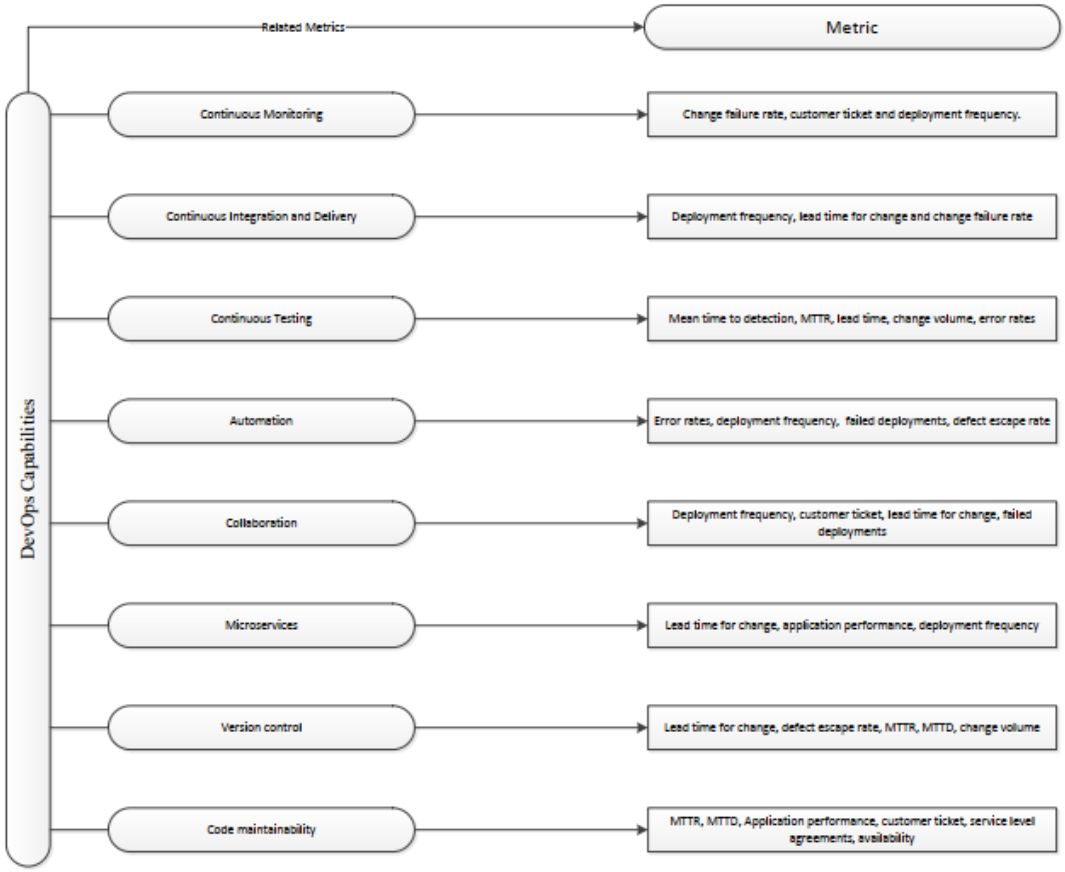


Figure 11 and 12 contains a graphical thematic analysis of various DevOps metrics, their measurement, monitoring, advantages, and disadvantages of each metric. Additionally, the diagram contains the professionals concerned with each stage alongside their roles and responsibilities. The figures are based on the tabular on analysis table 19 available at appendix. By understanding these ideas, one can conceptualize the ideas in the thesis and the argument presented. Therefore, it helps in grasping what is discussed and what is recommended for improving the state of DevOps. From the thematic analysis, there was capturing of the findings related to DevOps metrics and their surrounding aspects. I particularly analysed various sources and grey literature to come up with related aspects of each metric such as their monitoring style, measurement units and style, advantages, and disadvantages. I recorded this from various studies under themes and sub themes and hence came up with unified diagram that represents all of them.

The Figure 13 summarizes the key ideas and suggestions obtained in the due efforts to achieve the study aims and answer the research questions. The diagram contains the introduction

summary, literature review, and the overall thesis structure. The diagram helps the reader understand the process and the main finding discussed in the thesis. The diagram also highlights the thesis suggestions for improving DevOps. This is achieved by understanding DevOps metrics, business objectives, and critical results.

Accordingly, the literature review focuses on the DevOps in general business goals while considering areas such as maturity models in DevOps, an initial look at the DevOps metrics and objectives, and critical results goals which are different for different teams. These are used to suggest improvements and know what changes would possibly improve the metrics. There is also a description of the DevOps metrics measuring and monitoring processes alongside the measurements and monitoring costs. Moreover, there are highlights of various business objectives and critical results, including more productive teams, response to customer needs, quick fix of bugs, and the competitive market advantage.

Finally, an overview of the introduction chapter sets out the roadmap for the whole thesis. The diagram also outlines the DevOps capabilities, processes, tools, and the DevOps pipeline.

Overall, figure 13 and 14 below shows the graphical representation of the main ideas of the thesis. It is a summary of ideas which together attempts to answer the research question and therefore help in achieving the set objectives and finally the aims and goals. Figure 13 talk about metrics and their surrounding environment while figure 14 shows the summary of the relationship that exists between the business OKRs and the team's capabilities

Figure 13
Thesis Structure Summary (a)

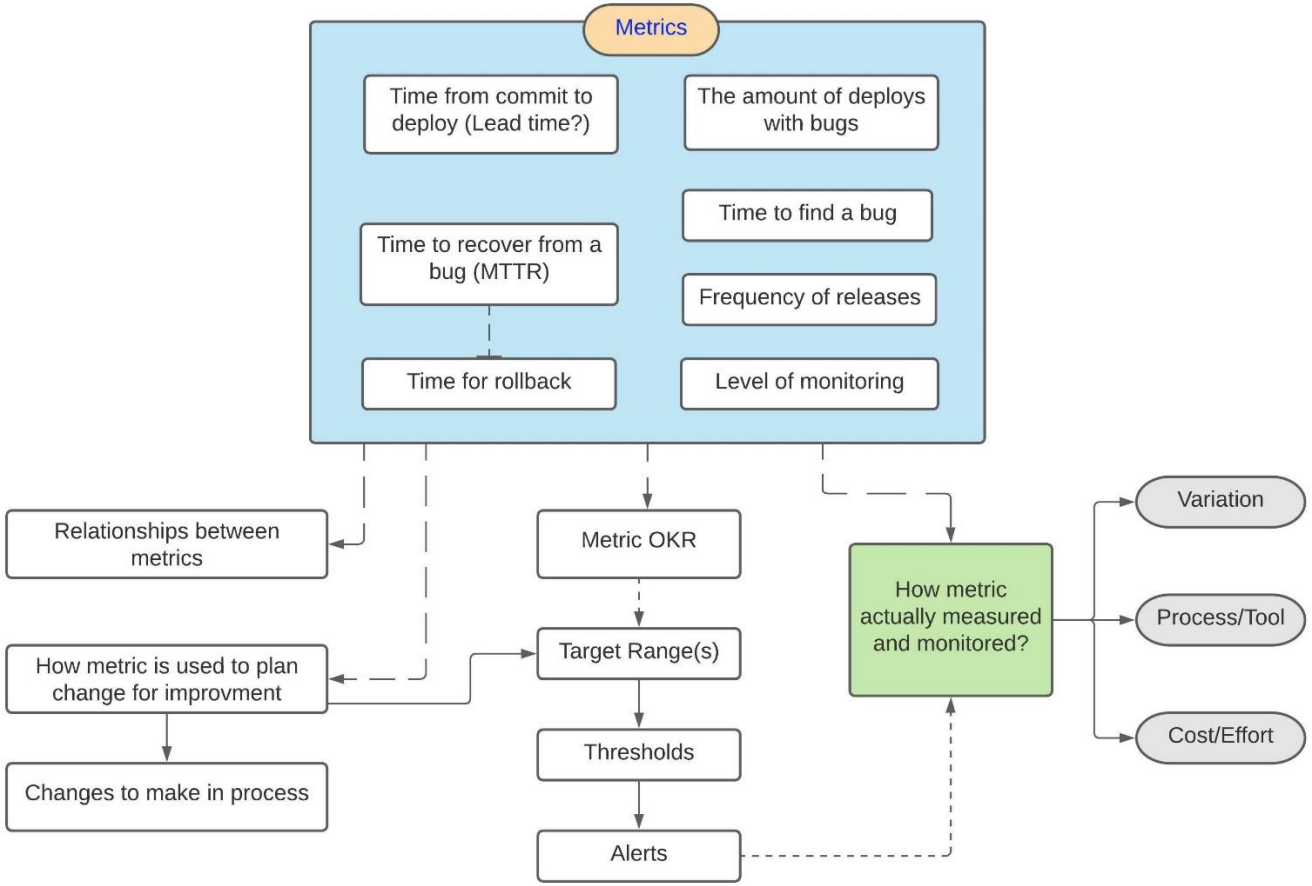
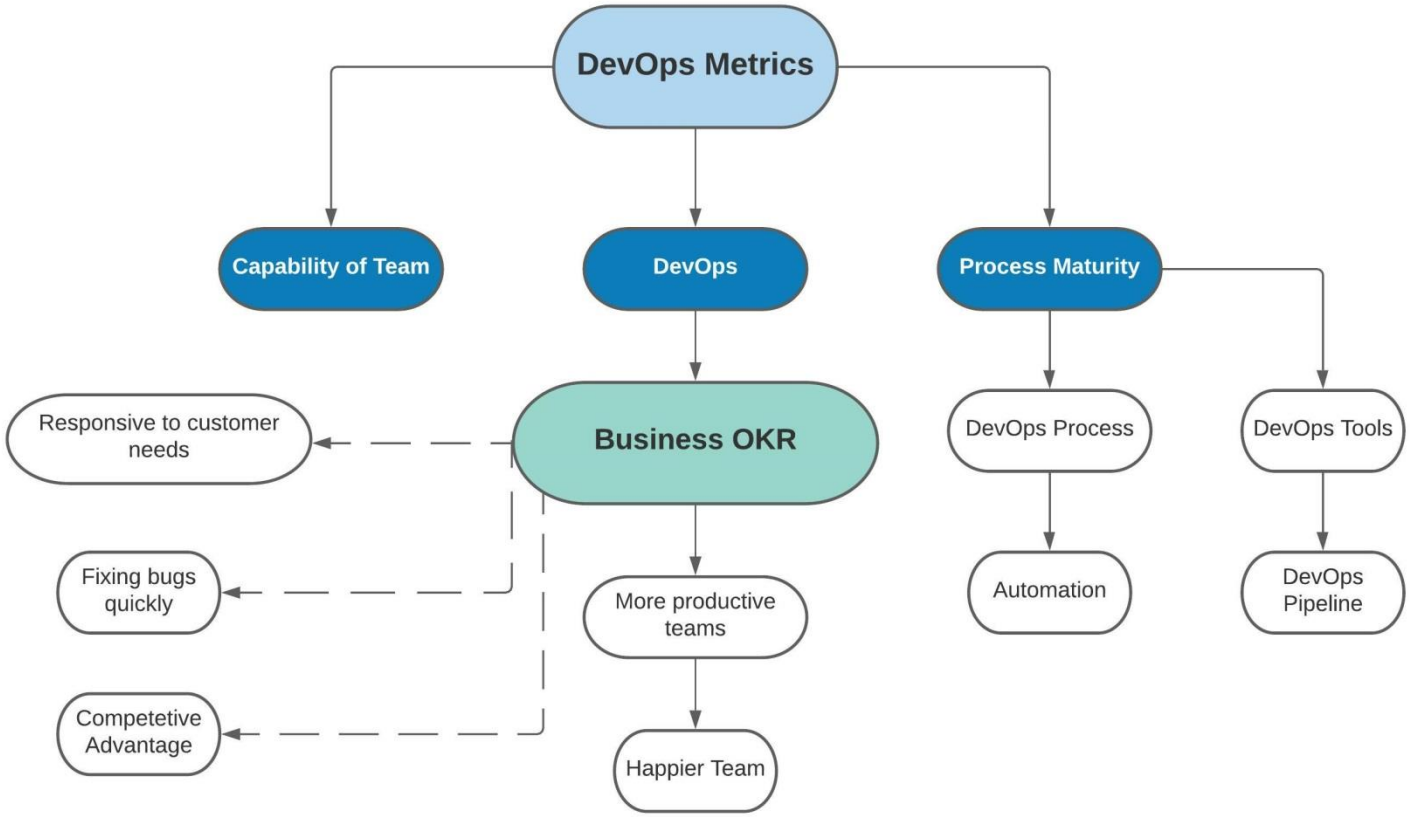


Figure 14
Thesis structure summary (b)



4.10 Summary of DevOps Challenges, Their Solutions and Mitigation Strategies

Table 17

DevOps Challenges, Solutions, And Mitigation Strategies Summary

Challenge	Solution	Mitigation Strategy
Automation of repetitive tasks. Automation is a critical issue that needs various considerations before being activated to ensure that the expenses are minimal. (Pandya, 2021)	Non-usable codes or processes should not be automated.	Automation should be when new processes and practices have been introduced, and all bottlenecks have been removed.
Incentive driven approach. Ensuring that the DevOps approach is business and product-oriented is not easy for DevOps. (Pandya, 2021)	Developers should be incentivized to increase speed to market and operations to ensure reliability, security, governance, and availability.	The teams should be marching towards the same goal.
Operability with DevOps. Creating an enabling working experience needs a lot of conviction and encouragement and, therefore, cranky. (Pandya, 2021)	Enterprises should assess their operational tools and processes and modernize software delivery structures to increase agility and transparency.	Processes such as incident management, request management, and problem management should be improvised and revised upon change.
We are improving Change management. Handling the changes due to service outrages is an issue that needs particular attention for quality changes. (Pandya, 2021)	Enterprises with legacy processes and modernize their processes and be more agile.	Using tools such as Codemagic and other third-party software can help in migration.
Environmental Challenges. DevOps involves changing from traditional infrastructure to microservices, making it complex for the developers and operations professionals who are so much used to the traditional infrastructure. (Dumitrescu, 2021)	Creation of infrastructural blueprints and implementation of continuous delivery to ensure that all environments are identical	Teams should agree on standard blueprints for the execution of DevOps processes and also implement continuous delivery.
Maturity in DevOps and SDLC. Adopting the necessary training and tools required for DevOps is also a challenge for many start-ups. (Pandya, 2021)	Enterprises should help teams adopt DevOps tools and technologies and invest in training them.	Solicitation and improvement of feedback by the teams.
Manual testing and deployment. Manual testing and deployment make the who process slow and tedious and therefore need to be addressed to ensure motivation for maximum production. (Eeles, 2019)	Automation of frameworks and deplo6ment processes. This can be done through software and cloud services.	Enterprises should consider the implementation of test procedures in their development processes.

5. Chapter 5 - Discussion

5.1. Introduction

In this chapter, the thesis presents an in-depth discussion of the results obtained during the research period and various conclusions made in the findings chapter. The discussion backed up with the findings suggests the answers to the main research question on how the DevOps team's capability can be evaluated to recommend the possible steps a DevOps team should take to realize its capabilities. This will be presented systematically while emphasizing the DevOps team's key metrics to consider, their objectives, and the critical outcomes aimed at realizing their full potential. By evaluating these metrics, objectives, and key results, the team will know which step to take next and how to carry out their activities orderly and in the best approach possible. This will also enable the team to know objectives, key results to change, and the metrics to eye on to increase their productivity (Jabbari et al., 2016). The discussion also recommends the ways to increase productivity and the metrics to emphasize. By realizing the metrics to emphasize, the DevOps team will make some reasonable judgments to improve their productivity. Finally, the discussion will present the achievements and challenges which are experienced or might be met alongside the journey of DevOps team realization. The discussion also presents possible ways to mitigate the challenges along the journey and maximize the achievements. In general, the discussion will cover the DevOps metrics emphasizing the four primary metrics every DevOps team should concentrate on while paying critical attention to the two most crucial DevOps teams, which are the backbones for change. The discussion will also showcase suggestions on accelerating DevOps metrics, switching objectives and key results, and the steps towards realizing a more efficient DevOps team.

