A Systematic Literature Review on DevOps and its Definitions, Adoptions, Benefits and Challenges

Gustaf Österberg

38864

Master's Thesis in Computer Engineering

Supervisor: Kristian Nybom

Åbo Akademi University

Faculty of Science and Engineering

Department of Information Technologies

April 2020

Abstract

The increased pace of software development and rapid changes of business- and technological requirements have introduced a new type of software development culture called DevOps. IT Operations and Software Development have endured poor communication and collaboration with each other, which have resulted in bottlenecks throughout the software development life cycle. The discrepancy has contributed to delayed software updates with inferior quality and a culture, that complicates the goals of organisations, to produce high-quality software for its customers. DevOps can streamline the software development process by removing the constraints on the teams, by increasing collaboration, automation, sharing and measurement principles. This is possible through the change of mindset and culture of the organisation. Since the concept of DevOps is loosely defined, it is an interesting topic to research. This thesis provides an insight on the definitions, implementations, benefits and challenges of DevOps, through a systematic literature review. The literature review was conducted by identifying and analysing the related literature on DevOps. The results were 25 primary studies on the definition, adoption methods, benefits and challenges of DevOps. The result presents ways to define DevOps, some tools and methods on how to adopt DevOps within an organisation and some perceived benefits and challenges of adopting DevOps. The contribution of the thesis provides an overall understanding of DevOps and its core principles and practices. The contribution also includes, approaches for adopting DevOps, the premise behind its adoption and obstacles that may occur during the process.

Table of Contents

List of Abbreviations

CD – Continuous Deployment

CDE – Continuous Delivery

CI – Continuous Integration

QA – Quality Assurance

RAD – Rapid Application Development

SDLC – Software Development Life Cycle

SLA – Service-Level Agreement

TPS – Toyota Production System

XP – eXtreme Programming

List of Figures

List of Tables

1 Introduction

Software development has previously been a slow and tedious process, generally adopting slow development methods such as waterfall and the v-model. These models have previously been used in large organisations, while in smaller organisations, more agile methods have been utilised to develop software. The smaller organisations have proven that by decreasing the release cycle of a project they can significantly reduce risks and overall development times. This is something that larger organisations have started to realise, and they also want to rake the benefits of faster release cycles. Since larger organisations usually have more tedious bureaucracy, they are limited from easily implementing agile principles into their development process [1].

In larger organisations the development process involves several teams, all with different goals and priorities. Development teams are responsible for creating the software and with an increasingly faster development pace, with changes and new features demanded to be released up to several times a day. Operations teams are used to tailor the software for the customer environment, which involves ensuring that the software runs smoothly without failures. Operations teams usually do not want frequent changes, since they increase the risk of failures within the production environment. Operations teams have therefore had problems to keep up with the faster development pace resulting in bottlenecks appearing between the teams [2].

These teams have generally not communicated very well nor worked well together, resulting in even further bottlenecking. These issues have resulted in long release cycles, up to several months, which impair the competitiveness of the organisation. More agile methods can cut release cycle times down to only a few hours resulting in better risk management, more satisfied customers and overall better competitiveness [2].

A way to tackle these problems is DevOps, which is the emergence of a new way of collaborating and releasing software. DevOps is coined from the names of two core teams, development and operations. The term does not have a clear definition and is portrayed as an extension of Agile methods [3]. DevOps uses certain practices, tools and automation to make the overall creation of software more efficient. By adopting DevOps, organisations try to tear down the figurative wall, that stands between developers and operation personnel. DevOps therefore puts emphasis on changing the culture of the organisation and increasing collaboration between the different teams within the software development process [2].

1.1 Research Motivation

The motivation for conducting this research is that DevOps lacks a clear definition. There is no clear model or framework for how to implement it and organisations might be unaware of the benefits and challenges it imposes [4] [3] [5]. DevOps is also a rather new concept, with the term emerging in 2009, and getting ground several years later [6]. The novelty of DevOps contributes to the fact that there is quite scarce amount of research conducted on the topic, with article databases only providing a few hundred results, when searching for “DevOps”. Nevertheless, DevOps can be considered a topic that organisations are quite interested in, which also provides some merit for conducting the research. According to [7], there are still several challenges for implementing DevOps:

> *Through 2022, 75% of DevOps initiatives will fail to meet expectations due to issues around organizational learning and change.*

This prediction gives some insight on the research problem and motivates continued research on DevOps. This thesis aims to provide a deeper understanding on the challenges, which the future DevOps initiatives probably must tackle. Furthermore, the State of DevOps Report of 2018, provides some insight on the importance of highly evolved organisations for adopting DevOps [8].

As stated before, there has been quite little academic research conducted into DevOps. The literature mainly discuss common principles and practises, while some researchers try to define DevOps. The problem is that organisations adopt DevOps differently, depending on their organisational structure, available tools, etcetera [9]. Defining a clear framework or method around DevOps is therefore challenging and most organisations adopt DevOps differently, with mixed success [8]. This thesis will provide some general insight into the emergence of DevOps, some key topics related to DevOps, and central concepts. The main contribution of the thesis is to provide different views on how

DevOps is defined, different adoption methods, and finally, benefits and challenges of DevOps.

1.2 Previous Research

There has been some prior research that adopted a similar research approach as this thesis. The prior research will be presented to provide an understanding on the premise of the research in the thesis and how the thesis might have been influenced by prior research.

The literature review [5], discuss the principles and benefits of DevOps, such as core principles like culture of collaboration, automation, measurement, sharing and Quality Assurance (QA). However, the literature review does not discuss DevOps adoption nor benefits or challenges very comprehensively. Therefore, this thesis aims to contribute to the existing literature, by reviewing a large number of studies, in order to get a broader view of different aspects of DevOps.

The systematic literature review [10], is conducted in a similar fashion as the study of this thesis. The systematic literature review discuss the definitions of DevOps, its practises, benefits and challenges. They try to define DevOps using the literature and provide some benefits and challenges of adopting DevOps. The conclusions were that there are no definitive framework or process to adapt DevOps, nor a clear definition of DevOps. Nevertheless, the review provides an overview of DevOps, which is similar to this thesis.

Lastly, the multivocal literature review [11], provides definitions, practices, benefits and challenges on DevOps. The summary of the findings were that DevOps is a cultural change with a set of practices, such as changing the responsibilities between development and operations, automating the software development and continuously monitoring the software. Benefits were realised as improvements of release cycle times, Continuous Deployment and knowledge and skills. The challenges were defined as problems to automate and high demand for skills.

2 Key topics

The Software Development Lifecycle (SDLC) is a systematic approach to software development. The SDLC is a process which contains phases vital for developers, such as planning, analysis, design, implementation and maintenance, as shown in Figure 1. The need for such a model exists, since creating software today is a very rigorous process, requiring a vast number of stakeholders and resources. With so many collaborators, the risks of projects have increased, since they all want their own agenda fulfilled. The projects are also vastly different and require different approaches to the SDLC, which are called software development models. Some examples of such models are: The Waterfall Model, The V-Model, The Spiral Model, Rapid Application Development, and others [12]. The models range from very non-iterative models like Waterfall to very iterative models like Agile [1]. DevOps can be seen as a further step beyond Agile and is by some research defined as an extension to Agile [3].

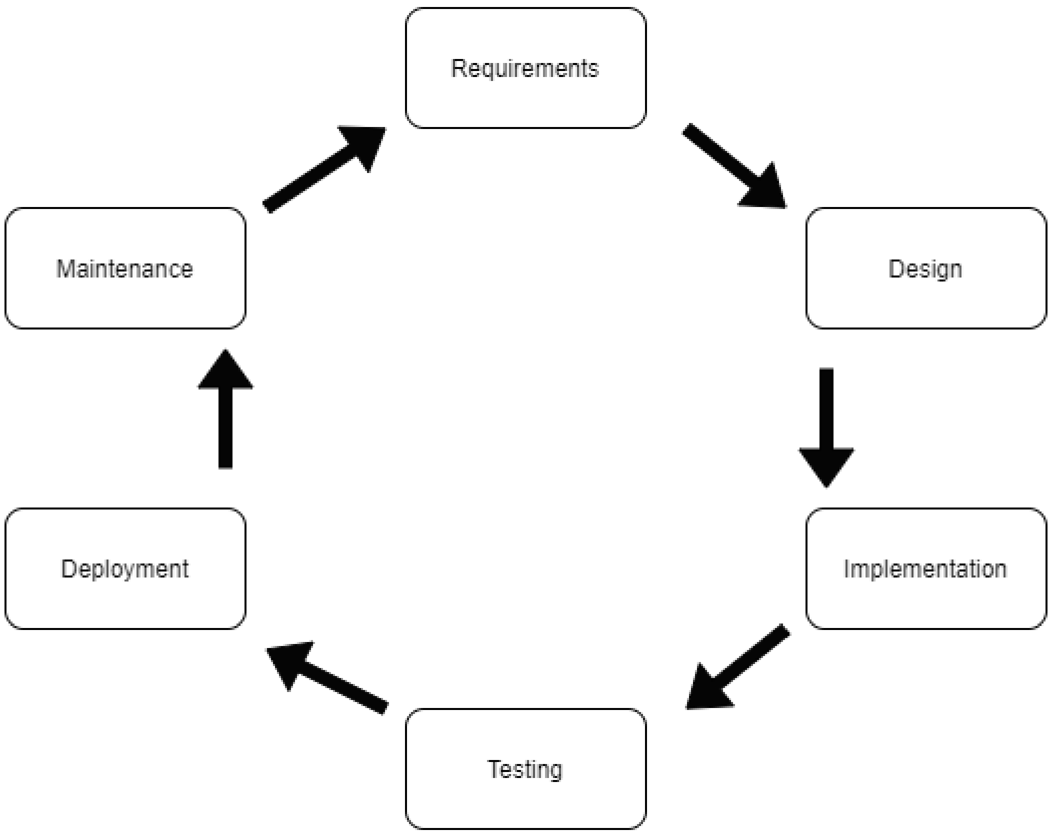

Figure 1. The Software Development Lifecycle, where the different phases of software development ties together to potentially form an infinite loop. Partially adopted from [1].

2.1.1 The Waterfall Model

The Waterfall Model has been in use since Dr Winston Royce published his article "Managing the development of large software systems" [13]. The model portrays the different steps of software development in a downwards fashion, shown in Figure 2, which is why the model is named The Waterfall Model. The article highlights some key properties that The Waterfall Model emphasises on. Communication via documentation is the most important part of the model, which ensures that no one has a significant impact on the project without it being written down. Similar to an extensive documentation the project has to be designed and analysed thoroughly in the beginning of the project to reduce the risk of errors. Subsequently, each step has to be completed as thoroughly to reduce the risks of problems surfacing in the later stages, when they are more difficult to resolve. Each step of the process has milestones specified to ensure the completion of the product. This worked quite well when the systems were developed in a simple domain where cause equals effect i.e. the domain was subject to little change. In the following sections, more iterative models are presented [1].

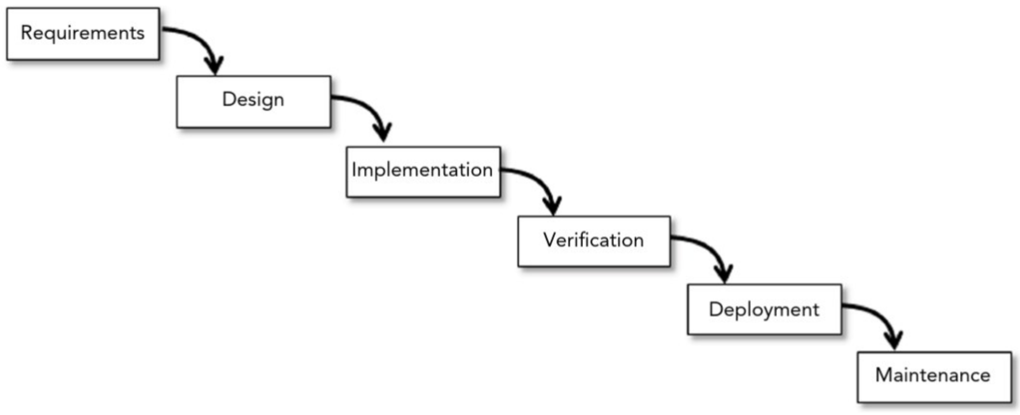

Figure 2. The Waterfall Model, which portrays the software development process in its different steps, illustrated like a waterfall [1].

2.1.2 Agile software development

Some of the Agile thinking has been around since the birth of the Lean Toyota Production System (TPS), developed by Toyota in the 1960s. While the TPS focuses on Lean manufacturing of products, on a production line, Agile thinking is fixed on Lean product development. Agile thinking is therefore more reliant on innovative and creative development environments, since these are core principles of product development. During the 1990s, many frameworks were developed, utilising Agile thinking. The Rapid Application Development (RAD) framework was developed during the mid-1990s, focusing on creating prototypes and developing through an adaptive process, with less emphasis on planning and documentation. Later in the 1990s, the eXtreme Programming (XP) framework was created, which focuses on the technical values and practices of software development. XP also emphasis on pair programming, extensive code review and unit testing. Following these new frameworks, the software development domain has shifted from slow and non-iterative development methods, such as Waterfall and the V-Model, to these faster and more iterative Agile frameworks [14]. A proper definition of Agile software development was first introduced in 2001 through the Agile Manifesto [15], which explains the principles of agile development. The manifesto brings forward a culture where the customer is in the focus and where change is welcomed.

The Agile manifesto brings forward four main values of Agile software development. Firstly, Individuals and interaction over processes and tools, which describes how motivated and enabled people are the factor that brings the most value to the customer. Nevertheless, people need excellent tools and processes to enhance the capability to deliver superior value. Secondly, the Agile manifesto values working software over comprehensive documentation. Working software brings added value directly to the customer and proper documentation which delivers value to the stakeholders. Thirdly, customer collaboration over contract negotiation, which means that collaboration between the customer and the development team is crucial. The teams must understand the business side of the project to be able to assess the time frame and business need of the project. Furthermore, the business side must have knowledge about the product and its technology to ensure that there is no misunderstanding when assessing the functionality and requirements of the product. Fourthly, responding to change over following a plan, which means that change has to be embraced and should be made the main part of the development process, instead of treating it as an external threat that has to be combated. There are a few models describing change, which are discussed in [1]. The first model showcases 11 steps of leadership, for example, build close relationships with staff, be tolerant, trust one's staff, be a visionary, express one's feelings, be dynamic, and be humble. The second model discusses how to take advantage of provided opportunities, how change should be achieved and how to strengthen the information stream throughout the organisation. Lastly, the J-curve change model presents five stages that describe behaviour patterns that organisations express [14].

2.1.2.1 Scrum

The goal of agile development methods is to increase the development pace, communication, improve reactivity and reduce risks of software development. Scrum focuses on completing a small number of tasks within a small timeframe, to ensure the full completion of each task before tackling the next ones. Therefore, the main difference between traditional software development methods and Scrum is that the planning horizon for a project is usually defined in weeks, instead of months or years. These few weeks long development cycles are called sprints. The sprint can be seen as a short-ranged schedule, with its own tasks and goals. The tasks of the sprint are defined in the sprint backlog, which contains those tasks that are meant to be completed during the sprint. The goal of the sprint is to have the defined tasks fully implemented with a fully working product or prototype by the end of the sprint. This provides a way for the development team to receive feedback on the working software after each sprint and to reduce the risks and misunderstandings that may arise during the project. The tasks are mainly derived from requirements defined at the beginning of the project. In scrum, these requirements are usually called user stories, which are used to present the way the software is meant to function. User stories are usually portrayed on a story board with different areas depending on the state the user story is in, i.e. backlog, sprint, testing or completed. The amount of work a user story will require can be described by using story points [16].

2.1.3 Lean software development

The Waterfall model comes with its flaws and is not suitable for every project. Practitioners wanted to find alternative methods, that would better suit software development. The solution was found in the Lean software development methods, which was introduced by Toyota in the 1960s. The main principles of Lean software development are to eliminate waste, respect the team, deliver quickly and with quality, defer commitment and building knowledge. Waste elimination refers to clarifying requirements and eliminating unnecessary features of the project. Useless repetition and bureaucracy are also to be eliminated. The project team needs respect to work efficiently and the members should be able to work in a cross-functional manner to make the use of resources more efficient. Decision making has to be made in an intelligent fashion. A new feature should be implemented at a later stage of the project, if several factors are still unclear. It is also important to take the customers' needs into account, before making extensive investments into a project. Lastly, Lean development methods emphasise short development iterations to provide value to customers in a fast manner [1].

2.1.3.1 Kanban

The Kanban system originates in the TPS, which is a type of lean manufacturing system developed by Toyota in the 1960s (discussed in section 3). The idea of Kanban was introduced during the beginning of the 21st century and further defined by David J. Anderson in 2010 [17]. In his book, Anderson defines six core practices of Kanban. The key is to make the workflow visible, by using boards and post-it notes, as shown in Figure 3. Kanban limit the amount of work (WiP limit) performed at a given time, by using quantitative limits on the number of ongoing tasks. These limitations on the number of tasks worked on at a given time, reduces the number of failed deliveries, which keeps customer relations on a high level. Furthermore, by following these two practises, organisations can more easily detect bottlenecks and improve communication by identifying the stakeholders that are tied to a specific task. The third practise is to manage the workflow by resolving current issues before tackling new tasks. It is also important to validate whether or not the new feature or change produces the desired value and lives up to the goals set in the planning stage. The goal of Kanban is to have a fast and continuous workflow. To manage the workflow, tasks are categorised into different levels of urgency, which are defined by the Service-Level Agreement (SLA), specified in agreement with the customer. Daily stand-up meetings are held to meet the SLAs at the desired time and to enhance the communication between the team members. Therefore, feedback is a vital practise of Kanban. The daily stand-up meetings should be conducted in the teams but also on a larger scale, throughout the entire value chain to provide insight into the current work situation. Kanban also emphasises that policies must be definitive and clear to all of the stakeholders. If a policy does not work, it has to be changed accordingly. Lastly, the use of methods and models provides better collaborative improvements. It is not necessary to reinvent the wheel, when already proven solutions can be reused to ensure the added value of the practise. The Kanban core practises distance themselves from strict prescriptions and set rules by implementing a loose but adaptive workflow, with positive restrictions tied to the workload and priorities [18].

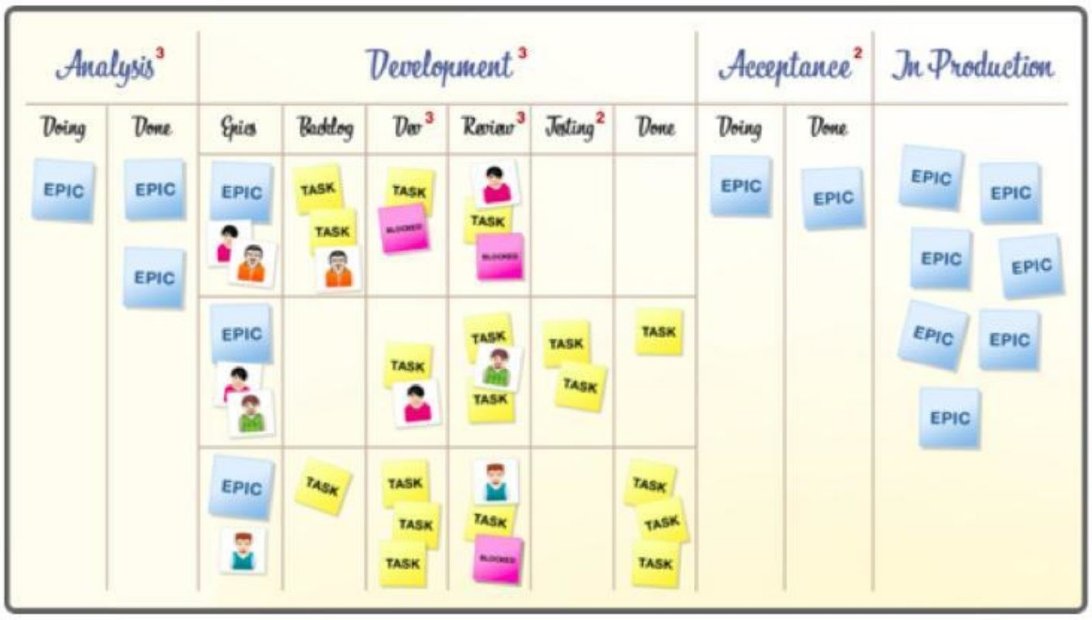

Figure 3. An example of a Kanban Board [19].