5.2. DevOps Metrics

The old saying that you cannot improve what you do not measure also applies in DevOps. In order to fulfil the promise of the DevOps teams of faster development and deployment to the market, the teams need to collect, analyse and measure various metrics. DevOps metrics are data points that reveal the performance of the DevOps software development pipeline and aid in identifying and getting rid of any bottleneck in the process (Hall). DevOps try to bridge the developers and the operations teams the Dev team is mainly composed of developers, and the operations team is composed of persons such as system administrators. Therefore, DevOps metrics provide the data needed by the DevOps teams to have visibility and control over their development pipeline. DevOps metrics can track both technical capabilities and team

processes in software development. DevOps metrics allow the DevOps team to measure and assess their workflows and track their progress towards achieving their goals; improved application performance, faster release cycles, and increased quality (Sallin et al., 2021). Though numerous metrics help realize the return on investment for an enterprise, the focus will be on the four main metrics: mean Lead time for changes, deployment frequency, mean time to recovery, and change failure rate. These are broadly categorized into two, the ones used to measure speed and the ones used to measure the DevOps team's stability. Depending on the performance of the companies, they are either categorized as low performers, medium performers, high performers, and elite performers based on the DevOps metrics discussed below (Angara et al., 2018).

As indicated above, there is overreliance on the metrics and suggestions that they are the ones used in measuring the effectiveness of DevOps. This thesis however goes a step deeper into analysing the categories of these major DevOps metrics and giving advice and on how they are supposed to be monitored and measured. Also, the thesis in this section how the individual metrics can foster changes in different sections of the organization. It compares and contrasts the existing information on DevOps metrics individually while stating their contribution in classifying and determining the level of growth of a DevOps team in an organization.

5.2.1.Lead Time for Changes

Mean lead time for change can be described as the time it takes for the code to be committed and the code to be running in the production environment. This depends on the company's capabilities and the team's skill set. It can take one day for elite performers, and for medium and low performers, it takes from one to six months. Speed is one of the vital things that are critical when it comes to developing and deploying a new product to the market. Today, the business competition is based on your IT systems. Therefore, there are demands by the business for faster release and deployment of working software as soonest possible when the need arises for a particular organization. In DevOps, lead time for change is critical for measuring performance. The research is trying to answer the question, how can lead time for change be improved in DevOps? From the findings, this can be done by automating the deployment pipeline, releasing the software to the market more often, embracing more minor releases, focusing on the lead time, and ensuring the quality of each release.

Automation is one of the critical changes that have been introduced in the IT industry and development environment. According to research, a more significant portion of software failures and delays are caused by human error. Therefore, human error will be minimized with

the introduction of automation, translating to better production. The existence and realization of cloud services have made automation much possible for various processes. Some activities which can be automated to reduce the lead time for change include testing, error detection, and correction, among others.

5.2.2.Deployment Frequency

This is the frequency of code deployment, and it includes the bug fixes alongside tracking the frequency in which code increments on staging, testing, and production are done. This is one of the critical measures of success in a development environment. It deals with how long it takes to correct bugs and errors in code and then execute that code. Improved deployment frequency means an increase in the effectiveness of the DevOps groups in a production environment. Therefore, this can be increased through continued efforts to remove and avoid bugs, hiring competent developers and operations engineers, and having the right tools for production.

Resources are sure of making and guaranteeing performance in a DevOps start-up, especially the workforce. There should be a massive investment in the skill set for an enterprise to post maximum contentment when achieving the set goals and key objectives. This helps ensure that the errors are minimal and that there is quality work within the shortest period possible, through investing in the right tools and skill set, the common objectives, and critical results for DevOps, which are product development, continuous delivery, continuous testing, release management, and DevOps monitoring.

5.2.3.Mean Time to Recovery

The DevOps team uses this to measure the time used for the developers to recover from a production failure. It is one of the DevOps metrics used to measure the stability of the DevOps team and therefore contributes to the team's effectiveness. This answers how long it takes to recover from a failure in a production environment. This can be attributed to its counterpart mean time to failure, which measures how often the system fails in a production environment. If recovery is as low as possible, the team is effective. This can continually be improved as the team grows from low to elite performers. This can be improved through avoiding unnecessary system changes which would otherwise conflict with the DevOps slogan of frequent releases and, therefore, a better solution of changing this. Therefore, this can be improved by focusing on the recovery options and algorithms since failure will always happen. This can be better by having automated testing, test-driven developments, and coding the fix along with tests.

One of the revolutionary aspects of DevOps is the automation of integration and testing. In order to improve this sector, the testing and integration tools are supposed to be provisioned with error detection and correction mechanisms and algorithms to make sure that there is less time to recover. This is in contrast to its counterpart mean time to failure, which focuses on how frequently an error or failure occurs. For example, if a bank system fails severally in a day but takes a few seconds to detect and correct the error, the customers might not notice and therefore will not be furious. However, if the same ban system fails once in 5 years but takes the whole day to correct the software, there will be inconsistency, and customers will be amused.

5.2.4.Change Failure Rate

These percentage changes result from poor service failures and need to be fixed. This DevOps metric measures the efficiency of the deployment process, and it is presented in percentages. Elite performers have a change failure rate of about 0 – 15%, while the low performers DevOps have a change failure rate of about 40 – 65% in their deployment process. This metric can be hard to measure and quantify since failure can result anywhere. This change failure rate can be reduced by ensuring that the development process is reliable, consistent, and automated.

Human error will always be in existence, and therefore the efforts should be cantered on how to minimize the error and not eliminate it. In this sense, integration testing automation helps minimize the change failure rate among DevOps, as seen in the findings chapter.

5.3. DevOps Metrics Report Acceleration and Measuring

5.3.1.Acceleration

DevOps state acceleration and improvement require the improvement and considering the improvement of the four key DevOps metrics. The key metrics are the deployment frequency and the mean lead time for a change (Debbiche et al., 2019). When the stability and the frequency of the deployment of a DevOps team are accelerated, the overall state of the DevOps report is accelerated. Therefore, acceleration requires employing mechanisms to improve the four key DevOps metrics.

5.3.2.Measuring

DevOps transformation indicates a significant investment of money, time, and resources to an organization. Consequently, the ability to measure progress clearly and accurately enables you to define objectives, improve efficiency and track success to ensure maximum productivity. DevOps KPIs provide complete detail of the impacts and business value of the DevOps. Well-

defined goals are a critical determinant in establishing key metrics (Elliot, 2015). The management is supposed to decide on which business outcomes will have the greatest return on investments to the organization, while the DevOps team has to decide on which KPI will help them in business-driven processes and initiatives. The Table 18 shows the critical DevOps metrics and how they should be measured in a business environment. How often and in which units should the DevOps metrics be measured for effective business success. Through flowing the frequency and the units of measurement, the DevOps team will have met the business objectives and key results, which are mainly faster and quality delivery to the market.

Table 18

DevOps Metrics Measurement

DevOps metric	Measurement
Deployment frequency	Daily or weekly based on the demand of the product under consideration. Measures the number of deployments done in a certain period. On-demand requires multiple times per day.
Change failure rate	They are graded from 100%. This is the percentage ratio of failed deploys to the number of deploys done.
Meantime to recovery	This is the time it takes for a service outrange to be corrected without affecting the delivery of the services. Monitoring is based on customer satisfaction with the service provision.
Lead time for change	Daily or weekly, or monthly. This is when a code functionality is developed and enters a production environment. It can be measured and monitored using software such as an issue tracker integrated into the development pipeline or manually setting the clock.
Change Volume	Small or medium or large. This is the number of software changes that are required as a result of the outside world feedback.
Level of monitoring	High or medium or low.

Measurement and monitoring of the DevOps metrics are entrusted to various persons, both the operations team and the developer's team. These are the ones who report to the management on how the DevOps teams are performing and the changes required in order to align to the business goals. Through continuous monitoring and improvements, the DevOps team attain peak success which enables the business in question to improve their return on investment and appreciate the existence of the DevOps team and their activities. Among the experts entrusted with monitoring and measuring the DevOps, the performance includes the system admins in the operations team, DevOps engineers, and the Site reliability engineers. Site reliability engineers monitor the uptime of the systems while the DevOps manage the entire DevOps team.

5.4. Cost of DevOps metrics monitoring

Every improvement and good thing always come with its price. Therefore, DevOps improvements are also not an exception. This section discusses the cost of improving and

maintaining the four key DevOps metrics. In monitoring and improving the state of the DevOps report, there are specific tools and skillsets required. Therefore, this cost should be catered for and considered as an investment. According to be improvements of some of the DevOps metrics, there are the requirements of automation and update and investment of the best resources when it comes to the human labour and the tools. Therefore, to consider the improvement, the team should consider and decide whether the investment is viable. Through this, the investment requirements are calculated, and the money invested would otherwise yield profit for the company. In this line, the organization sets the business needs and goals while the DevOps team decides which KPIs will be used to attain given business goals and therefore work towards achieving them. By achieving the given goals, the state of DevOps is improved, and the return on investment is realized.

Highlighting some of the requirements, there are cloud services that need to be done in subscription terms, and hence their cost should be calculated and included before taking a step further into using the services. Employment of expert software integration and testing tools such as protractor required a well outlined and developed financial plan. Therefore, in pursuit of improving the state of the DevOps report, the cost should be highly considered. Through the cooperation of the management and the DevOps team, the cost is subsidized, and the return on investment is realized.

5.5. Systematic Mapping of DevOps Tools

DevOps tools are critical requirements in ensuring stability within a DevOps team. They also help determine how well the team will perform and its success rate. Various tools in the market are used during the development and deployment process. Therefore, the concerned team should be conversant about which tools to use to accelerate their activities and make them more favourable (Bass, 2017). In the finding chapter, it explored various categories of tools, the level at which they are used in the development pipeline, and their suitability for what they do. It is right to say that it is not easy to get a tool that completely suits your operations, and therefore, in conjunction with various tools, each competent in a particular area realizes the best results. DevOps' main aspects and focus are frequency of deployment, quality, and speed; there is a need to maintain a consistent software delivery which would otherwise be made possible by correct mapping of DevOps tools. These tools are classified into task execution tools, cloud and container management tools, system provisioning and configuration tools, application integration tools, application deployment tools, and pipeline orchestration tools.

When improving or starting a DevOps project or having a start-up, an enterprise is supposed to invest the right tools to make sure that they realize the best results. There are recommended tools in each stage of the DevOps pipeline and the correct approach to development. By matching the tools alongside the skill sets, the state of the DevOps report will be drastically improved.

5.6. DevOps Monitoring Practices, Capabilities, Challenges, and Mitigation Procedures

Various practices are involved and adopted by DevOps. There are also various potentials and capabilities of the DevOps team, which help get the best out of their activities and improve the four metrics (Senapathi et al., 2018). However, where there are advantages, also disadvantages exist. There are various shortcomings attributed to the existence and activities of DevOps. As stated in the finding chapter, these shortcomings include the loss of employment to some IT experts due to the automation of various resources. The DevOps Capabilities, challenges, and Practices are discussed as follows:

5.6.1. DevOps Capabilities

This subjection compares and contrasts the findings and the existing literature in terms of DevOps capabilities and their interconnection with the DevOps metrics and business OKRs. DevOps consists of both the developers the operations engineers. These are then managed by the DevOps engineers who are trusted with various roles such as pipeline monitoring. The operations engineers are mainly tasked with deployment in the production environment, while the developers are tasked with coding. Through collaboration, these experts realize vast chunks of work within a short period (Karamitsos et al., 2020). DevOps capabilities can be divided into three main categories: technology and automation, management and process, and management monitoring.

i. Usage of version control in all their artifacts. This is the practice and the act of managing changes in software code. Version control systems help the software developers and DevOps teams manage code changes in software source codes. Version control has using tools such as Git have made it easy to track changes making it easy for software to be corrected in case of a bug or error without affecting other code segments. Therefore, this reduces the lead time for change since teams can easily work in individual components and easily identify them. More has to be done to ensure that these tools are able to connect to production environment and therefore any change that is done to code directly reflects to the production environment as the programmer wishes.

ii. Automation of the deployment. This involves automating the process of the unit and integration testing. Cloud systems also provide a way of automating software integration. Linkage of version control systems and deployment tools will see a new dawn in the field of software development. This will ensure that tests are done while the product development is ongoing and therefore upon commit to the version control, the code is directly deployment to production environment. Presently, the tools are separate but there are efforts to join them to ensure that the benefits of using these systems as one is garnered.

iii. We are implementing continuous deployment. This is made a success by using software such as configuration as a code (CI/CD) or the Rapid Deploy application. Due to the tools required, continuous deployment is also challenging in terms of cost. Therefore, there is need for invention of new tools which will reduce cost and be affordable to all teams.

iv. We are implementing continuous delivery. This is the DevOps practice of building, testing, and delivering software code improvements and the user experience using automated tools such as Rapid Deploy. Continuous delivery can be well implemented by ensuring that the code is always in deployable mode. This can be achieved through eliminating the bulk of integration testing to ensure that changes from all developers is easy integrated into production environment. Also, this can be enhanced further by eliminating code freezes and hence fully migrating to processes that ensure continuous delivery.

v. It makes the flow of work visible throughout the value stream. This is made possible by using automation tools to monitor and visualize repetitive work and therefore lead to accountability. This can be improved further by coming up with tools which can easily analyse user feedback in production environment and suggest changes to the development team accordingly. This would otherwise ensure that quality monitoring is automated as well and hence making frequent improvements easy and robust.

vi. They are working in small batches with a lightweight approval process. The DevOps improve by targeting lightweight unit deployments and improving their state. This can in turn be improved by ensuring that there are more advanced tools and expertise. Therefore, there is need to make sure that the batches are correct and to standards. Also, familiarity with the process in presence of a DevOps engineer can be a great boost at this point ensuring that every team member us used to the normal flow of activities hence improved reaction.

vii. They are encouraging and supporting learning. DevOps teams can learn and acquire new knowledge by introducing new working tools and software development methodologies.

This can be necessitated and improved by giving a challenge to the member each time. This will encourage them to always seek for new knowledge and therefore a great boost to team enhancement.

viii. They support and facilitate collaboration among teams. DevOps work as a team to realize a common goal and encourage teamwork and collaboration. Through this, their state is improved and continues to be more popular. Collaboration however can further be improved by ensuring that there are more accessible tools and that the resources are all withing the development pipeline. Sometimes, collaboration is failed by poor communication. Team members do not clearly understand their contribution to the teams and hence this interferes with the performance. To avoid this, each member check can be automated, and also clear communication given backed with follow up and proper documentation of instructions.