3 Central concepts in DevOps

The concepts behind DevOps, such as lean development, had been used previously for many years. A prime example is the TPS, which was developed by Toyota's chief of production in the 1960s. The TPS had three main principles, keeping inventory low, minimising the que of orders and maximising the efficiency of the manufacturing process [20].

In 2001, Agile was introduced by a group of 17 technology leaders, such as Alistair Cockburn and Martin Fowler. Together they created a manifesto, "The Agile Manifesto" that described agile principles for software development. The main argument for the movement was to eliminate slow and documentation-oriented software development methods. The goal was to make these slow methods more iterative and agile with the customer and end-user in focus. They wanted to remove requirement documentation and large deadlines and more focus on using the working software itself as a way to measure progress [21]. These Agile principles were still not enough for some agile practitioners.

Patrick Debois, who is a consult and passionate about agile development, realised the mismatch between the IT operations and software development teams. In 2008, he discussed his frustrations with another agile practitioner, Andrew Schafer. Together, they shared their issues on software development and the discrepancy between the IT operations and software development teams. In 2009, Debois organised the conference "Devopsdays" in Belgium, where the term DevOps was used for the first time. He gained confidence to act, after he had listened to the now famous "10+ Deploys per Day: Dev and Ops Cooperation at Flickr.", a presentation given by two Flickr employees at the O´Reilly Velocity conference earlier in 2009. This presentation is seen as the important moment when the methods, now known as DevOps, first gained their ground [22].

A few years later in 2011 the DevOps movement started to build open-source tools by using the virtual deployment environment Vagrant. This was to implement the philosophies created during the years prior. In 2012, IBM joined the DevOps market by offering consulting services for adopting DevOps [2]. A year after in 2013, IBM invested further by acquiring UrbanCode, a company that helps organisations enable

Continuous Delivery in the cloud, on-premise and in mainframe applications. Later the same year, the author Gene Kim released “The Phoenix Project”. With the book, Gene Kim brought DevOps into the mainstream of software development. He has also been proficient within the DevOps community by releasing the yearly “State of DevOps Report” together with Jez Humble and Puppet Labs. The report is built around data received from surveys performed by technical professionals from around the world [21] [23] [24].

DevOps is a portmanteau of the two teams, Development (Dev) and Operations (Ops). The Development team is the entity that create code, test the code and provide QA on the developed code and software. They are often the teams that business personnel are dependent on to create software in a fast-paced environment. This results in the fact that software developers are forced to produce their code very fast [25]. The problems occur when the other entity, the Operations team, is involved. The Operations unit is responsible for maintaining the live software in the production environment, that the customer is continuously using. The operations team has to ensure that the software is running without problems and they are not keen on updating and changing the running software as often as the developers are producing new features. This results in a bottleneck between developers and operations personnel. The problem is intensified when there is poor or lack of communication and collaboration between the two units [25].

3.1 DevOps practises

DevOps practices are used to bring development and operations into alignment to better be able to deliver quality software in a fast manner. According to [26], the main objective of DevOps is to achieve Continuous Deployment, which is possible by utilising DevOps practises. Therefore, the core DevOps practices will be presented in this section.

3.1.1 Continuous Planning

Planning is part of the SDLC and is usually conducted in the beginning of the project. Planning can be seen as a precondition to continuous practices. Therefore, the success of the whole project can depend on the planning phase and inadequate planning in the development process is usually a cause of the failure of a project [27]. The problem with planning, within software development, has been the lack of flexibility and agility. Planning has been disjointed and divided into planning segments, each occurring for an extensive amount of time. Therefore, planning also require Agile-like methods to better synergise with the faster paced and more customer-oriented development standards [27]. Continuous planning provides better synergy between the business side and the developers. With continuous planning, the plan-cycles are much shorter, which provides the developers with smaller tasks, which also provide working software faster to the customer. The customer is able to provide feedback of the product, which will be reviewed and used in the next planning phase. The planning-phase is therefore brought to a similar level of agility and flexibility, as the rest of the SDLC, resulting in a more quickly produced product, with higher quality [28].

3.1.2 Continuous Monitoring

Development teams ensure the stability and functionality of their applications by continuously monitoring them in the production environment. The monitoring is used to assure success of the deployment process, how the application is running once deployed and to receive knowledge of the usage behaviour of the system. The automation of the monitoring process is enabled by using monitoring tools, such as Graylog, Kinesis and New Relic, which can also send automated notifications to team members, in case of errors. The data from the monitoring process is used in graphs and dashboards to display trends and patterns of the stability of the system [29].

3.1.3 Continuous Deployment & Continuous Delivery

When delivering new changes to production, the practice, in the scope of DevOps is called Continuous Delivery (CDE). The commonly used term describes the iterative delivery of software, in short cycle times to make sure that the software is always in a deployable state. The deployment process is validated through automated tests and quality checks. CDE is the method of delivering changes to production for manual approval. In CDE, the code is not continuously deployed to production like in Continuous Deployment (CD) [30]. The practice of CD is used when an organisation wants to continuously and automatically deploy changes to the production environment. According to [30] [31], the CD practice is an extension to CDE and is considered more challenging to adopt that CDE. Furthermore, CD practices are supported by the usage of the CI/CD Pipeline (see Section 5.2.1), which enables the change to be automatically built, tested, configured and deployed.

3.1.4 Continuous Integration

Within software development the process, which is responsible for running code, running unit tests, validating code and checking compliance is called Continuous Integration (CI). CI is the practice where team members of software development integrate their work on a frequent basis, usually a few times per day. The CI process is in most cases automated and is considered a central part of DevOps. The goal of CI is to bridge the gap between developers and operations personnel by using automation in the building and testing of software [31]. Martin Fowler defines CI as follows:

> *Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.* [32]

Fowler continues to define some key practices of CI, such as automation, single source repositories, that everyone should commit code to the primary code branch and that broken builds should be fixed immediately. Automation can be used for several parts of the continuous integration process, such as build-, test- and deployment automation. Code repositories are used to store code, test scripts, database schemas, properties files and install scripts. It is important that everyone has access to the repository and that they all store the essential files used within CI. Fowler continues to describe the benefits of CI and the core benefit of CI is reduced risk and a reduced number of errors in the system. Risk is reduced since CI makes the whole integration process is very predictable and the time it takes to integrate is also low, due to automation. Errors in the code (bugs), are easier to find, due to automated tests, which identify errors as they are written instead of later in the process, when they are harder to resolve [32].

3.1.5 Continuous Testing

Testing has always been a part of the development process of software. Testing the software comes with a cost, since the time it takes to create the tests, run them and evaluate the result, reduces the time available for producing more code. Nevertheless, testing is a very important part of the development process and the extensiveness of the bugs are likely increased the further testing is postponed. The solution for this problem can be Continuous Testing, which is testing that is performed in the background of the developer's computer. The tests, often regression tests, are performed continuously while the developer writes code. This is done to prohibit the new code from affecting existing functionality and to prevent new trivial bugs from appearing into the new feature or functionality. The continuous testing works by keeping the version of the code being tested, in synchronisation with the code being edited. The continuous test suite can then provide feedback continuously without any input from the developer, since the developer does not have to think about when to run his tests. Continuous testing is said to reduce development times by up to 10-15% for single-developer projects [33].

3.1.6 Automation

When implementing DevOps, a key characteristic is the highly reduced release cycle times. To achieve this, some parts of the SDLC have to be automated. A pipeline (see

CI/CD Pipeline) is usually used where software code goes through a set of stages, where it is tested and evaluated. If the code fulfils the pre-set requirements it will advance to the next stage in the pipeline. These tests and evaluations are done automatically, which enhances the need of proper automation. When the software code passes all stages, it is ready to be deployed into production, which also might be automated, if continuous deployment is used. By using this type of pipeline, with its automated steps, organisations achieve a more iterative and agile approach to software development [34]. As previously stated, automation is used all over the SDLC and according to [35], there are several technological enablers of DevOps that utilise automation. Some of the technological enablers are build-, test-, deployment-, monitoring-, recovering- and infrastructure automation. The different types of automation practices can be used to streamline the development process, while letting the employees focus on research and development tasks instead of monotone testing and build tasks.

3.2 DevOps Culture

The concept of organisational culture was first introduced in the 1950s and further defined by researchers such as Geert Hofstede and Edgar Schein. Organisational culture is the idea that a group of people work together, with shared values and behaviours. The culture also influences how members of the culture react to changes of the culture. In the scope of DevOps, culture plays a major role and is seen as a core part of the DevOps movement. This implies that DevOps cannot be implemented by starting to use a set of tools or workflows. The correct organisational culture is mandatory for a successful DevOps adoption [4] [36] [37] [38]. The first step of implementing a DevOps culture is to remove the concept of having separate development and operations teams. This provides a way for the teams to work towards common goals, without hindering each other's work. This is not something that can be changed in an instant and requires the cooperation of the whole organisation. The challenge can be to convince senior management that the culture might have to be changed in order to fully rake the benefits of DevOps [39]. According to [38], there are some key cultural characteristics that define the DevOps culture. Open communication is the backbone of a DevOps culture. Therefore, communication practises such as ticketing systems, rigid request procedures and a general siloed mentality are considered to be a detriment to a successful DevOps implementation. It is far more important to discuss and develop the product throughout its lifecycle by reviewing its requirements, resources, features and schedule. The product should be in focus and different metrics related to the production environment and the build sequence should be available to everyone. Further, responsibility and motivation to create the best product, is another part of the DevOps culture. Developers and operations personnel should be rewarded for creating a good product, not by number of lines of code or by successful deployments. Subsequently, developers must have a mindset of responsibility and proudness of the code they create, to the extent that they want to supervise its correct functionality. For a collaborative culture to exist, respect should be shown to all team members and other teams. The contributions of others should be recognized, while respectful discussions and the ability to teach and learn from others is vital for everyone's learning experience. Finally, the development-, operations-, QA- and management teams have to trust each other's abilities to create a successful product. If the teams do not trust each other, the implemented DevOps practises will not perform to their full potential.

4 Systematic Literature Review

This chapter presents the protocol used to conduct the systematic literature review. The protocol can be utilised to reproduce the study and provides an insight into the validity of the research. The protocol describes the premise of the research, the method of retrieving the literature, the criteria of selecting the literature, how data was extracted from the primary studies and how the data was interpreted. The purpose of the review was to gain insight into what DevOps is, how it can be utilised for software development and in what ways the implementation affects the organisation.

4.1 Literature Review

A. Research Questions

The following research questions (RQs) were defined:

RQ1. How is "DevOps" defined?
RQ2. How to adopt DevOps in a company?
RQ3. What are the benefits of DevOps?
RQ4. What are the challenges of DevOps?

The objective of the first research question (RQ1) is to provide a clear definition of DevOps, the characteristics of DevOps and its practices. The second research question (RQ2) has the goal of defining the required steps organisations must take to successfully adopt DevOps. Research question three (RQ3) aims at highlighting the perceived benefits of adopting DevOps in an organisation. The last research question (RQ4) presents challenges and shortcomings of adopting DevOps in organisations. This question is limited to the direct challenges of adopting DevOps itself, not challenges that might exist prior to adopting DevOps.

B. Strategy to find primary studies

Based on the research questions, search strings were defined to find the most relevant prior studies on the topic. The somewhat trivial search strings were selected, since existing research on DevOps is quite scarce, which resulted in a relatively low number of search hits.

I) Search Strings: The search was conducted using the search strings in Table 1.

Table 1. Search strings used.

#	**Search string**
1	"DevOps" AND "adopt*"
2	"DevOps" AND "benefit*"
3	"DevOps" AND "challenge*"
4	"DevOps" AND "defin*"

II) Databases: Three databases were selected for the study:

- ACM Digital Library
- IEEE Xplore
- Science Direct

The search strings were applied to the different search methods of each digital library. Duplicates were automatically removed from the collected results.

C. Inclusion Criteria for Primary Studies

The inclusion criteria for the primary studies were as follows:

- The article was written in English.
- The article was published in a journal, conference proceeding, conference workshop.
- The article discussed some or all of the defined research questions.

D. Title and abstract screening

The inclusion criteria were applied during the title and abstract screening process. Since the whole study was performed by only one researcher, the title and abstract screening was performed once and by one person. This somewhat increased the researcher bias of the title and abstract screening process. Since the articles that were found were on a manageable level, the abstract and title screening was performed concurrently.

E. Full Text Screening

To find the most suited articles for the primary study, a full text screening process was completed. The inclusion criteria as defined above, were applied while conducting the full text screening. The articles that did not contain relevant information to the study were excluded. The full text screening was performed by one researcher, which might have reduced the validity of the screening process.

F. Quality Assessment of the Primary Studies

The quality assessment of the study was performed by checking whether the selected primary studies from the previous phase meet the minimum quality requirements of the study. If the paper did not meet the requirement, it was excluded from the study. The studies that passed this quality assessment are the final papers in the primary study. This part of the research was also performed by one researcher, which might increase the threat to the validity of the study.

The checklist used as a reference for the quality assessment is shown in Table 2. Each statement is evaluated on a three-level numeric scale, the levels being: yes (2 points), partial (1 point) and no (0 points). With 14 questions in the checklist, the maximum number of points a study could receive were 28 points and the minimum of 0 points. The articles had to receive a fourth of the maximum points (28/4 = 7) to be included in the final primary studies. Therefore, an article that received 7 or less points was excluded from the research for having lacklustre quality, with this study in mind. It is important to note that the articles were assessed on the premise of relevance to this study. Even a very well-written article could have been excluded, if it did not have the desired relevance in regard to this study. The checklist was designed to find relevant articles to this study, and the aim was not to rank the articles depending on the scored points. The goal was simply to exclude those articles that did not contribute enough towards the research.

Table 2. Checklist defining the quality of the primary studies. Adopted from [40].

#	Question
Theoretical contribution	
1	Is at least one of the research questions addressed?
2	Was the study designed to address some of the research questions?
3	Is a problem description for the research explicitly provided?
4	Is the problem description for the research supported by references to other work?
5	Are the contributions of the research clearly described?
6	Is there sufficient evidence to support the claims of the research?
Experimental Contribution	
7	Is the research design, or the way the research was organized, clearly described?
8	Is an empirical study presented?
9	Is the experimental setup clearly described?
10	Are results from multiple different experiments included?
11	Are the experimental results compared with other approaches?
12	Are negative results, if any, presented?
13	Are the limitations or threats to validity clearly stated?
14	Are the links between data, interpretation and conclusions clear?

G. Data extraction

The data extraction was performed by using the data extraction form shown in Table 3. The primary studies were analysed with the predefined research questions in mind. For every primary study, the most relevant keywords were extracted for each research question. The keywords were used to divide the research questions into categories, shown in Table 5. The data extraction was performed by one researcher, which might affect the validity.

Table 3. Data Extraction Form. Partially adopted from [40].

Data Item	**Value**	**Notes**
General		
Data extractor name		
Data extraction date		
Study identifier		
Title, authors, year, journal/conference/workshop		
Author affiliations and countries		
Publication type (journal, conference or workshop)		
DevOps related		
RQ1: How is “DevOps” defined		
RQ2: Tips for adopting DevOps in a company		
RQ3: What are the benefits of DevOps?		
RQ4: What are the challenges of DevOps?		

5 Results

In this section the results of the study is presented. The search was performed on June 26, 2019, using the search strings mentioned (see Section 4.1.B). The search yielded 554 articles. Based on the initial abstract and title screening process there were 77 articles selected for full text screening. During the full text screening, 34 articles were selected for further analysis. These articles provided a basis for answering the specified research questions. Nevertheless, they still had to be further assessed based on their theoretical- and experimental contribution. The quality assessment (see Table 2) was performed and a set of 25 primary studies were selected for the final study, shown in Table 4.

Table 4. The primary studies included, with the research questions they provided answers for.

Study Identifier	**Reference**	**RQs Answered**
S1	[3]	RQ1 & RQ2
S2	[41]	RQ2 & RQ4
S3	[42]	RQ1
S4	[29]	RQ1 & RQ2 & RQ3 & RQ4
S5	[26]	RQ1 & RQ2 & RQ3 & RQ4
S6	[4]	RQ1 & RQ3 & RQ4
S7	[36]	RQ2
S8	[31]	RQ2
S9	[43]	RQ1 & RQ2
S10	[44]	RQ1 & RQ2
S11	[45]	RQ2
S12	[46]	RQ2
S13	[47]	RQ1 & RQ3 & RQ4
S14	[48]	RQ2 & RQ4
S15	[49]	RQ4
S16	[50]	RQ2
S17	[51]	RQ4
S18	[52]	RQ1
S19	[53]	RQ4
S20	[30]	RQ4
S21	[54]	RQ3
S22	[55]	RQ2 & RQ4
S23	[56]	RQ2 & RQ3 & RQ4
S24	[35]	RQ1 & RQ2
S25	[57]	RQ1 & RQ2

The data provided through the data extraction process was categorised depending on theme and research question. Three themes were identified for each research question. The themes were selected based on the impact they made for answering the research questions and the backing they received from the primary literature. The generated themes are shown in Table 5.

Table 5. The identified themes, and the primary studies discussing them.

Theme	**Study Identifier**									
Definition										
Collaboration	S1	S6	S7	S8	S9	S10	S12	S16	S18	S24
Ownership and Responsibility	S4	S5	S7	S9	S10	S24	S25			
Principles of DevOps	S3	S6	S9	S13	S18					
DevOps Adoption										
CI/CD pipeline	S5	S12	S13	S23						
Monitoring	S5	S7	S8	S11	S12	S23	S24			
Tools	S2	S4	S7	S10	S14	S16	S19	S23	S25	
Benefits										
Faster release cycles	S4	S5	S6	S13	S21	S23				
Higher productivity	S4	S5	S6	S21						
Quality	S4	S5	S6	S13	S23					
Challenges										
Lack of skills and knowledge	S4	S5	S13	S15	S17	S23	S14			
Resistance to change	S5	S6	S13	S15	S17	S14				
Scarcity and cost of tools	S2	S5	S6	S17	S19	S20	S23			

5.1 DevOps definition (RQ1)

It is clear that there is no unified definition of DevOps in published research. There are some attempts to try to define the concept. Nevertheless, there are still not enough research to support a certain way of defining DevOps. In this section, key defining topics on DevOps will be provided.

5.1.1 Collaboration

While conducting the literature review, eleven primary studies related to collaboration were found [S1, S6-S10, S12, S16, S18, S23, S24]. All of the primary studies related to collaboration, emphasise on the fact that collaboration between the development teams, operations teams and other teams involved in the SDLC is important for DevOps.

The primary studies [S1, S7, S9] list collaboration as a core category or component of a successful DevOps implementation. Collaboration results in better synergy between the development and operations teams and is often discussed in junction with culture, metrics and sharing of responsibilities. Collaboration is a way for development and operations teams to foster DevOps adoption, resolve the lack of communication between the teams and provide a means to adopt DevOps principles [S6-S9].

According to [S8, S10, S12], collaboration is key to enable CD and to reduce the release cycle of the development process. This requires collaboration throughout the SDLC, including the developers, operations personnel, testers and QA personnel. The teams must only collaborate, it is not required for the different teams to do both operations- and development tasks.

The primary articles [S16, S18], provide a few definitions of DevOps and collaboration is a key part in most of them. This provides some ground to the fact that collaboration is a vital part of the definition of DevOps. Lastly, the primary study [S24] presents cultural enablers for DevOps, such as shared goals, shared ways of working, shared values, responsibility and collective ownership. These enablers require a substantial amount of collaboration to function properly.

5.1.2 Ownership and Shared Responsibility

Many researchers conclude that ownership of code and shared responsibility between teams are key defining characteristics of DevOps [S4, S5, S7, S9, S10, S24, S25]. According to [S4, S5], ownership and responsibility are required to achieve the desired steps within the SDLC. Development teams should take full responsibility for their software and carry out changes fast, while the operations team assists with automation, security knowledge, scalability and performance. Furthermore, the developers that create the software should also be responsible for it. This is also the case after the software has been deployed into production, developers cannot just hand over the completed work to the operations team, to then forget about it.

The primary article [S5], also discusses ownership and responsibility and describes DevOps as a journey, where developers not only hand over their finished work to system administrators, but also show responsibility for their work and communicate with the administrators to ensure a collaborative handover. Development teams should also be responsible for writing infrastructure scripts and partake in the monitoring of the system after its deployment.