5.6.2. DevOps Practices

The DevOps teams are tasked with various engagements in a development pipeline. The effectiveness of how they perform their roles forms the basis of how effective a DevOps team is (Waller et al., 2015). Therefore, these roles are vital in realizing the four performance measuring metrics discussed at the chapter's start. The various practices include:

i. Configuration management. DevOps teams ensure that all software development and platforms are well configured and maintained. They can improve their levels and achieve quality performance by ensuring this. This can further be improved through employing quality configuration management tools and ensuring that there is a track for detecting and correcting the misconfigurations. This would otherwise improve customer satisfaction and experience and hence improving customer ticket metric.

ii. Continuous integration. The software developers continuously merge their software code changes in a shared repository, improving their performance and productivity. Changes for each developer can easily be tracked by enhancing continuous delivery. Build and test can be automated to increase continuous integration which further would impact lead time for change and deployment frequency.

iii. Continuous delivery. This is tasked with the ongoing provision of software code changes and user experience in their environment by using automated software and ensuring high-quality code. This practice is ensuring that the customer is always updated and hence improving the customer experience. This can be enhanced by having tools that easily bridge development and production environment in that developers get analysed feedback, work on

them as they deploy each change. This would be helpful in enhancing mean time to recovery and also customer ticket.

iv. Infrastructure as a code. This enables the DevOps team to test applications within a production-like environment during the software development cycle. Acceptance testing always takes time but with infrastructure as a code tool, this can be done during unit testing and therefore lead to a lot of improvements in the nature of final product, minimize revision and therefore be cost effective.

v. Continuous monitoring. DevOps is ongoing building, testing, and delivering software code and user environment. This is done in the last SDLC cycle. Continuous monitoring can be improved through enhancing the tools used at each stage of software development and coming up with a better reporting tool. This would otherwise ensure that reaction to changes is manged well and that all members of the team are well informed.

vi. Continuous deployment. This involves continuing the ongoing integration of software in a software environment. This helps improve continuous delivery and deployment in a DevOps software development pipeline. Deployment can be improved through integrating versioning tools with deployment tools. This will ensure that are changes are tracked as they are being made and consequently shared in the production environment. This is far from the existing tools in isolation and hence bringing some latency which should be removed at this point.

5.6.3. DevOps Challenges

Challenges are inevitable in any engagements of life and machine processes. In the presence of challenges, solutions are found, and more maturity is realized. Therefore, in DevOps, various challenges need to be suppressed to realize the best. These challenges include:

i. We are integrating tools of various domains. The primary goal of DevOps is to have a shared working environment. Therefore, integrating building, testing, and deployment tools become a challenge. This is prevalent also in ensuring that we have a dedicated tool for managing the whole DevOps pipeline processes.

ii. Migration from the traditional infrastructure to microservices. An organization will be updated in this fast-pace moving world or otherwise will be replaced by other organizations that embrace change. However, it is not easy for organizations using the older infrastructures to migrate to the new microservice infrastructures, which needs

things such as automation and configuration management. Therefore, the vast requirement of migration to microservices becomes a challenge. These changes also need to be managed and therefore there is less manpower skills to perform the actions. This in response needs further look into ensuring that the required expertise is available.

iii. Conversion of well-developed and defined processes to efficient ones (Snyder & Curtis, 2017). For change to be induced, one should look at all the processes, identify areas that need improvement, and then work towards those improvements. However, this becomes a challenge whereby some processes are reluctant to change, delaying the organization's activities. DevOps combines the development team and the IT stakeholders, and therefore some processes may tend to lag. I strongly agree that absence of well-defined procedures also lead to confusion among the processed and the involved parties and hence leading to latency.

iv. Adoption of new tools in a working environment. While changing to DevOps, some new improved tools are required by the DevOps team. However, the teams might not be familiar with the tools, which becomes a challenge. It can also be noted that these tools are new to many developers and therefore require time to familiarize with. In absence of commitment and prior training, they might not be fully explored and hence inefficient.

v. Resistance to change by the majority. Most team members and stakeholders might be trying to change. Some concerned people might feel that the advice to change to DevOps is a bad reflection and, therefore, might not be smooth. According to SLR, change might take time and at the same moment may lead to delays in processes. Therefore, any organization needs to be ready for any cost that might not yield profits at the transition period.

vi. Separate tools for the dev and ops teams. The development and operations teams use different tools in their work. Therefore, finding the difference, eliminating it, and integrating these tools to work in unison becomes a challenge. Also, it could be of greater impact if these tools would be merged by coming up with a more advanced tool that can carry out all these processes.

5.6.4. Ways of Mitigating the challenges and improve the State of DevOps

A problem is first identified, clarified, defined, the root cause is examined and then solution is designed. Therefore, challenges are like problems and they need plan for them to be solved. The mitigation strategies and the ways on how to improve the state of DevOps in this section was arrived at after examining various materials some published and other unpublished to identify the root cause and plan accordingly. Therefore, the recommended ways suggest a possible solution for improving DevOps and avoiding most of the challenges. These challenges can be minimized and contribute to an improved state of DevOps report and make monitoring easy. This can also aid in realizing the main DevOps performance measures and contribute to efficient DevOps services. The mitigation procedures are as follows:

As seen in the finding chapter, good and well-functioning management is key to the adoption and success of DevOps. This involves having good guidance and support, which is needed to ensure that the teams are stable and provided with the right tools and resources. This can result from sharing common organization goals and working towards achieving them.

DevOps teams work together, and therefore collaboration is a crucial concern. The team members need to collaborate and work towards common goals to achieve the set objectives and key results. This creates visibility, and hence, teams can track changes easily leading to a more improved DevOps team.

Use of automated testing and integration tools (Artač et al., 2016), there is a generation of reports and analytics of the team functioning and therefore helping in the process of making change and suggesting team improvements in order to achieve the business goals and improve the returns on investment.

The problem of tools integration can be solved by having the best orchestration tools suitable in providing an enterprise-level framework for the automation of the tool's environment (Karl et al., 2016). These tools are advantageous since they provide enterprise-level advantages and improve the DevOps team.

In order to have the best and accelerated DevOps monitoring and reports, organizations need to consider the management, the tools, and the cohesion of the working environment. in that way, enterprises will be able to realize massive success using DevOps and hence be ranked as elite performers.

5.7. DevOps Business OKRs

DevOps OKRs - objectives and key results are the goals setting frameworks that help highlight the priority items in an organization or in a specific department hence offering the direction towards alignment of the goals for the teams and companies (Leopold, 2021). There are common goals for the DevOps teams across the companies, and they align with the common business goals. DevOps use a combination of the agile development methodology integrated with suitable tools to aid in faster delivery of software and services. OKRs are used to trace the success of a DevOps initiative in an organization. OKRs drive progress and increase the team's productivity since everybody works in alignment with what is set in the OKRs document.

Some of the common DevOps OKRs are product development – the team focuses on ensuring that the desired quality of a product needed by a customer is developed. This ensures that the teams are efficient and working on a common goal. The analysis shows that this can be further enhanced by ensuring that there are shared goals and that both DevOps team and the management are involved in the goal setting process and there is proper financing of the project. Also, the customer should be actively involved in the process of development. This will ensure that the final product is deployed faster and that it is in agreement with the user needs.

Competitive advantage. Shifting to microservices which is the idea of DevOps, ensures that the organization stays on top of its competitors. When change comes, the benefit comes to those who embrace it first, and therefore, the more developed and focused the DevOps, the larger the market share. My thoughts on this based on the studies is that the earlier the transformation the greater the benefits. Having functioning teams aids in making sure that an organization captures the market early enough.

More productive teams. In order to ensure that you get the best out of the human resources employed, you must make your team happy. The psychology of positivity argues that a free human being is more productive than a human being doing what they feel is not meant for them. Through the creation of happy teams, productivity is increased, and hence the state of DevOps is accelerated. I argue that where there is freedom, there is productivity. When the team works on motivation without being pushed gives good results. Therefore, the OKRs should be based on ensuring that each member of the team feels comfortable and suited for the role they are playing.

The primary source of errors is a human being, and therefore, in the presence of human beings, errors are inevitable. It is always wise to ensure that the developed system does not

interfere with the security or lead to loss. Therefore, there is a need to identify bugs and their correction quickly. This is viable through automation and the existence of a happy and more productive DevOps team capable of identifying bugs and correcting them. The DevOps OKRs framework can further be improved by ensuring that there are well laid out strategies backed up with plan to improve. There should also be a mission and vison guiding the whole organization and the individual teams to always do better than expected. Vison would also be shared among leaders who would trickle it down to team member to work and leave by it hence ensuring smooth achievement of the set OKRs.

5.8. Maturity Models in DevOps

DevOps is not a destination but a journey towards developing a more reliable and efficient pipeline, automation and development, and business teams. Maturity models are put in place to ensure that you can assess where your DevOps team is in their journey towards achieving the set OKRs. The organization commonly uses maturity models such as CMMI, CMM, and ISO/IEC 15504 in accessing and measuring their performance and progress.

In DevOps, there are various maturity models, such as the IBM DevOps maturity model, whose in-depth access and adoption can promote continuous software delivery access leads. Mohammed's maturity model is based on the capability maturity model integration (CMMI). It includes five maturity levels: initial, managed, defined, measured, and optimized based on four key components: quality, interaction, automation, and accountability, following Mohamed's maturity model, increased visibility, operational efficiency, and risk evasion. Other maturity models include the HP enterprise DevOps maturity model, Buena DevOps maturity model, and Feijter maturity model. They are all geared towards ensuring that there is a realization of maximum performance by the DevOps and that the teams are aware of where they are in terms of performance in a given dimension, such as automation, software delivery to market. The above discussed maturity models each has its own strength and weaknesses. Therefore, in order to access most of these models, there is need for the teams to ensure that they have their customized maturity model to help them align some of their concepts and also focus on their specific areas of improvement. This would otherwise help them in improving each day and taking their DevOps team to the next level and hence enhancing the state of DevOps.

5.9. A Case Study of Capital DevOps

Capital one is famous for being at the fore front when it comes to the adoption of new technologies to help it in transforming customer experiences in the banking sector (Wiedemann et al., 2019). Less often but crucial to capital one is the practices and mindsets

that positions the organization at a point to make such bold steps to use new technologies to transform the company activities and functioning. The company's latest embrace is the concept of DevOps.

George Brady, chief technology officer and executive vice president at capital one reiterated that about a decade ago they realized in order to continue being a great bank, they have to reinvent themselves as a digital technology company. He further added that in order to be able to be a great technology company, they have to build and architect their own systems and set up a developer culture which would otherwise help them attract and retain the most talented developers.

Capital one company laid the foundation of using DevOps which is the latest technology in strengthening the company's culture in 2010 when they shifted from waterfall to agile software development methodology. DevOps uses monitoring, automation and continuous integration of new code in achieving faster development cycles and more frequent, more reliable releases, which is good for a company which wants to be responsive to customer feedback soonest possible.

The company has a cloud first policy in which new applications are architected for and deployed in the cloud. In addition, capita one is also increasing the use of microservices and open, integrated architectures (Dinner, 2020). The technology is continuously increasing their capabilities in collecting and getting insights and feedback from the customers in which they are able to act on quickly by the aid of DevOps. For faster responses, the company uses AWS services such as Amazon VPC, amazon simple storage service, amazon elastic compute cloud, and amazon relational database service.

Use of AWS cloud services have been one of the turning points for the company's DevOps team. one of the advantages associated with using AWS services include reduction on the time required to build infrastructure for a new application as a result of using instantaneous infrastructure available at AWS hence ensuring that the DevOps team has the building blocks required to start working on a new software product (Soni, 2016). DevOps has also helped capital one to cultivate a more collaborative environment. According to Andrukonis, adoption of DevOps has made the developers to be more concerned with the software in terms of uptime, supportability, and monitoring unless the period before DevOps when their work ended after submitting the application to the operations team. Also, DevOps on the cloud has helped the designers, developers, and engineers to collaborate towards improving the customer experience (Andrukonis, 2017).

5.10. Proposed Strategy on How to Improve the State of DevOps Measurement and Monitoring

The thesis findings based on sources such as Atlassian (Atlassian, 2017) and Puppet annual state of DevOps report suggests that the key to improve the state of DevOps is efficient measuring and monitoring. Therefore, after successful analysis of the results and consideration of various aspects such as the teams and tools, I came up with some proposed ways in which state of DevOps measuring and monitoring can be improved. Therefore, this section evaluates the difference between the least and the most performing DevOps teams. Upon examination, it suggests the practices that leads to successful software delivery and operation performance which is ideal for an organization to grow in terms of DevOps monitoring and measuring. Among the ideas suggested and discussed is how a healthy team mitigates burnouts in times of challenges, how highest performers can continue to be at the par, how to realize significant benefit by leveraging the five key team's capabilities and creating a strong DevOps capabilities implementation foundation through documentation.

In order to meet the changes of this ever-changing business environment, companies need to deliver and operate software quickly and reliably. The quick and reliable deployments give room for running experiments and getting immediate feedback from the customers. Therefore, the two more vital things to consider when measuring and monitoring DevOps metrics is software delivery performance and operational performance (Debbiche et al., 2019). According to the thesis findings concerning the strength of each DevOps metric, I suggest the usage of four metrics for improving software delivery performance and another additional metric for improving operational performance. Software delivery performance can be measure by considering the throughput and stability. Lead time for change and deployment frequency can be used to measure throughput. Stability can be measured using the mean time to recovery and change failure rate. Therefore, for classification as either elite, high or low performer, an organization must have a significant performance in the four metrics. Elite performers do well in all the four metrics while low performers do worse in all the four metrics. Therefore, focusing on the four DevOps metrics is a one of the proven ways of increasing DevOps performance.

Among the suggestions on how to improve the state of DevOps measuring and monitoring is usage of cloud services. Cloud helps in development pipeline automation among others. According to recent trends, cloud has been at the forefront in ensuring that the DevOps team throughput has been improved. According to the findings on the state of DevOps as presented

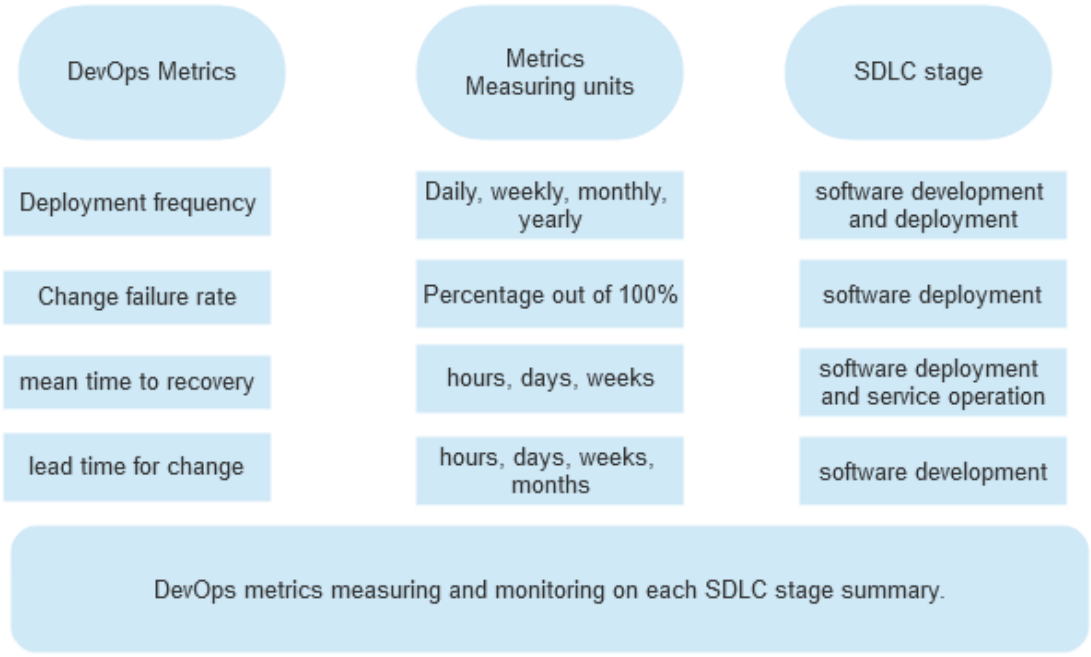
in Puppet annual report on state of DevOps adoption (Puppet, 2019), many people are migrating to hybrid and multi cloud which is making them increase the business value and access more customers and hence aligning with DevOps objectives and key results. The main reason behind usage of multi cloud is to leverage the benefit from each cloud provider alongside availability.

Another suggestion to accelerate the state of DevOps measuring and monitoring is use of quality documentation. This ranges from readmes and user manuals. An excellent documentation helps the readers to accomplish their goals, is comprehensive, up to date, accurate, clear, well organized, and findable. Documentation can be improved by proper utilization of both internal and external materials by the content providers accompanied by the technical team. For successful documentation, one should document the critical use cases for the products and services being deployed to the market, creating a clear strategy for editing and updating existing documentations and defining the owners of the software product or service. Through this, even if the team is changed, there will be effectiveness on the teams and hence improving the state of DevOps metrics measuring and monitoring.

Another suggestion on how to improve the state of DevOps measuring and monitoring is integration of security in the entire DevOps pipeline. According to the findings on monitoring tools and other DevOps tools such as orchestration tools (Rafi, Yu, Akbar, Alsanad, & Gumaei, 2020), a higher percentage of the elite performance had intact security integrated with their system. Security is one of the vital pillars to consider in this technological world. A small security breach can lead to a huge damage and disruption of the services. Therefore, ensuring that the security required is put in place improves the team's throughput since there are unnecessary fears and concerns alongside corruption of the systems.

In general, improving the state of DevOps monitoring and measuring requires the support of the who organization. The management should be willing to support and collaborate with the DevOps team into ensuring that the necessary tools, expertise, and security is guaranteed (Nielsen et al., 2017). With this in place, the deployment frequency is increased, lead time for change is reduced, mean time to recovery is significantly reduced and the change failure rate is also reduced. This leads to effectiveness in the organization activities, delivering value to customers soonest possible and having efficient services and hence huge returns on investment.

Figure 15
DevOps Metrics Monitoring and Measuring Stages Representation



The Figure 15 shows a graphical representation on the discussion on the key DevOps monitoring and measuring in each SDLC stage. The diagram was adopted from thematic analysis through checking on how carious DevOps teams measure their efficiency in terms of each DevOps metrics. The data included review of Amazon web services and also how they manage their platform and ensure that metrics such as mean time to recovery does not affect their users. The diagram shows a visual representation of each metric, its measuring unit, and the stage at which it is so critical. From the diagram, it is evident that DevOps measuring, and monitoring is vital at each stage. Therefore, paying a key attention to each metric displayed above can lead to positive change in a DevOps team. This can be achieved through involving the required resources and efforts. Therefore, for the state of DevOps monitoring and measuring improvement, all the factors have to be considered and the necessary risk taken. With due efforts, the success of DevOps metrics monitoring, and measuring will be easily achieved.

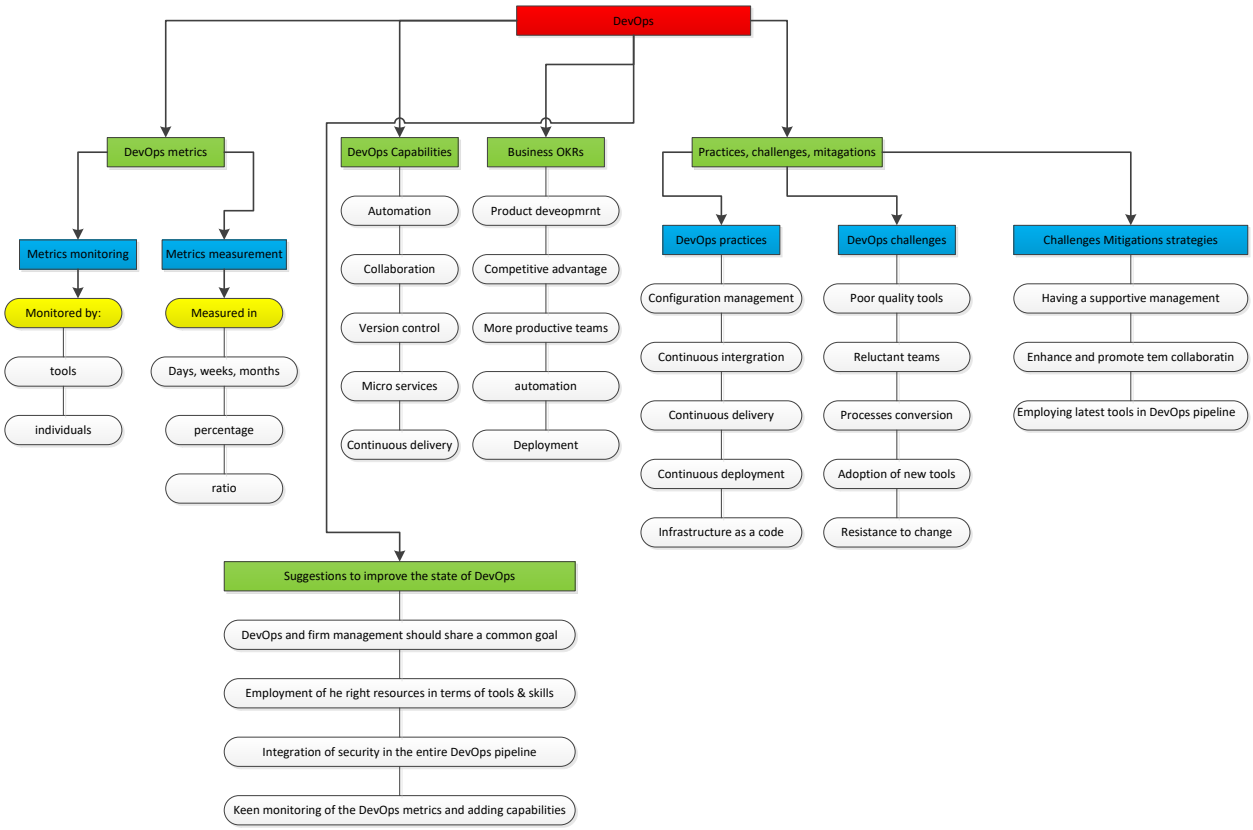
5.11. Summary of Key Ideas and Conclusions

In summary, DevOps needs to sharpen various aspects to achieve the set objectives and key results. DevOps metrics is one of the key performance indicators for the DevOps teams and hence needs to be well taken care of in order to realize the best performance. With the first

two metrics, deployment frequency and mean lead time to change measure speed, and mean time to recovery and change failure rate represent a measure of stability. For excellent performing DevOps team, speed and stability are needed and therefore crucial to pursue by DevOps team and management collaboration. By realizing these four metrics and controlling the cost of running the enterprise, the team is ranked as elite performers. However, effectiveness remains a dream without understanding the tools and human resources required for DevOps. Therefore, the leads and the management should consider their decision when investing in the tools and human resources to be outstanding in the company. There are various techniques for measuring DevOps metrics achievements, including how often they release software to the market, how long do the team take to correct an error in the system and the time it takes for the team to introduce new changes in a development environment. With deployment frequency at a daily, lead time for change being less than one day, mean time to recovery being less than one hour, and a change failure rate of 0 – 15%, the DevOps team is described as efficient. Efficiency is the goal of every DevOps initiative and, therefore, can be attained through team collaboration with management and avoidance of challenges such as the adoption of new tools and the existence of separate tools of development and operations team by being flexible for change.

Figure 16

Discussion Summary



The Figure 16 is a graphical representation of the key metrics which determine the developers team throughput and the operations team reliability. It shows the metrics, their measurement, and their monitoring. It also evaluates the individuals responsible for measuring and monitoring each metrics along the advantage and disadvantage of using the suggested monitoring and measuring unit. Therefore, a proper and strategized consideration and improvement of these metrics brings an overall growth to the activities of the concerned DevOps team.

6. Chapter 6 - Conclusion

The research aimed to find the possible methods for evaluating a DevOps team's capabilities. The evaluation results would otherwise recommend the subsequent actions taken for the teams to continue building their DevOps capabilities and improving their state alongside growth. This was achieved through the evaluation and understanding of the common objectives and critical results for DevOps – this looked at how the DevOps team is working in order to achieve its set objectives and key results alongside what changes should be made to the objectives and key results in order to expand their capabilities. Secondly, the research intensively looked at the metrics playing a critical role in describing the functions of the DevOps team – this considered how the metrics could be measured in practice and involving different teams. By deep analysis of these metrics, the team's capabilities are realized and therefore aid in ensuring that the change in the objectives and key results is aimed at realizing goals and more capabilities. The research also considered the achievement and challenges which are experienced in the process of improving DevOps practices. These mainly aids in ensuring that achievements are maximized and that the challenges are minimized to realize some considerable success.

In improving the DevOps capabilities, the management should be well informed on the business value of investing in the teams. Therefore, in this case, the research focused on the main metrics that can be used to measure the performance and define a DevOps team's success. These metrics were found to be as follows: mean lead time for a change, deployment frequency, mean time to recovery, and change failure rate. The research also focused on measuring and monitoring metrics effectively and efficiently. It involved outlining various methods used in DevOps monitoring and the different professionals used in measuring the DevOps progress and capabilities. The research also outlined and discussed various methods and parameters used in measuring different DevOps metrics, which are crucial to performance evaluation. In addition, the research mentioned various stages of the Software Development Life Cycle where these metrics are measured and how. The research further looked at the cost of doing all the said activities to improve DevOps capabilities. It outlined a cost analysis of the metrics monitoring and measuring and therefore engaging the possible risk. Finally, the researchers analysed a DevOps metrics profile and identified possible areas for improvements. This was said to help create insight for the DevOps team in terms of performance, productivity, scaling, and improvements and hence a possibility for suggesting a change to a better DevOps team.

DevOps objectives and key results are the key drivers to success since they push each and every stakeholder of the organization to look at the bigger picture. I would recommend that for expertly performance that the teams always share a common goal and management always remain in touch with the employees to encourage them and create a sense of mutual responsibility.

6.1 DevOps Objectives and Key Results

The research extensively examined drafting the team's objectives and key results specified for a certain period. It also extensively covered what must be considered when switching from one set objective and key results plan to another. The research also states the need and how the whole enterprise should share the objectives and main change in the long run. The research also states the roles of various stakeholders in ensuring that the set objectives and key results are achieved. Accordingly, the steps of switching from one plan to another are outlined with a precise explanation.

The primary purpose for DevOps is to ensure faster development and quickly make changes and provide first-hand feedback for those changes when the need arises. The research suggests that objectives and key results help the enterprise set business goals and key results. Therefore, these goals should align with the business mission and vision. For a successful DevOps team aligned to the business goals, the objectives should be innovative, measurable, achievable, realistic, and timely. The team should set their limits according to their strength and set a period to give them enough time to achieve.

According to the research, DevOps is that the one who develops software has complete control of that software in a production environment. Different understanding of the term DevOps when setting objectives and key results includes having complete control over infrastructure, automation, and deployment. In this case, operations teams aid in processes such as automation in the deployment pipelines, security, scalability, and more so working together in solving crisis when it arises. In this cooperation, the set objective and key results are achieved, and therefore creating a room for improvements every day.

Ultimately, the DevOps team objectives and key results define the business priorities. Therefore, the use of OKRs aid in identifying the business process and progress since they aim at the final product. Through OKRs, there is a precise plan formulation strategy that ensures that the team focuses on the final product. Considerably, some of the common objectives and critical results for DevOps include increasing the rate of deployment frequency, shortening

deployment cycles, and quality releases that align with the business goals. This gives the enterprise a competitive advantage by delivering the product according to the client's expectations. It also makes the teams more productive since everyone aligns with the set OKRs. All this results in a happy team that is more productive by reducing errors in the DevOps development pipeline.

6.2 DevOps Capabilities, Challenges, and Practices.

OKRs align precisely with the capabilities, including using version control, implementing continuous integration, automating the deployment process, and using trunk-based development methods. Challenges are the obstacles that prevents DevOps' adoption and growth. They prevent the potential that underlies DevOps teams and hinder their growth towards the prime level of performance. Some of the DevOps challenges include difficulties in adopting new tools in a working environment, conversion of well-developed and defined processes to efficient ones. Migration from traditional infrastructure to microservices and integration of tools from various domains. One of the most outstanding achievements of DevOps teams is integrating teams from different domains to work together.

There are various practices in that DevOps are involved in their day-to-day engagements. Their main aim of faster deployment to the market drives all their energy towards achievement. These practices define what a DevOps team is and how they are supposed to do in a production environment. Some of the practices crucial to the team and drive their success include configuration management, continuous integration, continuous delivery, infrastructure as a code, continuous deployment, and continuous monitoring, among other beneficial engagements. This forms the basis of their effectiveness, and therefore how well they perform depends on the percentage completion of their engagements. Therefore, this is also one of the performance measures in place depending on their performance and defining how well the team performs. These practices can be improved by doing more regression testing, always striving for continuous improvements, and taking hold of the basics of the domain every individual in the team is concerned with.

Challenges are inevitable in any field. However, these challenges aid in unleashing more potential and opening more opportunities for improvements and the development of uniqueness. In DevOps, various challenges have prevented and its growth over the years. It is a common thing that not all people embrace change. Changing from the old software development infrastructure to microservices is not a one-day implementation. Therefore, the adoption and growth of DevOps require the efforts of the entire enterprise, from the

management to all the teams in the firm. However, there are always critics who need to be severity fought to include change. This can be only possible through having supportive management that is ready and flexible for change and then proves the critics wrong. Therefore, the success and continuity of the DevOps team depends on the collaboration of the enterprise into focusing and realizing change. Other Challenges such as adopting new tools call for the management to pan on the right resources to transform their development pipeline.

It should be noted that resources do include tools only and human resources. With the right tools but less experienced and knowledgeable human resources lead nowhere. Therefore, the research suggests that the firm should invest in the right human resources to ensure that change is well. Through this, DevOps adoption and capabilities improvements will be massive.

DevOps capabilities are one key issue of discussion in this research. Accordingly, their capabilities define their performance and hence the need for improvement in day-to-day capabilities. Capabilities are defined as to what extent someone can perform or showcase their abilities. Abilities are earned in our day-to-day learning, and therefore, the more a team is exposed and presented with different ways of performance, the more that team improves in performing their roles. For example, if you are faced with a problem, in solving that problem, you gain much knowledge, which helps you to do many other things and solve many other problems. Therefore, DevOps capabilities improvements are not a one-day task but a growing exponential curve. Some of the DevOps capabilities need interactions with more knowledgeable professionals to accumulate the skills needed to outstand in that given field. Therefore, the research suggests that the teams be more optimistic and open to learning new ideas and adopting new ways and tools in their working environment. Some of the capabilities which are open and have greater chances of improvement include: using version control for all artifacts in production which can be possible for agile development approaches and usage of advanced tools in increasing productivity and deployment rates, automation of deployment process which can be done through using the same deployment process for your software, allowing anyone with the required credentials to deploy any version of artifacts to any environment on demand in a fully automated fashion and usage of same package for every deployment environment, implementing continuous integration which can be improved through ensuring there is secure senior leadership support before starting continuous integration, cleaning your environments and building only once, and using trunk-based development methods which can be improved through practices such as developing in small batches, performing synchronous code reviews, implementing comprehensive automated

testing, having a fast build and create a core group of advocates and mentors. Through implementation and monitoring of the said capabilities, the research suggests that the DevOps capabilities can be revived and improved hence accelerating the state of DevOps.

DevOps capabilities can be enhanced by encouraging team work and also empowerments. Consequently, challenges are inevitable and therefore when doing a design or else making sure that these challenges are minimized. Also, I would recommend frequent education to the team to keep increasing their knowledge to minimize the number of errors.