According to [S10], some research groups have started to adopt DevOps practises. The lack of a QA team within research groups, is considered viable since researchers should take responsibility of their own projects and therefore test their own code. This method is also in line with the development methods discussed by [58], which describes Facebook's lack of a QA team. The developers are themselves responsible for their code, for writing test cases and testing them. They also have to support the operational use of the software they have created.

The primary studies [S7, S9, S24, S25], emphasise on the fact that shared responsibilities between the development and operations teams is a vital part of the DevOps culture. Both the development and operations teams have to take shared responsibility of all the stages in the SDLC. Blaming the other team for errors and not providing help to resolve issues originated by the other team, is not how DevOps should be practised [S7, S24, S25]. Therefore, teams must make decisions together, be accountable for the work they have completed and collaborate to solve the problems that might arise.

5.1.3 Defining principles of DevOps

According to the primary studies [S3, S6, S9, S13, S18], DevOps is mainly defined by using four core principles: culture, automation, measurement and sharing, which is also referred to as the CAMS-model, first introduced by Damon Edwards and John Willis [S6, [59]]. The model contains the most important principles of DevOps and can be used as a framework for implementing DevOps. These four principles are often grouped together when researchers try to define DevOps.

Beyond these four principles, the primary studies [S6, S9] provide a few more principles that they believe are vital for defining DevOps. They both characterise QA as a principle of DevOps and emphasise on that development, operations and customers all have to perform their duties in a reliable and efficient manner. Quality is increased if these stakeholders work in a close relation, to understand issues and risks. Leanness is also mentioned in the primary study [S6]. As previously mentioned, Lean is the backbone of both Agile methods and DevOps. Lean processes are vital for DevOps to enable continuous practises to develop and deliver software continuously and in an incremental fashion [S6]. As a result, six core principles were defined. These principles, shown in Figure 4, are: Collaboration, Automation, Measurement, Sharing, Quality Assurance and Lean.

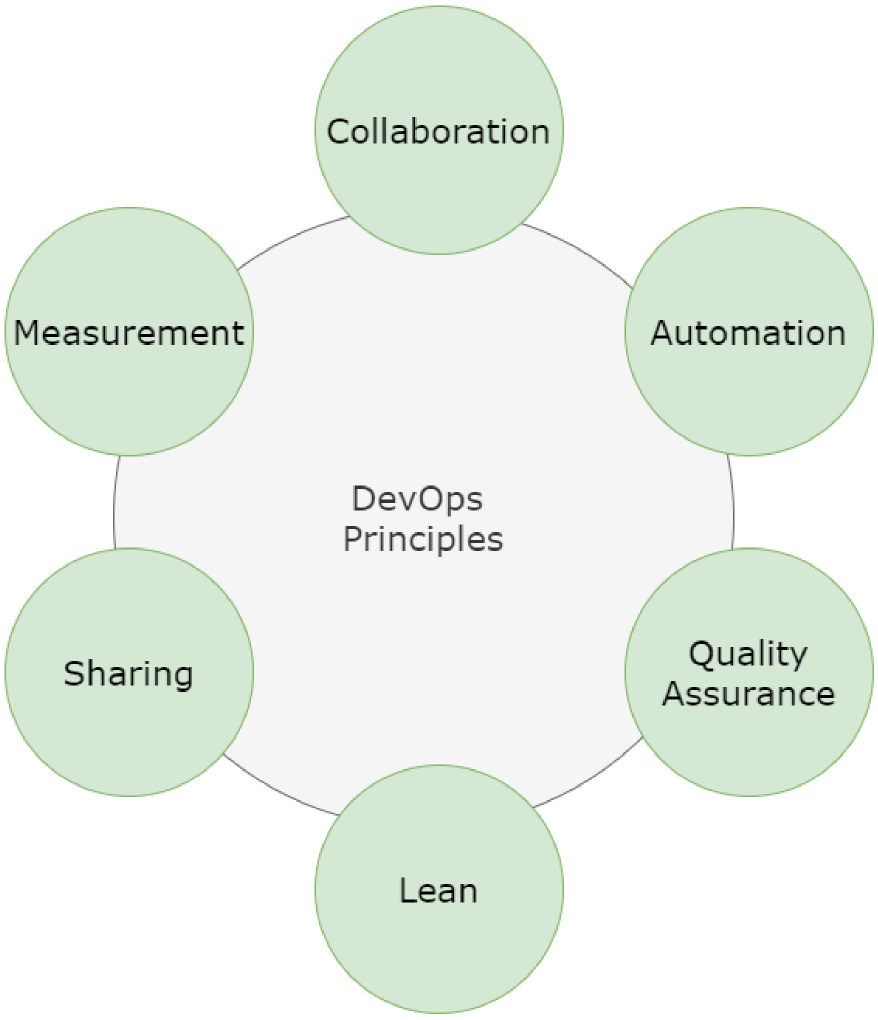

Figure 4. The defining principles of DevOps.

5.2 DevOps adoption (RQ2)

The process of adopting DevOps can be challenging in many ways (see Section 5.4), since there is no clear method or framework for adopting DevOps. Organisations might also want to adopt DevOps in slightly different ways, depending on their current DevOps maturity level. This section will provide some general suggestions on the methods and tools that can help during a DevOps adoption process.

5.2.1 CI/CD Pipeline

The primary studies [S5, S12, S13, S23] describe the main objectives of a successful DevOps adoption. The objectives are to achieve continuous delivery and to successfully implement a CI/CD Pipeline. The pipeline has automated steps in the delivery process, from testing the code to deploying it into production. Each step validates and tests the code change, and if approved, the code is sent to the next stage. The process is aborted if the code change fails any of the pipeline stages and the developers are notified to resolve the problem.

The primary study [S23] describes the CI/CD Pipeline as a way to fully automate the delivery process, to significantly reduce release cycle times and to reduce costs and risks of the software development project. Further, they define the CI/CD Pipeline as the last step of the supply chain in software development. The automated pipeline can be implemented differently, depending on organisation. The solution presented in [S5] consists of the following steps: units tests, platform tests, deliver to staging, application acceptance tests, deploy to production and post deploy tests, shown in Figure 5. The primary studies [S13, S23] present similar pipelines, including steps like building, automated testing and deployment. According to [S5, S12, S23], the pipeline was quite challenging to implement, requiring a wide range of tools (see Section 5.2.3) and testing to implement. The whole process also has to be monitored, which will be discussed in the next section.

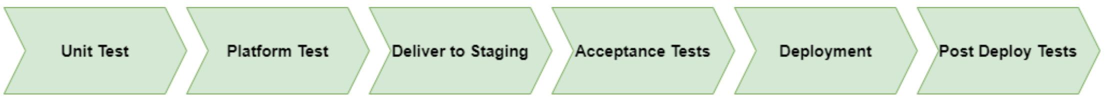

Figure 5: CI/CD Pipeline, adopted from [S5]

5.2.2 Monitoring

Monitoring is a vital part of DevOps and can be applied to most parts of the SDLC. Monitoring is discussed in the primary studies [S5, S7, S8, S11, S12, S23, S24]. In the context of DevOps, the monitoring process is often automated and performed continuously [S5, S7, S8, S11, S23, S24]. Some researchers believe that it is paramount that the monitoring process is a part of the continuous delivery process and that logs are continuously aggregated during the coding face [S8]. Some of the primary studies [S7, S24] also divide monitoring into two categories, infrastructure monitoring and application monitoring. The infrastructure is monitored to enhance the planning and development processes to bring increased value and business results. According to the primary study [S7], monitoring automation is *"the ability to monitor the applications and infrastructure without human intervention".* The monitoring process is there to control the functionality and state of the system and to alarm the developers if something in the system malfunctions. The notification can be sent out, using a chat tool, such as Slack or Hip Chat [S5, S7, S11]. The primary study [S11], presents a model for DevOps task categories and communication. In this model, monitor automation is a system actor within the SDLC and workers are monitored, information is extracted from the Source Control System and the state and functionality of all other systems within the SDLC are monitored, notifying the correct actor if an issue occurs. According to [S5], dashboards are a basic service, used for monitoring releases, users using the system and the country the users originate from. There were also dashboards to enable unique teams to monitor their part of the application. Lastly, the primary study [S12], discusses monitoring as a means to confirm whether a deprecated feature is still in use or whether it can be removed from the application.

5.2.3 Tools

The primary studies that discuss DevOps tools are [S2, S4, S7, S10, S14, S16, S19, S23, S25], and they all present tools as a core factor in adopting DevOps. Tools are used throughout the development process and within the CI/CD Pipeline. The primary studies mainly discuss tools used for communication, monitoring, testing and releasing. Tools are used to support the development of software and aid the development and operations teams to more easily collaborate, develop and deploy software. Collaboration tools are discussed in the primary articles [S7, S16]. Collaboration tools such as Slack and Hip Chat are used for communication via messaging and to report alarms through notifications from monitoring tools.

According to [S4, S16, S14, S19], tools used for version control are mainly git-based tools, such as GitHub and Bitbucket. Git enables each developer to control its code in a local repository and the developer is then able to push the changes to a centralised repository or pull changes done by others. The older versions of the code are also available, if any changes have to be reverted. Monitoring tools such as New Relic, Graylog and Kinesis are briefly mentioned by [S4]; they provide monitor automation, the ability to send out error notifications to the teams and graphs and dashboards that represent the monitored data. According to [S4, S19], testing can be done using Selenium or TestNG, with the help of the automation tool Jenkins. Selenium is able to test for clicks, links, CSS, text, tag names, etcetera. After the testing is completed, tools such as Jenkins, Chef, Puppet, Ansible and Docker are used for automation, containerisation, delivery and deployment. These are also the tools that enable continuous integration and the CI/CD Pipeline [S4, S10, S14, S16, S19].

5.3 Benefits (RQ3)

When analysing literature on DevOps, there are many benefits of adopting DevOps. During the literature review, I found six articles that display these benefits [S4-S6, S13, S21, S23]. All the articles provide similar conclusions of the known benefits of DevOps. Therefore, the merits of DevOps should be known for organisations adopting DevOps, even though the process is far from trivial.

5.3.1 Faster release cycles

A release cycle describes the process that begins with the completion of code and ends with the release of the code in production. In a DevOps setting, the release cycle is often very short, ranging from a few hours to a few weeks. All of the six articles above agree that faster release cycles are a benefit for adopting DevOps, which implies that more frequent release cycle times is one of the more prevalent benefits of DevOps. This is also supported by the fact that DevOps is believed to extend Agile principles [S1, S13], which also addresses more frequent release cycle times as one of its core principles [60].

Faster release cycle times were also discussed in the case study performed in [S4]. They conclude that *“An improved speed in the delivery of software changes was the most* *commonly perceived benefit of DevOps"*, which was the perspective presented by four out of five case companies in their study. The release times were reduced to a few days from previously being several months. The articles [S5, S6, S21] also mention that DevOps reduces the release cycles and it can be concluded that release cycles is an important benefit of DevOps adoption. Lastly, the survey results of [S23] conclude that:

> *[...] majority of the respondents highlighted the advantage of the DevOps activities are found to reduce the software cycle time as what they believe DevOps practice can achieve.*

Since this statement is also consistent with the other literature discussing benefits of DevOps, one can conclude that one of the key benefits of DevOps is faster release cycle.

5.3.2 Higher productivity

Productivity can be defined in many ways, for example as the measurement of the effectiveness of a person or system to convert inputs into outputs. The value of productivity is calculated by dividing the output, with the costs incurred or resources spent. With software development, productivity can be measured by lines of code. The number of lines of code can be compared to another factor, such as the time it take to write the line, which can be measured as lines of code per hour. Within the scope of this study in mind, the output is the feature or code that is being released and the input the labour and other costs of the process. Productivity can also be measured as number of releases in a given timeframe. There are some practices of DevOps that increase productivity, such as automation, better collaboration and communication. The primary study provided some proof of improved productivity when adopting DevOps. The primary studies [S4, S5, S21] argue that better communication, less bureaucracy and decreased organisational boundaries improved the productivity of development and operations teams. Further, the development and operations team must improve their communication to collaborate better, which would increase their productivity. By adopting DevOps, bottlenecks can be eliminated, which can greatly increase the productivity of the different teams. According to [S6], the implementation of continuous integration and continuous feedback increases the productivity of development teams.

Lastly, productivity is enhanced by sharing knowledge across the different teams, by tools to manage the shared knowledge and by tracking the knowledge needs throughout the development life cycle [S21].

5.3.3 Quality

In the scope of this research, the term quality is used to describe the quality of code and the quality of systems and applications. The literature on the benefits of DevOps is unanimous in the sense that both code quality and application quality is an important benefit of DevOps [S4-S6, S13, S23]. The primary studies [S4, S5, S23] have conducted case studies, analysing companies using DevOps. They all came to the same conclusion that the quality increases when implementing DevOps practices. They also conclude that with increased responsibility and ownership of developers' own code, the developers tend to produce code of higher quality. As a result, they believe that the work they do is more significant, since they are also a part of the deployment and postdeployment-stages. Furthermore, with smaller incremented releases the developers are more confident that their code is going to pass the tests and be deployed into the production environment. The primary studies [S6 and S13], discuss the QA of DevOps and conclude that DevOps can drive QA by enhancing communication and feedback loops. The case study conducted by [S14], reports that all the companies in the study recognised that DevOps enhances production quality and reduces risks. The automation of processes, such as testing and deployment, made a strong impact on the quality of the code. Faster release cycles enabled deployment in smaller increments, which increased the production quality, since the risks and quality were easy to control.

5.4 Challenges (RQ4)

While the benefits of adopting DevOps are clear, there are still several challenges related to adopting DevOps. There were eleven articles that discussed challenges of adopting DevOps [S2, S4-S6, S13, S15, S17, S19-S20, S22-S23]. I was able to find three main themes that especially challenged organisations, while adopting DevOps. The themes were the lack of skills and knowledge of the personnel implementing DevOps, the resistance to change by management and the organisation itself and the cost and scarcity of tools and automation.

5.4.1 Lack of skills and knowledge

From the literature it became clear that DevOps lacks a clear definition or framework on how it is supposed to be implemented. This results in companies being uneducated about DevOps, which makes it more difficult to adopt. Therefore, the lack of education, skills and knowledge around DevOps is an extensive challenge, which is also discussed in most of the articles related to DevOps challenges [S4, S5, S13, S15, S17, S23]. The articles present a clear problem with the lack of skills, knowledge, education and clear instructions on how to adopt DevOps. In [S4], the researchers found that the necessary skills and knowledge are required to develop, test, integrate and deploy software. In one of the cases in their study, the company adopting DevOps had vast difficulties with new technologies and platforms being implemented at the same time. Even in a company with a high skillset of technology and knowledge, the practitioners had problems with the DevOps approach. In another setting, the primary study [S5] acknowledges that recruiting the correct personnel with adequate skillsets is vital. Furthermore, lack of knowledge leads to poorer DevOps adoption:

> *The lack of appropriately skilled staff can lead to slowing down of the DevOps adoption journey because the capabilities needed are missing at the time of need.*

The primary studies [S13, S15, S17] conclude that one of the main challenges of DevOps is the lack of understanding and expertise on how to adopt DevOps. This is seen as one of the main challenges of adopting DevOps. Lastly, the researchers in [S23] establish that the guidelines of DevOps adoption are lacking and can lead to delays in the software release cycle.

5.4.2 Resistance to change

Another considerable challenge for adopting DevOps is the resistance to change, expressed by practitioners and upper management. The literature discuss the matter from two perspectives, the practitioners and the managers. The practitioners, such as members of development and operations teams, might resist the adoption of DevOps for fear of losing their job or by not agreeing with the collaborative atmosphere. The different teams might have separate skills and tasks, which make it more difficult to work together [S5, S13]. According to [S15], there are conflicting goals between development and operations teams. Developers tend to want new features and bug fixes to be released in production rapidly. Meanwhile, the operations team want to keep the releases to a minimum, to preserve the stability and reliability of the system. The other perspective is that of the senior management resisting DevOps adoption. The literature provides some views on the problems emerged, as a result of senior management. According to [S15]:

> *If the benefits of adopting DevOps are not clear, top management will resist by questioning the feasibility and wisdom of implementing it.*

The tools of DevOps are expensive and require proper management, which might deter managers from seeing the value of DevOps. DevOps adoption is also limited by hierarchical and inflexible management style [S6, S15, S17]. The researchers in [S17] discuss the lack of productivity in the beginning of the adoption process, which is portrayed as a problem for senior management. As a result of these productivity issues, senior management can be hesitant to adopt DevOps,

5.4.3 Scarcity and cost of tools and automation

Another common challenge that emerged from the primary studies was the lack of tools, cost of them and poor knowledge of their usage. The primary studies [S2, S5, S6, S17, S19, S20, S23] all present tools as a challenge for adopting DevOps. DevOps practitioners are having difficulties finding the correct tools and using them in a reliable manner for continuous practises [S2, S5, S20]. Furthermore, the initial setup of the tools, the experimenting and making the decision on which tool to use, is seen as a time-consuming, slow and complex endeavour [S5]. The high cost of DevOps tools is also seen as a negative and management have a difficult time justifying the investment. The tools are often unproductive in the early stages and require some setup and management, which increases the threshold for implementing them [S6, S17, S19]. The primary study [S20] point out that a lack of standardisation between tools is a hindrance for adopting continuous practises and, therefore, DevOps. Lastly, tools be a liability if they are managed poorly. The primary study [S23], found out that:

> *The evidence of this study shows that the asset can become a liability if the resources control is managed wrongly. Respondents mentioned the failure of resources control could lead to resources overhead during the integration of the source codes when all source codes have been deployed too often. As a result, this will jeopardize the Continuous Delivery Pipeline. In some cases, the automated test is failed because of the environment used in the production is different and very complex to be executed with the automated test.*

According to the statement above, the repeated deployment of features can result in the version in production being too complex, failing the tests executed within the Continuous Delivery Pipeline.

6 Discussion

The result of this study provides some key areas that present the essence of DevOps. Since there has not been any clear prior definition of DevOps, one of the main contributions of this thesis has been to provide approaches to define DevOps. The following themes were extensively discussed in the primary literature: collaboration, ownership of code and shared responsibilities between teams. While these themes are important for defining DevOps, they do not provide a conclusive definition for DevOps. This fact provides the notion that a clear definition of DevOps was not obtained in this research. The reasons for this are the scarcity of prior research into DevOps and the lack of a clear definition of DevOps in prior literature. DevOps is more of a philosophy of software development, than a concrete framework and organisations have different means of adopting DevOps.

The adoption of DevOps is highly dependent on the organisations ability to automate the various steps of the SDLC. A common practice used to achieve this, is to implement a CI/CD Pipeline, that the new piece of code passes through, while being tested and finally deployed into production. Sustaining a high level of deployability can be challenging, but as shown in [46] it is more than possible and the benefits can be tremendous. To research more into the CI/CD Pipeline, it would be beneficial to implement a pipeline and try to find a general framework that could be followed to more easily execute the implementation. Throughout the whole CI/CD Pipeline and SDLC, the system is monitored for user behaviour, feedback and errors. The result of this study provides a conclusive belief that monitoring is mainly automated and used continuously for better efficiency and reliability. This is quite logical, since monitor practices is better to do in the background, without having to use the time of developers for monitoring. The research provided the notion, that monitoring is one of the more mature DevOps practises, with several tools available for most of the monitoring needs. Tools are also used for several other DevOps practices and they are a vital part of a successful DevOps adoption. Tools are used to support practitioners with collaboration, development and deployment tasks. Tools enable the automation of redundant and easy tasks, which increases the efficiency of the software development teams. The State of DevOps Report of 2018 [61], also view tooling as an important part of DevOps, and they found that: *"highly evolved orgs are 44x more likely to contribute to other teams'* *tooling”*. This provides some proof that organisations that share their tools and use them correctly, have a higher DevOps maturity level than organisations that do not.