6.3 DevOps Metrics

Improvements can be applied to what can be measured. Therefore, the adage that you cannot improve what you cannot measure also applies in DevOps. A useful DevOps metric is measurable, relevant, reliable, actionable, and traceable. Therefore, metrics are data points used to measure the DevOps performance and suggest improvements on the said team. The research suggested four main DevOps metrics according to the google Dora performance measure scheme. The metrics represented the main performance measures that lead a team towards achieving the set KPIs. The four main metrics emphasized include deployment frequency, mean time to recovery, mean lead time for a change, and change failure rate. This involved analysing the cost of monitoring these metrics, how they can be improved, and categorizing teams according to their level of performance following these metrics.

According to the level of performance relative to these metrics, there are three categories of DevOps teams according to performance: prime performers, high performers, medium performers, and low performers. These different DevOps teams categories are realized by looking at their deployment frequency mean lead time for a change. The elite performers have deployment frequency being set at daily while deployment frequency for low performers is at six months. Therefore, from start-up to elite performers is a process that requires the engagement of every individual in the organization. It also requires a management that is flexible for change and supportive with being flexible to change. This is evidenced by finding elite performers and being at the top in term of performance.

Initially, DevOps was not recognized, but it is considered a unique skill in many companies due to their massive success. This is due to elite performers who fought hard to ensure case studies for DevOps start-ups. These organizations suggest that the DevOps team and the management play a crucial role in improving DevOps performance and their growth. Supportive management ensures that the deployment frequency is maximized and that the

lead time for change is minimized by ensuring that the team has the required tools and resources.

Different metrics have different ways of measuring. However, one metric cannot be used to measure and determine the team's performance. The combination of performance measures of the four main metrics is the ones that are used to measure DevOps team performance and therefore classify it as either low, medium, high, or elite performer. Then, what are the different units used in measuring the DevOps metrics? Deployment frequency is how code increments are deployed to staging, testing, and production (Lwakatare et al., 2019). It is measured through how many times code is deployed daily, weekly, monthly, or yearly depending on the capabilities of a team. Mean time to recovery is used to measure the time it takes to return the system to its normal state when an error occurs—measured in seconds, minutes, hours, or days used to correct the system. It should be the shortest possible to the point that the users of the systems do not notice. Mean lead time for change is essentially how long a team takes to go from a code committed to code successfully deployed running in the production environment. It is measured in days, weeks, or months with prime performers taking less than a day and low performers taking up to six months. Change failure rate is the percentage of changes that resulted in destroyed services such as service outrange or impairment and need to be fixed. It is measured on the percentage of the deployments attempted to those that failed. With the lowest change failure rate percentage, lowest lead time for a change, high deployment frequency, and low mean time to recovery, the team is classified as a prime performer since that is an optimal performing team.

DevOps monitoring is a key towards planning on the improvement of DevOps capabilities. However, this requires funds in order to be a success and impactful. It helps in monitoring the critical bottlenecks in the DevOps team's progress. DevOps monitoring involves overseeing the entire development process from product planning, development, integration, testing, deployment, and operations (Alnamlah et al., 2021). It engages the whole and real-time view of the status of applications, services, and infrastructure in the production environment. Traditional monitoring is no longer enough to realize the full potential. Therefore, some tools are developed and recommended for monitoring DevOps and giving figures for the performance of each metric. These tools are cloud-based, and they include SolarWinds AppOptics, Sensu Go, Microsoft Teams, among others which range from $3 to $30 per month.

DevOps metrics are concerned with measuring the extent and efficiency of the team in product development and their response to software processes. Therefore, these metrics

should also be put into tools that can be able to record and find aggregate based on the performance of each key metric. Having a universal tool for carrying out the metrics evaluation and reporting.

6.4 DevOps Maturity Models

The DevOps maturity model is a model which determines where an organization stands in the DevOps journey while defining what is required to be done in order to accomplish the desired result. In the study, it came across various maturity models used by various organizations in order for them to achieve the desired results. This entails understanding DevOps adoption as a journey but not a destination. The DevOps maturity model determines growth through continuous learning from both teams and organizations perspectives. With more acquired skills and capabilities, the teams can handle issues of scale and complexities.

Types of maturity models include the IBM DevOps maturity model, whose deep examination can aid in promoting continuous delivery. This model suggests the implementation and adoption of four main aspects: planning and measuring, developing, and testing, releasing and deploying, and monitoring and optimizing. However, this maturity model has a flaw whereby it does not have a clear rationale for capital approach in obtaining DevOps maturity and it is not applicable to non-IBM running software's.

The second maturity model considered is the Mohammed DevOps maturity model. The suggested system is based and predicated on the capability maturity model integration. It includes five maturity models distributed across the four key components: quality, automation, interaction, and accountability. Accordingly, implementation of this model is said to improve operational efficiency, increase visibility, and mitigate risks such as downtime during adoption. Other maturity models studied and discussed were Hewlett Packard Enterprise DevOps maturity model, Buena DevOps maturity model, Eficode maturity model, and Feijter maturity model. All these models work towards a common goal of ensuring that DevOps adoption is improved.

Each DevOps maturity model studies uses a different feature and some unique aspects in determining the success of a team. Therefore, I would recommend development of a unified model which would be using the most obvious aspects in accessing a team and hence giving the correct maturity of a team.

6.5 DevOps Tools and Processes

Various tools are necessary for the DevOps development pipeline. These tools are used in various stages by both the development and the operations teams. According to the study, it was evident that to improve the DevOps team's state, and the teams must apply the necessary tools to post exceptional results.

According to the research, the development pipeline is divided into sections, each with the recommended tools that work best in a particular stage. With the adoption of microservices and the employment of automation in software development, there is a need to select the best tools for doing all the activities concerning various considerations such as security and shared resources.

The common tools involved in the DevOps development pipeline includes cloud and container management tools used for cloud management, system configuration and provisioning tools concerned with systems state definition, and task execution tools that allow the teams to run the scripts in different execution environments. Finally, we have application integration tools that aid in converting source code and other artifacts into a form in which code integration and validation can be done and released to the market. Therefore, by using the right tools in suitable environments while employing the right skills, the state of DevOps will be accelerated, and the challenges minimized while improving on their growth.

There is another dominant category of DevOps tools called Orchestration tools. These are DevOps tools which are used in automation of the workflow processes which help in the delivery of resources as a service. DevOps orchestration can be termed as the coordination of the whole enterprise in DevOps practices and automation tools employed in the process of accomplishing the set business goals and key results. This is what makes term orchestration so important in the context of DevOps. DevOps can be best orchestrated by practices such as shifting left with CI/CD, employing agile software development management, observability, monitoring, automation, and continuous feedback. In choosing an orchestration tool, factors such as the enterprise and IT deployment size, enterprise operating system, open source and commercial tools are considered first. Orchestration tools are of ultimate importance in improving DevOps capabilities because they reduce the cost of production, reduce the error rate, increase the team’s productivity, greatly saves the developers time through automation, they lead to faster operations and minimize system outranges while reducing the system’s downtime. Some of the best orchestration tools in the market today include Google Kubernetes Engine, Portainer, Mirantis Kubernetes Engine, Nomad, and Rancher. I would

recommend development of tools that are able to monitor and bridge all the stages in DevOps pipeline. This will ensure that there is faster maintenance and that also each team will familiarize with the required processes.

6.6 DevOps Metrics Measurement and Monitoring

DevOps metrics are the performance measure for DevOps. A combination of various DevOps metrics and how well they are accomplished determines the capabilities of that DevOps team. According to the research conducted, there are four DevOps metrics which are of ultimate importance to the DevOps teams and act as the Key performance Indicators for any DevOps if well outsourced. These metrics include mean lead time for change, deployment frequency, mean time to recovery and change failure rate. These four metrics have different units of measurement and approach to their monitoring. Successful measuring and monitoring of these four metrics can slowly lead to DevOps team lucrative achievements.

Briefly looking at each DevOps metrics, deployment frequency can be measure and evaluated on how often the team releases software to the market. This can be as frequent as daily, weekly, monthly, or annually. It can be monitored through feasibility study, market analysis or cost analysis. The second metric of importance is lead time for change which can be measured in term of days, weeks, or even months. This metric can be monitored by recording the number of successful changes made in certain period of time. The third metric of importance under study is the mean time to recovery and can be measured in terms of hours used in correcting an error in the system. This metric can be monitored by the use of tools such as SDM which makes guesses out of the system progress. The last metric of ultimate importance is change failure rate which is measured in terms of the percentage of the number of changes that led to damage services. Change failure rate can be monitored by surfacing data from CI/CD tools such as JenKins.

These suggestions and tools bring change in the performance of the DevOps. Therefore, adherence to the study discussion can result to increased team capabilities, higher business goal achievements and positive reputation in the market.

DevOps monitoring and measuring can be improved by ensuring that there are effective tools integrated in the Development pipeline. I would recommend that the teams agree on performance and common goals and then set a reminder for the tools to remind that it is time to comply especially for the metrics that use unit of time.

6.7 Summary

The study generally looked at the DevOps and how its state can be leveraged at the Organization level. The underlying problem that the thesis tried to solve was on how the relationship between DevOps metrics, capabilities and business objectives and key results can be enhanced to further improve the functioning of DevOps. This was enabled through answering questions such as the cost of measuring DevOps metrics, exploring the DevOps pipeline. In addition. The aim was to compare and contrast the existing DevOps metrics and recommend practices which can be adopted to further enhance the DevOps and hence encourage organizations to continue with their adoption.

6.8 Limitation and future recommendation

During the data collection process, it was not easy due to few available materials in relation to DevOps. This is as a result of the technology being at its growth stages and therefore having few researches. Also, the practical aspects of DevOps were not easily accessible since there are a few companies who have adopted the practice. Some of the future recommendations or research ideas on how to make the DevOps pipeline more effective, through investment in tools and also coming up with a unified too to aid in the management of the whole DevOps process. In addition, there is a great gap in the formality of DevOps, it is not well development and therefore lacks a rich background for introducing it in curriculum of most information technology schools. This can however be made a success through ensuring that there is well researched and proved syllabus for the framework and practices in DevOps.

References

Adam, H. (2022). *Measuring DevOps Success: What, Where, and How.* Retrieved from Cloudacademy: https://cloudacademy.com/blog/measuring-devops-success-what-where-and-how/

Aguiar Monteiro, L., Pessoa Monteiro, D., Carvalho Almeida, W., Cavalcanti De Lima, A., & Sette, I. (2020, 12). Methods of Implementation, Maturity Models and Definition of Roles in DevOps Frameworks: A Systematic Mapping. *Proceedings - 2020 International Conference on Computational Science and Computational Intelligence, CSCI 2020*, 1766-1773. Retrieved from https://www.researchgate.net/publication/351783429_Methods_of_Implementation_Maturity_Models_and_Definition_of_Roles_in_DevOps_Frameworks_A_Systematic_Mapping

Ahmed, W., & Rashdi, M. (2020). Understanding the influence of lean and agile strategies on creating firms' supply chain risk management capabilities. *Competitiveness Review: An international Business Journal*. https://doi.org/10.1108/CR-03-2020-0040

Airaj, M. (2017). Enable cloud DevOps approach for industry and higher education. *Concurrency and Computation: Practice and Experience, 29*(5). https://doi.org/10.1002/cpe.3937

Alnamlah, B., Alshathry, S., Alkassim, N., & Jamail, N. (2021). The necessity of a lead person to monitor development stages of the DevOps pipeline. *Indonesian Journal of Electrical Engineering and Computer Science, 21*(1), 348.

Al Thinyan, K. T., Ghawji, H., & Al Shehri, A. What are OKRs and KPIs and can they Coexist within an Organization?.

Amazon. (2021). *Cost*. Retrieved from aws.amazon: https://docs.aws.amazon.com/solutions/latest/aws-devops-monitoring-dashboard/cost.html

Andrukonis, J. (2017). *On-Demand Infrastructure on AWS Helps Capital One DevOps Teams Move Faster Than Ever.* Retrieved from Amazon: https://aws.amazon.com/solutions/case-studies/capital-one-devops/

Ankur, M. (2018). *What is DevOps, It's Working, Benefits, Tools in Detail*. Retrieved from https://www.dotnettricks.com/learn/devops/what-is-devops-and-devops-advantages

Angara, J., Gutta, S., & Prasad, S. (2018). DevOps with continuous testing architecture and its metrics model. In *Recent Findings in Intelligent Computing Techniques* (pp. 271-281). Springer, Singapore.

AppDynamics. (2021). *DevOps Metrics and KPIs: How To Measure DevOps? | AppDynamics - (A Cisco Company)*. Retrieved from https://www.appdynamics.com/topics/devops-metrics-and-kpis

Artač, M., Borovšak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. (2016, 7). Model-Driven continuous deployment for quality devops. *QUDOS 2016 - Proceedings of the 2nd International Workshop on Quality-Aware DevOps, co-located with ISSTA 2016*, 40-41.

Arundel, J., & Domingus, J. (2019). Cloud Native DevOps with Kubernetes: building, deploying, and scaling modern applications in the Cloud. In *Cloud Native DevOps with Kubernetes: building, deploying, and scaling modern applications in the Cloud.* O'Reilly Media.

Badshah, S., Khan, A. A., & Khan, B. (2020). Towards process improvement in DevOps: a systematic literature review. Proceedings of the evaluation and assessment in software engineering, 427-433.

Bahrs, P. (2013). Adopting the IBM DevOps approach for continuous software delivery – IBM Developer. Retrieved from https://developer.ibm.com/articles/d-adoption-paths/

Bass, L. (2017). The Software Architect and DevOps. *IEEE Software, 35*(1), 8-10.

Batra, P., & Jatain, A. (2020, 7). Measurement Based Performance Evaluation of DevOps. *2020 International Conference on Computational Performance Evaluation, ComPE 2020*, 757-760.

Battina, D. S. (2021). AI and DevOps in Information Technology and Its Future in the United States. *INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT), ISSN*, 2320-2882.

Bezemer, C., Eismann, S., Ferme, V., Grohmann, J., Heinrich, R., Jamshidi, P., . . . Willnecker, F. (2019, 4). How is performance addressed in DevOps? A survey on industrial practices. *ICPE 2019 - Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, 45-50.

Bramer, W., Rethlefsen, M., Kleijnen, J., & Franco, O. (2017, 12). Optimal database combinations for literature searches in systematic reviews: A prospective exploratory

study. *Systematic Reviews, 6*(1), 1-12. Retrieved from https://systematicreviewsjournal.biomedcentral.com/articles/10.1186/s13643-017-0644-y

Brereton, P., Kitchenham, B., Budgen, D., Turner, M., & Khalil, M. (2007, 4). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software, 80*(4), 571-583.

Breyter, M. (2022). Starting with Why. In *Agile Product and Project Management* (pp. 23-42). Apress, Berkeley, CA.

Bucena, I., & Kirikova, M. (2017). Simplifying the DevOps Adoption Process. In I. Bucena, & M. Kirikova (Ed.). BIR Workshops 2017, Computer Science. Retrieved from https://www.semanticscholar.org/paper/Simplifying-the-DevOps-Adoption-Process-Bucena-Kirikova/a3b318909b45843c6c17b9789fb2d9f134f569c4

Callanan, M., & Spillane, A. (2016, 5). DevOps: Making It Easy to Do the Right Thing. *IEEE Software, 33*(3), 53-59.

Cardenas, E. (2018). *Roadblocks to Successful DevOps Often Human, Not Technical - DevOps.com*. Retrieved from https://devops.com/roadblocks-to-successful-devops-often-human-not-technical/

Carturan, S., & Goya, D. (2019, 9). A systems-of-systems security framework for requirements definition in cloud environment. *PervasiveHealth: Pervasive Computing Technologies for Healthcare, 2*, 235-240.

Charrois, T. (2015, 3). Systematic Reviews: What Do You Need to Know to Get Started? *The Canadian Journal of Hospital Pharmacy, 68*(2), 144. Retrieved from /pmc/articles/PMC4414076/

Chen, L. (2017, 6). Continuous Delivery: Overcoming adoption challenges. *Journal of Systems and Software, 128*, 72-86.

Cito, J., Wettinger, J., Lwakatare, L. E., Borg, M., & Li, F. (2018, March). Feedback from operations to software development—a devops perspective on runtime metrics and logs. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment* (pp. 184-195). Springer, Cham.

Claps, G., Berntsson Svensson, R., & Aurum, A. (2015, 1). On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software Technology, 57*(1), 21-31.

Creasey, T., & Hiatt, J. (2018). *Metrics for Measuring Change Management.* Retrieved from Prosci: https://www.prosci.com/resources/articles/measuring-change-management-effectiveness-with-metrics

De Bayser, M., Azevedo, L., & Cerqueira, R. (2015, 6). ResearchOps: The case for DevOps in scientific applications. *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, 1398-1404.

De Feijter, R., Overbeek, S., van Vliet, R., Jagroep, E., & Brinkkemper, S. (2018, 6). DevOps Competences and Maturity for Software Producing Organizations. *Lecture Notes in Business Information Processing, 318*, 244-259. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-91704-7_16

Debbiche, F., Wrang, M., & Sinkala, K. (2019). Accelerating software delivery in the context of requirements analysis and breakdown for DevOps: A multiple-case study.

Debois, P. (2008). Agile infrastructure and operations: How infra-gile are you? *Proceedings - Agile 2008 Conference*, 202-207.

DevOps Expert. (2014, 4 29). *A "taxonomy" for Devops tools*. Retrieved from digital.ai: https://digital.ai/catalyst-blog/a-taxonomy-for-devops-tools

Diel, E., Marczak, S., & Cruzes, D. (2016, 9). Communication challenges and strategies in distributed DevOps. *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*, 24-28.

Dinner, A. (2020, Feb). Factors that Influence the Synergy between Development and IT Operations in a DevOps Environment. Western Cape, South Africa.

Dornenburg, E. (2018, 9). The Path to DevOps. *IEEE Software, 35*(5), 71-75.

Dumitrescu, S. (2021, 7 7). *16 Challenges of DevOps in 2021 – From Adoption to Implementation to Scaling.* Retrieved from Bunnyshell: https://www.bunnyshell.com/blog/challenges-of-devops

Dwivedi, N., Katiyar, D., & Goel, G. (2022). A Comparative Study of Various Software Development Life Cycle (SDLC) Models. *International Journal of Research in Engineering, Science and Management*, *5*(3), 141-144.

Dyck, A., Penners, R., & Lichter, H. (2015). Towards Definitions for Release Engineering and DevOps. *Proceedings of the IEEE/ACM 3rd International Workshop on Release Engineering*. Retrieved from http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7169442&filter%3DAND%28p_IS_Number%3A7169432%29

Eeles, P. (2019). *10 Challenges to DevOps Adoption and How to Overcome Them.* Retrieved from Contino: https://www.contino.io/insights/5-challenges-to-devops-adoption-and-how-to-overcome-them

Eficode Oy. (2015). *Devops Eficode.* Retrieved from eficode: https://www.eficode.com/hubfs/documents/Eficode-English-Devops-Guide.pdf?hsLang=en

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *Ieee Software*, *33*(3), 94-100.

Elberzhager, F., Arif, T., Naab, M., Süß, I., & Koban, S. (2017). From agile development to devops: Going towards faster releases at high quality - Experiences from an industrial context. *Lecture Notes in Business Information Processing, 269*, 33-44. Retrieved from https://www.researchgate.net/publication/309961705_From_Agile_Development_to_DevOps_Going_Towards_Faster_Releases_at_High_Quality_-_Experiences_from_an_Industrial_Context

Elliot, S. (2015). DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified. *IDC Insight*(December).

Erich, F., Amrit, C., & Daneva, M. (2014). *Report: DevOps Literature Review.* Retrieved from DOI: 10.13140/2.1.5125.1201

Erich, F., Amrit, C., & Daneva, M. (2017, 6). A qualitative study of DevOps usage in practice. *Journal of Software: Evolution and Process, 29*(6), e1885. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/smr.1885

Farroha, B., & Farroha, D. (2014, 11). A framework for managing mission needs, compliance, and trust in the DevOps environment. *Proceedings - IEEE Military Communications Conference MILCOM*, 288-293.

Fereday, J., Adelaide, N., Australia, S., & Eimear Muir-Cochrane, A. (2016, 11). Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development:. *http://dx.doi.org/10.1177/160940690600500107, 5*(1), 80-92. Retrieved from https://journals.sagepub.com/doi/10.1177/160940690600500107

Ferry, N., Nguyen, P., Song, H., Rios, E., Iturbe, E., Martinez, S., & Rego, A. (2020). Continuous Deployment of Trustworthy Smart IoT Systems. *The Journal of Object Technology, 19*(2), 16:1. Retrieved from http://www.jot.fm/contents/issue_2020_02/article16.html

Forsgren, N., & Kersten, M. (2018). DevOps Metrics. *Communications of the ACM, 61(4)*, 44-48.

Forsgren, N., Tremblay, M., VanderMeer, D., & Humble, J. (2017). DORA platform: DevOps assessment and benchmarking. (pp. 436-440). Springer, Cham.

Ghantous, G., & Gill, A. (2017, 7). DevOps: Concepts, Practices, Tools, Benefits and Challenges. *PACIS 2017 Proceedings*. Retrieved from https://aisel.aisnet.org/pacis2017/96

Godin, K., Stapleton, J., Kirkpatrick, S., Hanning, R., & Leatherdale, S. (2015, 10). Applying systematic review search methods to the grey literature: A case study examining guidelines for school-based breakfast programs in Canada. *Systematic Reviews, 4*(1), 1-10. Retrieved from https://systematicreviewsjournal.biomedcentral.com/articles/10.1186/s13643-015-0125-0

Gokarna, M. (2021, 1). DevOps phases across Software Development Lifecycle. Retrieved from https://www.techrxiv.org/articles/preprint/DevOps_phases_across_Software_Development_Lifecycle/13207796/2

Gottesheim, W. (2015, 2). Challenges, benefits and best practices of performance focused DevOps. *LT 2015 - Proceedings of the 4th ACM/SPEC International Workshop on Large-Scale Testing, in Conjunction with ICPE 2015*, 3.

Graves Portman, D. (2020, 9 23). *Accelerate State of DevOps Report*. Retrieved from Google: https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

Gunja, S. (2021). *9 key DevOps metrics for success | Dynatrace news*. Retrieved from https://www.dynatrace.com/news/blog/devops-metrics-for-success/

Haindl, P., & Plosch, R. (2019, 5). Towards Continuous Quality: Measuring and Evaluating Feature-Dependent Non-Functional Requirements in DevOps. *Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019*, 91-94.

Hall, T. (n.d.). *DevOps metrics | Atlassian*. Retrieved from https://www.atlassian.com/devops/frameworks/devops-metrics

Hemon, A., Fitzgerald, B., Lyonnet, B., & Rowe, F. (2020, 5). Innovative Practices for Knowledge Sharing in Large-Scale DevOps. *IEEE Software, 37*(3), 30-37.

Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020, 8). From Agile to DevOps: Smart Skills and Collaborations. *Information Systems Frontiers, 22*(4), 927-945.

Hodges, E. (2020, 4 15). *Measurement Automation & Production Test with APx - Audio Precision.* Retrieved from AP: https://www.ap.com/blog/audio-measurement-automation-production-test-with-apx/

Hombergs, T. (2017, 5 7). *Monitoring the Error Rate of a Spring Boot Web Application.* Retrieved from Reflectoring: https://reflectoring.io/monitoring-error-rate-spring-boot/

Humble, J., Kim, G., & Forsgren, N. (2018). Accelerate: the science of lean software and DevOps: building and scaling high performing technology organizations. IT Revolution.

Hussaini, S. (2014, 11). Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach. *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, 178-183.

Hüttermann, M. (2012). Building blocks of devops. In *DevOps for Developers* (pp. 33-47). Apress, Berkeley, CA.

Inbar, S., Yaniv, S., Gil, P., Eran, S., Ilan, S., Olga, K., & Ravi, S. (2013). DevOps and OpsDev: How Maturity. *Hewlett Packard Enterprise,*.

Institue of Work & Health (IWH). (2008). *Grey literature*. Retrieved from https://www.iwh.on.ca/what-researchers-mean-by/grey-literature

Ivanova, A., & Ivanova, P. (2018). *DATA ANALYTICS FOR DEVOPS EFFECTIVENESS.* Retrieved from Ceeol: https://www.ceeol.com/search/article-detail?id=779785

Jaatun, M. (2018, 8). Software security activities that support incident management in secure DevOps. *ACM International Conference Proceeding Series*. Retrieved from https://www.researchgate.net/publication/327005802_Software_Security_Activities_that_Support_Incident_Management_in_Secure_DevOps

Jabbari, R., Ali, N., Petersen, K., & Tanveer, B. (2016). What is DevOps? A systematic mapping study on definitions and practices., *24-May-2016*, pp. 1-11.

John, W., Marchetto, G., Németh, F., Sköldström, P., Steinert, R., Meirosu, C., . . . Pentikousis, K. (2017, 1). Service provider DevOps. *IEEE Communications Magazine, 55*(1), 204-211.

Jones, S., Noppen, J., & Lettice, F. (2016, 7). Management challenges for devops adoption within UK SMEs. *QUDOS 2016 - Proceedings of the 2nd International Workshop on Quality-Aware DevOps, co-located with ISSTA 2016*, 7-11.

Kang, H., Le, M., & Tao, S. (2016, 6). Container and microservice driven design for cloud infrastructure DevOps. *Proceedings - 2016 IEEE International Conference on Cloud Engineering, IC2E 2016: Co-located with the 1st IEEE International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, 202-211.

Karamitsos, I., Albarhami, S., & Apostolopoulos, C. (2020). Applying devops practices of continuous automation for machine learning. *Information (Switzerland), 11*(7).

Karl, H., Dräxler, S., Peuster, M., Galis, A., Bredel, M., Ramos, A., . . . Xilouris, G. (2016). DevOps for network function virtualisation: an architectural approach. *Transactions on Emerging Telecommunications Technologies, 27*(9), 1206-1215.

Katal, A., Bajoria, V., & Dahiya, S. (2019). DevOps: Bridging the gap between development and operations. *Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019*, 1-7.

Kersten, M. (2018, 3). A cambrian explosion of DevOps tools. *IEEE Software, 35*(2), 14-17.

Kersten, N. (2021, July 20). *The 2021 State of DevOps Report.* Retrieved from Puppet: https://puppet.com/blog/2021-state-of-devops-report/

Khan, A., & Shameem, M. (2020). Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process. *Journal of Software: Evolution and Process, 32*(10).

Khan, M., Jumani, A., Mahar, F., & Siddique, W. (2020). Fast Delivery, Continuously Build, Testing and Deployment with DevOps Pipeline Techniques on Cloud. *Indian Journal of Science and Technology*, 552-557.

Kim, J., Meirosu, C., Papafili, I., Steinert, R., Sharma, S., Westphal, F., . . . Manzalini, A. (2015, 6). Service provider DevOps for large scale modern network services. *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, 1391-1397.

Kitchenham, B., & Stuart, C. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Retrieved from https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

Kljaić, M., & Šćepanović, S. (2022, February). DevOps methodology usage in IT companies in Montenegro. In *2022 26th International Conference on Information Technology (IT)* (pp. 1-4). IEEE.

Ko, A. (2017). A three-year participant observation of software startup software evolution. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*, 3-12.

Kupsch, J., Miller, B., Basupalli, V., & Burger, J. (2017, 12). From continuous integration to continuous assurance. *2017 IEEE 28th Annual Software Technology Conference, STC 2017, 2017-January*, 1-8.

Lawrence, C. (2021, 2 4). *Deployment Frequency – A Key Metric in DevOps*. Retrieved from Humanitec.

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019, 9). A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys, 52*(6). Retrieved from http://arxiv.org/abs/1909.05409

Leopold, M. (2021). OKR - Strategy Development and Implementation in an Agile Environment: Introduction to the World's Most Successful Framework for Strategy Execution in the 21st Century Paperback. In *OKR - Strategy Development and Implementation in an Agile Environment: Introduction to the World's Most Successful Framework for Strategy Execution in the 21st Century Paperback.* Books on Demand.

Liberati, A., Altman, D., Tetzlaff, J., Mulrow, C., Gøtzsche, P., Ioannidis, J., . . . Moher, D. (2009, 10). The PRISMA statement for reporting systematic reviews and meta-analyses of

studies that evaluate health care interventions: explanation and elaboration. *Journal of clinical epidemiology, 62*(10), e1-e34.

Linke, K. (2019). Traditional and Agile Management Approaches. In *12th ILERA European Congress, Düsseldorf, Deutschland*.

Liu, Y., & Zhou, Y. (2017, February). *Digitala Vetenskapliga Arkivet.* Retrieved from diva-portal: http://www.diva-portal.org/smash/get/diva2:1078159/FULLTEXT02

Lopez-Fern ´ andez, D., Diaz, J., Garc´ıa, J., Perez, J., & Gonzalez-Prieto, A. (2021). DevOps Team Structures: Characterization and Implications. *arXiv preprint arXiv:2101.02361*.

Luz, W., Pinto, G., & Bonifácio, R. (2018, 1). Building a collaborative culture: a grounded theory of well succeeded devops adoption in practice. *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 23-34.

Lwakatare, L., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H., Bosch, J., & Oivo, M. (2016, 3). Towards DevOps in the embedded systems domain: Why is it so hard? *Proceedings of the Annual Hawaii International Conference on System Sciences, 2016-March*, 5437-5446.

Lwakatare, L., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., . . . Lassenius, C. (2019, 10). DevOps in practice: A multiple case study of five companies. *Information and Software Technology, 114*, 217-230.

Lwakatare, L., Kuvaja, P., & Oivo, M. (2016). Relationship of devops to agile, lean and continuous deployment: A multivocal literature review study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 399-415). Springer, Cham. https://doi.org/10.1007/978-3-319-49094-6_27

Macarthy, R., & Bass, J. (2020). An Empirical Taxonomy of DevOps in Practice. *46th Euromicro Conference on Software Engineering and Advanced Applications*, (pp. 221-228).

Makosi , R. (2017). DevOps Concepts. Вестник Хакасского государственного университета им. *НФ Катанова*, 32-34.

Maroukian, K., & Gulliver, S. (2020). Leading DevOps Practice and Principle Adoption. *International Conference on Information Technology Convergence and Services*. Retrieved from https://arxiv.org/ftp/arxiv/papers/2008/2008.10515.pdf

Matskin, M., Tahmasebi, S., Layegh, A., Payberah, A. H., Thomas, A., Nikolov, N., & Roman, D. (2021). A Survey of Big Data Pipeline Orchestration Tools from the Perspective of the DataCloud Project.

McCarthy, M., Herger, L., Khan, S., & Belgodere, B. (2015). Composable DevOps: Automated Ontology Based DevOps Maturity Analysis. *Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015*, 600-607.Melgar, A. S. (2021). DevOps as a culture of interaction and deployment in an insurance company. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(4), 1701-1708.