The result also presented benefits and challenges of adopting DevOps. The most discussed benefits were faster release cycles, higher productivity and higher quality. The fact that DevOps accelerated the development and release cycles is quite expected, since DevOps is an extension to Agile principles. Nevertheless, the fact that DevOps can reduce the release of a new feature, from a few months to a few days or even a few hours, is quite remarkable. The State of DevOps Report of 2013 [62] also back up the findings, stating that organisations deploy software 30 times more frequently and 8000 times faster than other non-DevOps organisations. This makes DevOps a very interesting way to enhance the productivity within software development. According to the result, higher productivity was also found to be a strong benefit of adopting DevOps. Productivity is a quite broad term, with many possible utilisations and definitions. Nevertheless, in the setting of DevOps, one way is to measure the amount of releases in a given timeframe. According to the result, many of the core DevOps principles, such as automation, increase the productivity of software development. The report [62] back this by stating: *“Version control and automation together enable the highest levels of efficiency and productivity”* This fact is quite sensible, since a more effective usage of recourses is a way to improve productivity. DevOps also improves the quality of code and the quality of developed applications. Developers tend to create code of higher quality if they are responsible for its whole lifecycle from creation to deployment. This is quite logical, since by forcing developers to be accountable for their own code, they have a greater chance to care about the end result, which should promote them to produce code of higher quality.

Lastly, there are also challenges of adopting DevOps. According to the results of this study, the lack of skills, education and knowledge around DevOps is a hindrance for its adoption. This result is also backed up by the article [63], which conclude that: *“Our findings show the importance of skills and skill categories to build effective and successful DevOps team”*. It is quite sensible that the skill level of DevOps is high, since developers now have to execute tasks that previously has been completed by the operations team. This problem is not improved by the second DevOps challenge, which is the fact that practitioners and management can have a high resistance to change. The results coined from the primary studies present that resistance to change is a considerable challenge for adopting DevOps. This fact is also backed up by [64] and the fact that change is hard to accept and the fear of losing ones job to automation, might even enhance it. The last challenge identified through the research, was the cost and scarcity of tools. The findings provide the impression that the correct tools can be difficult to find, and the initial setup of tools can be costly and time-consuming. Subsequently, the result provides a clear analysis on the definitions, adoption, benefits and challenges of DevOps. Nevertheless, further research into the topics concerning DevOps is required. This study could have benefited from another angle of approach, either by interviewing DevOps practitioners or by implementing a DevOps solution, such as a CI/CD Pipeline.

6.1 Threats to Validity

The research in this thesis has potentially been exposed to two types of biases, selection bias and publication bias. A systematic approach was used to collect data from previous literature on different aspects of DevOps. The protocol [40] was accurately followed, which means that the same results should be yielded if the study is conducted again by another researcher. Nevertheless, the protocol was performed entirely by one researcher, which can be seen as a threat to the validity and cause some of the steps of the protocol to be exposed to selection bias. The research in this thesis is also exposed to publication bias, since researchers and organisations tend to present only positive results. Organisations that have failed to adopt DevOps might therefore not write about it, leaving out important evidence for the drawbacks of DevOps. These two types of biases have been considered during the research process and the impacts of them has been mitigated. The mitigation has been accomplished by using a very long and thorough selection process for reducing the selection bias. The publication bias has been mitigated by assessing the quality of the articles, by ensuring that also negative results are present in the primary studies.

7 Conclusions

The purpose of this thesis was to provide an understanding of the definition of DevOps, how DevOps can be adopted and the benefits and challenges of adopting DevOps. The research was conducted by carrying out a literature review to receive an understanding on the previous research on DevOps and to answer the research questions. DevOps is an interesting concept with numerous potentials within software development. DevOps is something that organisations strives to adopt, but without a clear framework or definition, it is hard to convince organisations to implement DevOps. The result provides a view on the definitions, adoption, benefits and challenges of DevOps. Nevertheless, it is still difficult to provide a clear definition of DevOps and further research is needed for a conclusive definition to be found. The result is backed by 25 primary studies that were selected for analysis. The data extracted from these studies resulted in twelve core themes, that were selected on the basis of their impact for answering the research questions and the level of backing they received from the primary studies. This contributed to the characterisation of many key topics on the definition of DevOps, what organisations should have in mind while adopting DevOps, the benefits and the challenges of adopting DevOps. The result identified the key defining topics of DevOps to be culture of collaboration, automation, metrics, sharing, QA and Lean. These are the principles, which serve as the backbone of DevOps. The result also provided some insight into the structure of the CI/CD Pipeline, how organisations can monitor the development process and the usage of their applications and the tools they should consider when adopting DevOps. The benefits realised from the results were the faster release cycles of software in organisations adopting DevOps. The release cycle times were cut to mare days or hours, from several weeks or months. These improvements also resulted in higher productivity within the SDLC. It is possible to ensure higher quality of the produced software, through the automation of vital steps within the SDLC and better interaction with different stakeholders. Challenges of DevOps were also defined, and the most prevalent challenges were the lack of skills and knowledge of DevOps, the resistance to change and the scarcity and costs of tools. The overall result aid practitioners, management and anyone interested in DevOps on its several key areas and helps stakeholders of software development organisations to produce software more rapidly, more effectively and with higher quality.

Svensk sammanfattning

DevOps - en systematisk litteraturstudie över definitioner, implementationer, fördelar och utmaningar

Dagens programvaruproduktion präglas av hög konkurrens och höga krav. Företag är tvungna att producera sin programvara mer effektivt för att vara konkurrenskraftiga på marknaden. Programvaruproduktion har tidigare ansetts vara en trög procedur med begränsad flexibilitet. Vattenfallsmetoden har varit den dominerande metoden för programvaruproduktion, dock är metoden väldigt icke-flexibel och lämpar sig inte för alla projekt eller organisationer [1]. För att lösa problemen med vattenfallsmetoden infördes den agila systemutvecklingen i början av 2000-talet. Agila metoder förespråkar ett mer iterativt och flexibelt sätt att producera programvara, en bättre relation till kunden samt snabbare cykler för utgivning av programvaran [60]. Agila metoder är dock mest lämpade för mindre företag som endast har ett team som jobbar med hela programutvecklingsprocessen. I större programmeringsföretag är de som producerar programvara oftast uppdelade i två team, de som kodar programmet och de som publicerar koden i kundens miljö. Dessa två team har hittills inte kommunicerat eller samarbetat på ett effektivt sätt, vilket har resulterat i flaskhalsar inom programutvecklingsprocessen. Tanken bakom DevOps är att utöka de agila metoderna och få teamen att samarbeta och kommunicera bättre. Detta är möjligt att genomföra genom att förbättra företagskulturen, automatisera och övervaka programutvecklingsprocessen samt genom att dela kunskap och ansvar mellan teamen [65].

Syftet med avhandlingen är att få en bättre bild av hur DevOps definieras och införas. Syftet med avhandlingen är också att få fram positiva och negativa sidor med ibruktagandet av DevOps. Syftet stärks eftersom den befintliga forskningen i ämnet är begränsad och det inte finns en klar definition över vad DevOps är eller ett lämpligt ramverk för implementeringen av DevOps.

Metoden som använts för att uppnå avhandlingens resultat är en systematisk litteraturstudie. De primära artiklarna som använts som grund för resultatet, identifierades inom de elektroniska databaserna IEEE Xplore, ACM DL och Science Direct. Sökningen bidrog till 554 artiklar varav 25 användes. Datainsamlingen skedde med hjälp av ett datainsamlingsformulär som tillämpades på de artiklar som valdes inom ramen för litteraturstudien. Studiens reliabilitet kan ses som delvis partisk då alla steg inom undersökningen endast utfördes av en person. En väldigt specifik process följdes ändå för att identifiera de artiklar som tillhör studien, vilket torde bidra till en högre trovärdighet.

Undersökningen genomfördes genom att välja ut de artiklar som bäst lämpades för att besvara forskningsfrågorna. Valet av artiklar skedde i tre faser. Första fasen bestod av en titel- och abstraktgenomgång varav 77 artiklar accepterades till nästa fas. I den andra fasen valdes artiklar enligt textens hela innehåll och 34 artiklar accepterade i den andra fasen. I sista fasen bedömdes de resterande artiklarnas kvalitet ur ett innehållsperspektiv och enligt kvaliteten på själva forskningen i artikeln. De 25 artiklar som accepterades i denna fas var även de slutgiltiga primära artiklarna som undersökningen grundades på. Efter valprocessen och datainsamlingen identifierades de centralaste teman för varje forskningsfråga. Dessa teman var: samarbete, äganderätt och ansvar, principer för DevOps, övervakning, verktyg, snabbare lanseringscykler, högre produktivitet, kvalitet, brist på färdigheter och kunskap, motstånd till ändring och brist på verktyg. Då dessa kategorier var bestämda beskrevs de i avhandlingens resultat.

Avhandlingens resultat besvarar de fyra forskningsfrågorna. Den mest centrala delen av undersökningen var att hitta en definition för DevOps. De tre teman som utgjorde en betydelsefull del i datainsamlingen var samarbete, gemensamt ansvar och principer som definierar DevOps. Undersökningen visade att samarbete mellan de olika teamen inom produktionsutvecklingen är en av de centralaste egenskaperna hos DevOps. Genom samarbete strävar man efter att minska på kommunikationsproblem mellan de olika teamen inom produktionsutvecklingsprocessen. Via samarbete kan man också minska på den tid det tar att producera programvara. De olika teamen ska även ta gemensamt ansvar för de olika momenten inom programvaruutvecklingsprocessen. Kodare skall genom hela processen ta fullt ansvar över sin kod, och produktionsteamet skall hjälpa kodarna med automation och lanserandet av koden i kundens miljö. Eventuella problem ska lösas genom diskussioner oberoende ursprunget till problemet. För att definiera DevOps används oftast principer som samarbete, automation, delning och mätning.

Dessa kan även användas som ett ramverk för att införa DevOps inom organisationer. Målet med DevOps anses vara att uppnå kontinuerlig leverans av programvara som även bygger på automation. För att uppnå detta används oftast en pipeline som den nya koden passerar igenom, medan koden utsätts för tester och kvalitetskontroller. Utöver detta så används övervakningsprogram för att övervaka de olika processerna för att kunna ge feedback över hur bra det nya programmet fungerar för att meddela då det sker något fel. Dessutom är DevOps beroende av en del verktyg för att underlätta de olika stegen inom programvaruutvecklingsprocessen. Utöver övervakning, används verktyg för automatisering, kommunikation, sparandet av kod, testning samt lansering av programvara. Det finns en del fördelar med att använda sig av DevOps-metoder. Enligt undersökningen som utfördes förbättras den tid det tar att lansera programvara märkbart. Det framkom att då koden lanseras i kundens miljö mer frekvent ~~så,~~ förbättras kvaliteten och riskerna minskar eftersom man lanserar koden i mindre fragment. Produktiviteten ökar även när de olika teamen samarbetar bättre med varandra, när byråkratin minskar och när organisatoriska gränser minskar. Genom att införa DevOps kan man bli av med flaskhalsar inom programvaruutvecklingsprocessen, vilket kan öka produktiviteten märkbart.