Mishra, A., & Otaiwi, Z. (2020, 11). DevOps and software quality: A systematic mapping. *Computer Science Review, 38*.

MJÖLL , B., & HELÉN , E. (2018). Designing Continuous Toolchains Using Proposed Guidelines and Tool Capabilities. Gothenburg, Sweden.

Mohamed, S. (2015). DevOps shifting software engineering strategy Value based perspective. *IOSR Journal of Computer Engineering (IOSR-JCE), 17*(2), 51-57. Retrieved from www.iosrjournals.org

Mohamed, S. (2016). DevOps Maturity Calculator DOMC - Value oriented approach. *International Journal of Engineering Research and Science, 2*(2), 25-35.

Mohan, V., Ben Othmane, L., & Kres, A. (2018, 11). BP: Security concerns and best practices for automation of software deployment processes: An industrial case study. *Proceedings - 2018 IEEE Cybersecurity Development Conference, SecDev 2018*, 21-28.

Morales, J., Yasar, H., & Volkmann, A. (2018). *Implementing DevOps Practices in Highly Regulated Environments*. Retrieved from https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=531308

Mumbarkar, P., & Prasad, S. (2022, October). Adopting DevOps: Capabilities, practices, and challenges faced by organizations. In *AIP Conference Proceedings* (Vol. 2519, No. 1, p. 030029). AIP Publishing LLC.

Munn, Z., Peters, M., Stern, C., Tufanaru, C., McArthur, A., & Aromataris, E. (2018, 11). Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach. *BMC Medical Research Methodology, 18*(1), 1-7. Retrieved from https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-018-0611-x

Neelankavil, J. P. (2015). Primary Data Collection: An Introduction to Conclusive Research. In *International Business Research* (pp. 146-163). Routledge.

Nielsen, P., Winkler, T., & Nørbjerg, J. (2017). Closing the IT development-operations gap: The devops knowledge sharing framework. *CEUR Workshop Proceedings.*

Nybom, K., Smeds, J., & Porres, I. (2016, 5). On the Impact of Mixing Responsibilities Between Devs and Ops. *Lecture Notes in Business Information Processing, 251*, 131-143. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-33515-5_11

Olszewska, M., & Waldén, M. (2015, 9). DevOps meets formal modelling in high-criticality complex systems. *1st International Workshop on Quality-Aware DevOps, QUDOS 2015 - Proceedings*, 7-12.

Ohtsuki, M., Ohta, K., & Kakeshita, T. (2016, November). Software engineer education support system ALECSS utilizing DevOps tools. In *Proceedings of the 18th international conference on information integration and web-based applications and services* (pp. 209-213).

Paez, N. (2018, 12). Versioning Strategy for DevOps Implementations. *Congreso Argentino de Ciencias de la Informatica y Desarrollos de Investigacion, CACIDI 2018*.

Pandya, S. (2021, 6 8). *Common DevOps challenges and how to overcome them.* Retrieved from codemagic: https://blog.codemagic.io/devops-challenges-how-to-overcome-them/

Perera, P., Bandara, M., & Perera, I. (2016, 1). Evaluating the impact of DevOps practice in Sri Lankan software development organizations. *16th International Conference on Advances in ICT for Emerging Regions, ICTer 2016 - Conference Proceedings*, 281-287.

Perera, P., Silva, R., & Perera, I. (2017, 7). Improve software quality through practicing DevOps. *17th International Conference on Advances in ICT for Emerging Regions, ICTer 2017 - Proceedings, 2018-January*, 13-18. Retrieved from https://www.researchgate.net/publication/322515647_Improve_software_quality_through_practicing_DevOps

Plant, O. H., van Hillegersberg, J., & Aldea, A. (2021, September). How DevOps capabilities leverage firm competitive advantage: A systematic review of empirical evidence. In *2021 IEEE 23rd Conference on Business Informatics (CBI)* (Vol. 1, pp. 141-150). IEEE.

Punjabi, R., & Bajaj, R. (2016, May). User stories to user reality: a DevOps approach for the cloud. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)* (pp. 658-662). IEEE.

Puppet, D. (2020). *State of DevOps Report 2017*. Retrieved from Puppet: www.puppet.com/resources/report/state-of-DevOps report

Rafi, S., Yu, W., Akbar, M., Alsanad, A., & Gumaei, A. (2020). Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE. *IEEE Access, 8*, 105426-105446. Retrieved from https://www.researchgate.net/publication/341806881_Prioritization_based_taxonomy_of_DevOps_security_challenges_using_PROMETHEE

Rafi, S., Akbar, M. A., Yu, W., Alsanad, A., Gumaei, A., & Sarwar, M. U. (2022). Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis. *Applied Soft Computing*, *116*, 108377.

Rahman, A., Parnin, C., & Williams, L. (2019, 5). The Seven Sins: Security Smells in Infrastructure as Code Scripts. *Proceedings - International Conference on Software Engineering, 2019-May*, 164-175.

Ravichandran, A., Taylor, K., & Waterhouse, P. (2016). DevOps in the Ascendency. Berkeley.

Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L., Tiihonen, J., & Männistö, T. (2016). DevOps Adoption Benefits and Challenges in Practice: A Case Study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10027 LNCS*, 590-597. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-49094-6_44

Robson, G. (2019). *Monitor and Measure SLA Performance in Views and Reports*. Retrieved from Cloudset: https://support.cloudset.net/hc/en-us/articles/360003996577-Monitor-and-Measure-SLA-Performance-in-Views-and-Reports

Roche, J. (2019). Adopting DevOps Practices in Quality Assurance. *Communications of the ACM,* , 38-43.

Rowse, M., & Cohen, J. (2021, 1). A survey of DevOps in the South African software context. *Proceedings of the Annual Hawaii International Conference on System Sciences, 2020-January*.

Rzig, D. E., Hassan, F., & Kessentini, M. (2022). An empirical study on ML DevOps adoption trends, efforts, and benefits analysis. *Information and Software Technology*, *152*, 107037.

Sallin, M., Kropp, M., Anslow, C., Quilty, J., & Meier, A. (2021, 6). Measuring Software Delivery Performance Using the Four Key Metrics of DevOps. *In International Conference on Agile Software Development., 419 LNBIP*, 103-119.

Šćekić, M., Gazivoda, M., Šćepanović, S., & Nikolić, J. (2018, 7). Application of DevOps approach in developing business intelligence system in bank. *2018 7th Mediterranean Conference on Embedded Computing, MECO 2018 - Including ECYPS 2018, Proceedings*, 1-4.

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps capabilities, practices, and challenges: Insights from a case study. *In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering, Part F137700*, 57-67.

Shah, J., & Dubaria, D. (2019, 10). NetDevOps: A New Era Towards Networking DevOps. *2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2019*, 0775-0779.

Sharma, S. (2017). The DevOps adoption playbook: a guide to adopting DevOps in a multi-speed IT enterprise. John Wiley & Sons.

Simões Teixeira, D. (2019). *Maturity Model for DevOps*. Retrieved from https://repositorio.iscte-iul.pt/bitstream/10071/20297/1/Master_Daniel_Simoes_Teixeira.pdf

Singh, G. (2021). 15 DevOps Metrics | To Boost KPI's and Its Adoptions. Retrieved from https://www.xenonstack.com/blog/devops-metrics

Smeds, J., Nybom, K., & Porres, I. (2015). DevOps: A Definition and Perceived Adoption Impediments. *Lecture Notes in Business Information Processing, 212*, 166-177. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-18612-2_14

Snyder, B., & Curtis, B. (2017). Using Analytics to Guide Improvement during an Agile-DevOps Transformation. *IEEE Software, 35*(1), 78-83.

Snyder, H. (2019, 11). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research, 104*, 333-339.

Soni, M. (2016). End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery. *IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2016* (pp. 85-89). IEEE.

Spinellis, D. (2016, 5). Being a DevOps Developer. *IEEE Software, 33*(3), 4-5.

Stackpole, B. (2015). *5 disruptive technologies for 2016 | Computerworld*. Retrieved from Computerworld: https://www.computerworld.com/article/3010563/forecast-2016-5-disruptors-to-keep-on-your-radar.html

Steffens, A., Lichter, H., & Moscher, M. (2018). Towards Data-driven Continuous Compliance Testing. *CSE 2018: 3rd Workshop on Continuous Software Engineering @ SE18*. Retrieved from https://www.semanticscholar.org/paper/Towards-Data-driven-Continuous-Compliance-Testing-Steffens-Lichter/4ec4901463429ee2e9c45a99bdad5f7d55d459d9

Swersky, D. (2018). *What is MTTR? How to measure and improve your Mean Time to Recovery.* Retrieved from Raygun: https://raygun.com/blog/what-is-mttr/

Tanveer, S., & Peričić, T. (2019). Why systematic reviews matter. *Elsevier Connect*. Retrieved from https://www.elsevier.com/connect/authors-update/why-systematic-reviews-matter

Teixeira, D., Pereira, R., Henriques, T. A., Silva, M., & Faustino, J. (2020). A systematic literature review on DevOps capabilities and areas. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, *11*(3), 1-22.

Tomas, N., Li, J., & Huang, H. (2019, 6). An empirical study on culture, automation, measurement, and sharing of DevSecOps. *2019 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2019*.

Torble, T. (2019, 9). Change Management and Devops: Back to the Future. *ITNOW, 61*(3), 60-61. Retrieved from https://academic.oup.com/itnow/article/61/3/60/5552616

Trihinas, D., Pallis, G., Tryfonos, A., & Dikaiakos, M. (2018, 5). DevOps as a Service: Pushing the Boundaries of Microservice Adoption. *IEEE Internet Computing, 22*(3), 65-71. Retrieved from https://www.researchgate.net/publication/325708900_DevOps_as_a_Service_Pushing_the_Boundaries_of_Microservice_Adoption

Vaasanthi, R., & Prasanna, V. (2020, 4). Fundamental Challenges in Identifying Tools for DevOps Pipeline. *International Journal of Computer Applications, 176*(16), 36-38.

Veritis. (2019). *State of DevOps Report 2019: DevOps Not a Trend, The Standard!* Retrieved from Veritis: https://www.veritis.com/blog/state-of-devops-report-2019-devops-not-a-trend-the-standard

Vialle, G. (2018). *Monitoring to reduce Mean Time To Recovery (MTTR)*. Retrieved from Devoteam: https://nl.devoteam.com/expert-view/monitoring-to-reduce-mean-time-to-recovery-mttr/#:~:text=One%20of%20the%20goals%20of,measuring%20and%20reducing%20repair%20time.&text=You%20will%20find%20some%20reductions,out%20to%20be%20quite%20simple

Virtanen, A., Kuusinen, K., Leppänen, M., Luoto, A., Kilamo, T., & Mikkonen, T. (2017, 4). On continuous deployment maturity in customer projects. *Proceedings of the ACM Symposium on Applied Computing, Part F128005*, 1205-1212.

Wahaballa, A., Wahballa, O., Abdellatief, M., Xiong, H., & Qin, Z. (2015, 11). Toward unified DevOps model. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS, 2015-November*, 211-214.

Waller, J., Ehmke, N., & Hasselbring, W. (2015). Including Performance Benchmarks into Continuous Integration to Enable DevOps. *ACM SIGSOFT Software Engineering Notes, 40*(2), 1-4.

WATSON, M. (2017). *How to Measure Defect Escape Rate to Keep Bugs Out of Production.* Retrieved from Stackify: https://stackify.com/measure-defect-escape-rate/

Wettinger, J., Andrikopoulos, V., & Leymann, F. (2015). Automated capturing and systematic usage of DevOps knowledge for cloud applications. *Proceedings - 2015 IEEE International Conference on Cloud Engineering, IC2E 2015*, 60-65.

Wettinger, J., Breitenbucher, U., & Leymann, F. (2014, 1). Standards-based DevOps automation and integration using TOSCA. *Proceedings - 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, UCC 2014*, 59-68.

Wickramasinghe, S. (2021). *What Is DevOps? A Comprehensive Introduction – BMC Software | Blogs*. Retrieved from https://www.bmc.com/blogs/devops-basics-introduction/

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019, 7). Research for practice: the DevOps phenomenon. *Communications of the ACM, 62*(8), 44-49. Retrieved from https://dl.acm.org/doi/abs/10.1145/3331138

Wilson, N., Gross, J., & Bull, A. (2017). Towards Cultural Democracy: Promoting Cultural Capabilities For Everyone. In *Towards Cultural Democracy.* Retrieved from https://kclpure.kcl.ac.uk/portal/files/117457700/towards_cultural_democracy_2017_kcl.pdf

Wohlin, C. (2017). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *ACM International Conference Proceeding Series, 18th International Conference on Evaluation and Assessment in Software Engineering, EASE 2014. (ACM International Conference Proceeding Series, 2014).* Blekinge Institute of Technology. https://doi.org/10.1145/2601248.2601268

Xiao, Y., & Watson, M. (2019, 8). Guidance on Conducting a Systematic Literature Review:. *https://doi.org/10.1177/0739456X17723971, 39*(1), 93-112. Retrieved from https://journals.sagepub.com/doi/10.1177/0739456X17723971

Yin, L., & Filkov, V. (2020). Team Discussions and Dynamics During DevOps Tool Adoptions in OSS Projects | IEEE Conference Publication | IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/document/9285652

Yudovych, V. (2019, 6 25). *Important Metrics for Automation Testing - QATestLab Blog.* Retrieved from Qatestlab: https://blog.qatestlab.com/2019/06/25/automation-testing-metrics/

Zampetti, F., Scalabrino, S., Oliveto, R., Canfora, G., & Di Penta, M. (2017, 6). How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines. *IEEE International Working Conference on Mining Software Repositories*, 334-344.

Zarour, M., Alhammad, N., Alenezi, M., & Alsarayrah, K. (2019). A Research on DevOps Maturity Models. *International Journal of Recent Technology and Engineering (IJRTE), 8*(3). Retrieved from https://www.ijrte.org/wp-content/uploads/papers/v8i3/C6888098319.pdf

Zhao, J. (2014, 10). Combination of multiple databases is necessary for a valid systematic review. *International Orthopaedics 2014 38:12, 38*(12), 2639-2639. Retrieved from https://link.springer.com/article/10.1007/s00264-014-2556-y

Zheng, E., Gates-Idem, P., & Lavin, M. (2018, 4). Building a virtually air-gapped secure environment in AWS∗ with principles of devops security program and secure software delivery. *ACM International Conference Proceeding Series*. Retrieved from https://www.researchgate.net/publication/324460306_Building_a_virtually_air-gapped_secure_environment_in_AWS_with_principles_of_devops_security_program_and_secure_software_delivery

Zimmermann, O. (2017). Microservices tenets. *Computer Science-Research and Development*, *32*(3), 301-310.

Appendix

A List of abbreviations

CDD	Continuous Delivery and Deployment
CA	California
CD	Continuous Delivery
CI	Continuous Integration
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
DevOps	Developer & Operation
DMC	DevOps Maturity Calculator
FaaS	Function as a service
HPE	Hewlett Packard Enterprise
IaC	Infrastructure-as-Code
IoT	Internet of things
KPI	Key performance indicators
KSA	Knowledge, Skills and Abilities
MTBF	Mean Time between Failures
MTTD	Mean Time to Detect
MTTF	Mean Time to Failure
MTTR	Mean Time to Recovery
MTTR	Measuring Mean Time to Recover
OKR	Objectives and Key Results
PaaS	Platform as service
QA	Quality Assurance
SDM	Software delivery management
SE	Software Engineering
SLR	Systematic Literature Review
TOSCA	Standard Topology and Orchestration Specification for Cloud Applications
UDOM	Unified DevOps Model

DevOps Metrics Monitoring and Measuring Summary

Table 19

DevOps Metrics Measurement and Monitoring Summary

Article	Metric	Measuring method	Monitoring Method	Personnel Involved	Roles and Responsibilities	Advantages of the measuring method	Disadvantages of the measuring method
Measuring Deployment Frequency (Lawrence, 2021)	Deployment frequency. How often do the deploys happen? Daily	Daily or weeks based on demand. On-demand has multiple deploys per day. It is easily measured as a counter increment when a deployment happens.	Market analysis, cost Analysis, or feasibility study. This can be done by either pulling from the provider's pipeline or tracking via a dedicated step at the end of the pipeline.	Build Engineer. This is the professional who overlooks the quality and success of the product which has been developed and the time factor.	Automation orchestrator, deploying releases, software testing	It helps in knowing how well the team is equipped. Helps in registering outstanding performance. It helps the team know what to improve in order for them to keep growing.	It makes the team strain a lot where there are no quality tools. The software quality might be compromised.
Monitoring to reduce mean time to recovery (Vialle, 2018)	Meantime to recovery. This describes how quickly the teams can restore the services after an outrage.	It is measured in hours. This is measured as the time from when an impairment occurs to when it is resolved and then averaging the obtained values.	We use the SDM tool, a model to guess out of progress. Pager software such as VictorOps can help generate those values and monitor the progress.	Technical Support engineer. This professional ensures that it is well taken care of if a service outranges occur.	Researching, diagnosing, troubleshooting , and resolving customer issues	Customer satisfaction is guaranteed. Systems are well maintained.	It is costly. It requires much professionalism and hence is not favourable for small enterprises.
Metrics for measuring change management (Creasey & Hiatt, 2018)	Mean lead time for a change. This is the time from when the code is written to when the code is entering deployment.	In days, weeks, or even months. It measures the time required to implement, test, and deliver a single bit of functionality.	Several changes successfully in a period. It can be monitored through a clock that is started and then stopped after code enters production or using software such as source control data or issue tracker, which are	DevOps engineer. This is the professional tasked with managing and controlling the whole DevOps pipeline.	building and setting up new development tools and infrastructure, understanding the needs of stakeholders and conveying this to developers, working on	Leads to a more productive team. Leads to significant market share due to effectiveness in market delivery.	It is costly Requires expertise for faster deliverable hence unattainable easily.

Article	Metric	Measuring method	Monitoring Method	Personnel Involved	Roles and Responsibilities	Advantages of the measuring method	Disadvantages of the measuring method
			integrated with the development pipelines.		ways to automate and improve development and release processes, and testing and examining code written by others and analyzing results		
Measuring DevOps success: what, where, and how (Adam, 2022)	Change failure rate. What percentage of the changes made or deployment ? results in a service outrange?	It calculates the percentage of the number of changes that led to degraded services. This is measured as the total failed deploys to the total deploys made.	This is monitored by a monitoring software used to provide values for the deployment frequency and therefore gives the outranges in the deployment pipeline and mot in the outside world.	Technical Support engineer. This is the professional tasked with ensuring that the system is up and functioning well all the time.	research, diagnose, troubleshoot, and resolve customer issues in an accurate and timely manner	It helps in knowing how practical the application is. Measures the performance and skill set of the team	It is tedious since it is statistical. Needs expertise in order to conclude easily
Measuring and monitoring business hours spent on ticket SLA-3 (Robson, 2019)	Customer ticket	Taking a fraction of the number of tickets created to the number of customers active at that given time	Customer feedback, Use of quality monitoring tools.	Business Analyst	responsible for bridging the gap between IT and the business using data analytics to assess processes, determine requirements and deliver data-driven	It helps in getting the effectiveness of the developed application. It helps in getting the product acceptance ratio for the application	It is time-consuming since it requires collecting live data and varies in time.

Article	Metric	Measuring method	Monitoring Method	Personnel Involved	Roles and Responsibilities	Advantages of the measuring method	Disadvantages of the measuring method
					recommendations and reports to executives and stakeholder.		
How to measure defect escape rate to keep bugs out of production (WATSON, 2017)	Defect escape rate	Number of defects found in pre-production and comparing it to the production phase	Keeping a record of the number of defects and the units tested	System Admin	managing, troubleshooting, licensing, and updating hardware and software assets	Helps in identifying the capabilities of a team	It is resources intensive
11 test automation metrics and their pros & cons (Yudovych, 2019)	Automated test rate	Measured by the project-wise phenomenon	Integration of many tools, market analysis, or team discussion	Build Engineer	Automation orchestrator, deploying releases, software testing	It helps in improving the tools the team is using	It is costly
Measure application performance (Hodges, 2020)	Application performance	User satisfaction, average response time, error rates, and application and server CPU utilization	Observing the behaviour of the application, business analysis, alert and error data collection	System Admin	managing, troubleshooting, licensing, and updating hardware and software assets	It helps in suggesting improvements for the application. Helps in identifying the undesirable features of the application	It is time-consuming. It requires high analytical skills.

Article	Metric	Measuring method	Monitoring Method	Personnel Involved	Roles and Responsibilities	Advantages of the measuring method	Disadvantages of the measuring method
What is MTTR? How to measure and improve your Mean Time to Recovery (Swersky, 2018)	Meantime to detection	Adding all the incident detection times for a particular team member or period and dividing by the number of incidences	Comparing periods or team member values	Technical Support engineer	research, diagnose, troubleshoot, and resolve customer issues in an accurate and timely manner	It promotes individual hard work. It helps in instilling the spirit of teamwork in the workplace. It encourages quality development.	It is time-consuming. It requires high interpersonal skills.
Monitoring the Error Rate of a Spring (Hombergs, 2017)	Error rates	Total number of errors divided by the	Using applications such as Relic API,	Technical Support engineer	research, diagnose, troubleshoot, and resolve customer	Helping predict the performance of a product.	It is costly since it requires the usage of some

Literature matrix from the relevant publications:

Article	Metrics	Business Goals	Typical metric value ranges	What is the process to measure the metric?	How often is the metric measured?	Cost/Effort of metric monitoring
A Cambrian Explosion of DevOps Tools (Kersten M. , 2018)	Lead Time, Delivery Time, Service Level Agreements (SLAs), Customer Tickets	To increase software delivery productivity	Each metric ranges differently depending upon the integration of tools	Open-Source Tools such as Git & Rational ClearCase	Each metric ranges differently depending upon the integration of tools from hours to days and from days to weeks.	
The Path to DevOps (Dornenburg, 2018)	Cycle Time, Reliability & Throughput	Make Existing Process Faster & Reduce Cost of Execution	Depends on projects and organization size, can be measured daily or weekly	Through cross-functional highly skilled teams and networking software	It is usually Measured many times during a day or in a week	
A qualitative study of DevOps usage in practice (Erich et al., 2017)	Lead time	Increase Software Development Process Pace and Quality of Software	In days, weeks, or even months		High-performing teams typically measure lead times in hours, versus medium and low-performing teams who measure lead times in days, weeks, or even months	
Fundamental Challenges in Identifying Tools for DevOps Pipeline (Vaasanthi & Prasanna, 2020)	Deployment time and Integration	Right tool selection for building DevOps tool pipelines	Daily or Weekly basis or on demand	Market Analysis, Tool feasibility/desirability study, Cost Analysis		
Service Provider DevOps (John et al., 2017)	MTTR and change in failure rate	Service agility and elasticity	Ranges in hourly manner	SDM, (It is a model to take the guesswork out of progress)	It is typically measured in hours	
Evaluating the Impact of DevOps Practice (Perera, rt al., 2017)	Deployment Size, Time & Frequency	Deliver high quality products, satisfying the customer	It should be done and calculated separately for production and non-	Through Inferential statistical analysis based on various variables and respective	Successful Organization usually do on daily basis but even a weekly analysis	

Article	Metrics	Business Goals	Typical metric value ranges	What is the process to measure the metric?	How often is the metric measured?	Cost/Effort of metric monitoring
		needs	production in smaller rates and counts may vary depending on how large individuals work items are	indicators	can provide a better insight into which changes were most beneficial.	
Being a DevOps Developer (Spinellis, 2016)	Deployment Size and Time.	Enhanced Software Development.	Varies between hours, days, and weeks.	Control Interfaces and Scripted APIs.		
Methods of Implementation, Maturity Models and Definition of Roles in DevOps Frameworks: A Systematic Mapping (Aguiar Monteiro et al., 2020)	Continuous Integration, Delivery and Deployment.	Enhanced Software Development.		DevOps Maturity Models.		
Team Discussions and Dynamics During DevOps Tool Adoptions in OSS Projects (Yin & Filkov, 2020)	Automated tests	software quality, primarily by automating the processes of building, testing, and deploying software.	Project Wide Phenomenon.	Mainly through integration of multiple tools but also through Market Analysis and Team Discussions.		
An Empirical Taxonomy of DevOps in Practice (Macarthy & Bass, 2020)	Continuous Integration and Delivery.	bridge the gap between the conflicting priorities of the development and IT Ops teams.	Depends on DevOps Engineers and Ops Specialist.	automated pipelines on both physical and virtualized serve.	Varies due to expertise of Engineers and Ops Specialist.	
Measurement Based Performance Evaluation of DevOps (Batra & Jatain,	Deployment Frequency,	Quality Software and Customer Satisfaction	Usually done on daily basis and Demand		Daily Basis	

Article	Metrics	Business Goals	Typical metric value ranges	What is the process to measure the metric?	How often is the metric measured?	Cost/Effort of metric monitoring
2020)						
Towards Continuous Quality: Measuring and Evaluating Feature-Dependent Non-Functional Requirements in DevOps (Haindl & Plosch, 2019)	Application availability in non-functional requirements.	Specifying Software Quality with in functional and non-functional environments.		DevOps toolchains (e.g., a SonarQube plugin)		
Application of DevOps Approach in Developing Business Intelligence System in Bank (Šćekić et al., 2018)	Deployment Time.	Amplify agility, responsiveness, and faster time to market throughout the software delivery lifecycle.	Daily, weekly or on demand.		High performance teams can do it on demand and daily basis and low performance teams on weekly basis.	
Building a Collaborative Culture: A Grounded Theory of Well Succeeded DevOps Adoption in Practice (Luz et al., 2018)	Application usage & traffic and Availability.	Increase Maturity level of DevOps adoption.		Three Step Adoption Model.		
DevOps Making It Easy to Do the Right Thing (Callanan & Spillane, 2016)	Deployment Frequency,	Highly Improve the average release of cycle time.	Business wide implementation requirements.		Depends on projects and organizations size, can be measured daily or weekly.	
From Agile to DevOps: Smart Skills and Collaborations (Hemon et al., 2020)	Deployment Delivery/Delivery Time.	To align operational functions with development functions to meet market software needs.	Most organizations tend to analyze this on daily basis.	Through soft skills in DevOps automation level by highly skilled personnel.		

Article	Metrics	Business Goals	Typical metric value ranges	What is the process to measure the metric?	How often is the metric measured?	Cost/Effort of metric monitoring
DevOps in practice: A multiple case study of five companies (Lwakatare et al., 2016)	Deployment Frequency & Cycle Time.	web application and service development in small and medium sized companies to meet customers need.	It can range from days to weeks or dependent upon fixed projects schedules.	Languages and Tools varies from organization to organization depending on size and needs.	From daily to weekly based on production environment and within weeks or yearly for fixed schedule projects.	
LEADING DEVOPS PRACTICE AND PRINCIPLE ADOPTION (Maroukian & Gulliver, 2020)	Deployment Frequency, Size, Delivery and Service Level Agreements.	delivering value to the customer experience				
Research for Practice: The DevOps Phenomenon (Wiedemann et al., 2019)	Speed & Stability.	Customer satisfaction and profitability and team level outcomes including collaboration and work life balance.	Often weeks or days	Monitoring the key system metrics such as business or transactions metrics and other key performance indicators with fast feedback cycles and highly skilled cross functional teams.	Often weeks or days.	
DevOps Capabilities, Practices, and Challenges: Insights from a Case Study (Senapathi et al., 2018)	Deployment Frequency, mean time to recover and lead time.	To improve and optimize the service.	Depends on project need.	Monitoring services such as Datadog.	Based on project requirements and need	
NetDevOps: A New Era Towards Networking & DevOps (Shah & Dubaria, 2019)	Failed Deployment.	Increase revenue, cost reduction and cost avoidance.	Frequently during a day	CI/CD Build Servers and Pipelines including Sequential and Manual Infrastructure Provisioning and Snowflake Infrastructure with configurations.	Since it's a server-based monitoring so it's done on frequent basis.	
DevOps as a Service:	Deployment	modularity,		Unicorn Framework		

Article	Metrics	Business Goals	Typical metric value ranges	What is the process to measure the metric?	How often is the metric measured?	Cost/Effort of metric monitoring
Pushing the Boundaries of Micro service Adoption (Trihinas et al., 2018)	Frequency, Delivery Time, Availability	distribution, elasticity, and robustness of applications.				
Improve Software Quality through Practicing DevOps (Perera et al., Improve software quality through practicing DevOps, 2017)	Deployment Frequency and lead time.	Improving Software Quality	Usually on Daily Basis	performance engineers on testing teams focus on large-scale load tests in production-like environments	Multiple times a day for larger organizations.	
DevOps Team Structures: Characterization and Implications (Lopez-Fern ´ andez et al., 2021)	Lead time, Deployment Frequency and MTTR.	improve and accelerate the software delivery.	Usually done on daily basis	Platform servicing (e.g., CI/CD and realizing tools) and infrastructure (e.g., cloud infrastructure, virtualization or containerization, etc.)	Deployed in hourly manner every day.	
A Research on DevOps Maturity Models (Zarour et al., 2019)	Delivery Speed	Enhance the process of software development and delivery	Usually measured several times a day.	Using Maturity models such as Hewlett Packard Enterprise's and Capgemini's DevOps Maturity Mode.		
Service Provider DevOps for Large Scale Modern Network Services (Kim et al., 2015)	Deployment time, cost, frequency, and Quality Assurance	Agile service creation, and virtualized and software-defined network environment	Usually done on daily basis and on demand.	programmable interfaces monitoring.	Usually done on daily basis based on selection points through KPIs and KQIs.	
Versioning Strategy for DevOps Implementations (Paez, 2018)	Continuous Deployment Delivery			Version Control Strategy to support effective implementation of metrics and practices.		

Article	Metrics	Business Goals	Typical metric value ranges	What is the process to measure the metric?	How often is the metric measured?	Cost/Effort of metric monitoring
Innovative Practices for Knowledge Sharing in Large-Scale DevOps (Hemon et al., 2020)	Continuous Integration, Delivery and Agile	to meet the needs of a constantly changing software-development environment	Most organizations tend to analyze this on daily basis	Multi Functional Skilled Personnel.	Day to Day analysis	
DevOps Meets Formal Modelling in High-Criticality Complex Systems (Olszewska & Waldén, 2015)	Deployment Size & Software life cycle	Product Quality Improvement to avoid financial and human loss.		FormAgi Framework		
How is Performance Addressed in DevOps? A Survey on Industrial Practices (Bezemer et al., 2019)	Deployment Frequency and Delivery	Reduce the time between changing software and delivering these changes into production with high quality	Usually, to be done on regular basis	Application-level monitoring	Performance evaluation is not done on regular basis but mostly quarterly or yearly.	
Model-Driven Continuous Deployment for Quality DevOps (Artač et al., 2016)	Continuous Deployment	Quality awareness and Customer Satisfaction		Run time Quality Assurance Methods and feedback system.		

Note, this table is the summary of the SLR for DevOps metrics measurement and monitoring.