Slutligen finns det även utmaningar för att införa DevOps. Då företag har en snäv uppfattning om DevOps och deras anställda inte har den nödvändiga utbildningen, finns det risk för att implementeringen av DevOps inte lyckas eller blir bristfällig. Det kan även finnas en del motstånd från den högre ledningen, eftersom vissa aspekter av DevOps kan kräva stora investeringar och omstruktureringar inom organisationen. De anställda kan även ha en rädsla för att deras jobb blir föråldrat, då automation tas i bruk. Verktygen för att implementera DevOps kan även vara dyra och det finns inte alltid lämpliga verktyg för alla delar inom programvaruutvecklingsprocessen.

References

[1] R. Stephens, Beginning Software Engineering, John Wiley & Sons, Incorporated, 2017.

[2] A. Ravichandran, K. Taylor and P. Waterhouse, DevOps for Digital Leaders, Berkley, CA: Apress, 2016.

[3] R. Jabbari, N. bin Al, K. Petersen and B. Tanveer, "What is DevOps?: A Systematic Mapping Study on Definitions and Practices," *XP '16 Workshops: Proceedings of the Scientific Workshop Proceedings of XP2016,* 2016.

[4] N. de França, H. Jeronimo and G. H. Travassos, "Characterizing DevOps by Hearing Multiple Voices," *SBES '16: Proceedings of the 30th Brazilian Symposium on Software Engineering,* 2016.

[5] F. Erich, C. Amrit and M. Daneva, "Report: DevOps Literature Review," 2014.

[6] Google Inc, "Google Trends," 2020. [Online]. Available: https://trends.google.com/trends/explore?date=2009-02-24%202020-03-24&q=%2Fm%2F0c3tq11. [Accessed 24 03 2020].

[7] Gartner Inc, "Gartner Conferances," 2020. [Online]. Available: https://www.gartner.com/en/conferences/apac/infrastructure-operations-cloud-india/featured-topics/devops. [Accessed 24 03 2020].

[8] Puppet Labs (2019), "State of DevOps Report 2019," 2019.

[9] A. Rathod, "Different Organizations, Different DevOps Outcomes," MediaOps Inc., 2019. [Online]. Available: https://devops.com/different-organizations-different-devops-outcomes/. [Accessed 24 03 2020].

[10] M. Rütz, "DEVOPS: A SYSTEMATIC LITERATURE REVIEW," 2019.

[11] L. Lwakatare, "DEVOPS ADOPTION AND IMPLEMENTATION IN SOFTWARE DEVELOPMENT PRACTICE," 2017.

[12] N. Ruparelia, "Software Development Lifecycle Models," *ACM SIGSOFT Software Engineering Notes,* 2010.

[13] W. Royce, "Managing the development of large software systems," 1970.

[14] P. Measey, Agile Foundations: Principles, practices and frameworks, BCS Learning & Development Limited, 2015.

[15] Beck et. al, "Manifesto for Agile Software Development," 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed 10 2 2020].

[16] K. Pries and J. Quigley, Scrum Project Management, CRC Press LLC, 2010.

[17] D. Andersson, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 2010.

[18] K. Leopold and S. Kaltenecker, Kanban Change Leadership: Creating a Culture of Continuous Improvement, John Wiley & Sons, Incorporated, 2015.

[19] H. Hyytiälä, "The Kanban method," Reaktor Group Oy, 2011. [Online]. Available: https://www.reaktor.com/blog/the-kanban-method/. [Accessed 21 02 2020].

[20] Lean Enterprise Institute, Inc, "TOYOTA PRODUCTION SYSTEM," 2020. [Online]. Available: https://www.lean.org/lexicon/toyota-production-system. [Accessed 10 2 2020].

[21] S. Sharma, The DevOps adoption playbook : a guide to adopting devOps in a multi-speed IT enterprise, Indianapolis: John Wiley and Sons, 2017.

[22] S. Mezak, "The Origins of DevOps: What's in a Name?," MediaOps Inc, 28 1 2018. [Online]. Available: https://devops.com/the-origins-of-devops-whats-in-a-name/. [Accessed 10 2 2020].

[23] J. Hamunen, "Challenges in Adopting a Devops Approach to Software Development and Operations," 2016.

[24] IBM , "IBM UrbanCode," 2020. [Online]. Available: https://www.ibm.com/cloud/urbancode. [Accessed 10 2 2020].

[25] S. Hussaini, "Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through Systems approach.," *2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC),* 2014.

[26] M. Senapathi, J. Buchan and H. Osman, "DevOps Capabilities, Practices, and Challenges: Insights from a Case," *EASE'18 Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering,* pp. 57-67, 2018.

[27] B. Fitzgerald and K.-J. Stol, "Continuous Software Engineering and Beyond:Trends and Challenges," *RCoSE 2014: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering,* 2014.

[28] M. Virmani, "Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery," *Fifth international conference on Innovative Computing Technology (INTECH 2015),* 2015.

[29] L. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, P. Kuvaja, V. Heikkilä, J. Itkonen, T. Mikkonen, M. Oivo and C. Lassenius, "DevOps in practice: A multiple case study of five companies," *Information and Software Technology,* vol. 114, pp. 217-230, 2019.

[30] M. Shahin, M. A. Babar, M. Zahedi and L. Zhu, "Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges," *Proceedings of*

the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2017.

[31] M. Shahin, M. A. Babar and L. Zhu, “The Intersection of Continuous Deployment and Architecting Process: Practitioners' Perspectives,” *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,* pp. 1-10, 2016.

[32] M. Fowler, “martinFowler.com,” 1 05 2006. [Online]. Available: https://martinfowler.com/articles/continuousIntegration.html. [Accessed 11 2 2020].

[33] D. Saff and M. Ernst, “An Experimental Evaluation of Continuous Testing During Development,” *ACM SIGSOFT Software Engineering Notes,* 2004.

[34] R. Sturm, C. Pollard and J. Craig, Application Performance Management (APM) in the Digital Enterprise, Elsevier Inc, 2017.

[35] J. Smeds, K. Nybom and I. Porres, “DevOps: A Definition and Perceived Adoption Impediments,” *Continuous Strategy Process in the context of Agile and Lean Software Development,* pp. 166-177, 2015.

[36] W. Luz, G. Pinto and R. Bonifácio, “Building a collaborative culture: a grounded theory of well succeeded devops adoption in practice,” *ESEM '18: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,* 2018.

[37] T. Theunissen and U. Heesch, “Specification in Continuous Software Development,” *Proceedings of the 22nd European Conference on Pattern Languages of Programs,* 2017.

[38] M. Walls, Building a DevOps Culture, O'Reilly Media, Inc., 2013.

[39] P. Mahanta, V. Adige, A. Pole and R. M, “DevOps Culture and its impact on Cloud Delivery and Software Development,” *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring),* 2016.

[40] K. Nybom, A. Ashraf and I. Porres, “A Systematic Mapping Study on API Documentation Generation Approaches,” *2018 44th Euromicro Conference on Software Engineering and Advanced Applications,* 2018.

[41] L. Lwakatare, T. Karvonen, T. Sauvola, P. Kuvaja, H. Olsson, J. Bosch and M. Oivo, “Towards DevOps in the Embedded Systems Domain: Why is It so Hard?,” *2016 49th Hawaii International Conference on System Sciences,* 2016.

[42] V. Gupta, P. Kapur and D. Kumar, “Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling,” *Information and Software Technology,* 2017.

[43] T. Theunissen and U. van Heesch, “Specification in Continuous Software Development,” *EuroPLoP '17: Proceedings of the 22nd European Conference on Pattern Languages of Programs,* 2017.

[44] M. de Bayser, L. Azevedo and R. Cerqueira, “ResearchOps: The case for DevOps in scientific applications,” *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM),* 2015.

[45] C. Cois, J. Yankel and A. Connell, “Modern DevOps: Optimizing software development through effective system interactions,” *2014 IEEE International Professional Communication Conference (IPCC),* 2014.

[46] L. Chen, “Microservices: Architecting for Continuous Delivery and DevOps,” *IEEE International Conference on Software Architecture,* 2018.

[47] M. Ibrahim, S. Syed-Mohamad and M. Husin, “Managing Quality Assurance Challenges of DevOps through Analytics,” *ICSCA '19: Proceedings of the 2019 8th International Conference on Software and Computer Applications,* 2019.

[48] S. Jones, J. Noppen and F. Lettice, “Management Challenges for DevOps Adoption within UK SMEs,” *QUDOS 2016: Proceedings of the 2nd International Workshop on Quality-Aware DevOps,* 2016.

[49] M. Kamuto and J. Langerman, “Factors Inhibiting the Adoption of DevOps in Large Organisations: South African Context,” *2017 2nd IEEE International Conference On Recent Trends In Electronics Information & Communication Technology, May 19-20, 2017, India,* 2017.

[50] G. Doukoure and E. Mnkandla, “Facilitating the Management of Agile and Devops Activities: Implementation of a Data Consolidator,” *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD),* 2018.

[51] P. Perera, M. Bandara and I. Perera, “Evaluating the Impact of DevOps Practice in Sri Lankan Software Development Organizations,” *2016 International Conference on Advances in ICT for Emerging Regions (ICTer): 281 - 287,* 2016.

[52] D. Ståhl, T. Mårtensson and J. Bosch, “Continuous Practices and DevOps: beyond the buzz, what does it all mean?,” *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA),* 2017.

[53] A. Agarwal, S. Gupta and T. Choudhury, “Continuous and Integrated Software Development using DevOps,” *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE-2018),* 2018.

[54] S. Hussaini, “A Systemic Approach to Re-inforce Development and Operations Functions in Delivering an Organizational Program,” *Procedia Computer Science,* 2015.

[55] K. Kuusinen, V. Balakumar, S. Jepsen, S. Larsen, T. Lemqvist, A. Muric, A. Nielsen and O. Vestergaard, “A Large Agile Organization on Its Journey Towards DevOps,” *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA),* 2018.

[56] M. Zulfahmi, S. Sahibuddin and M. Mahrin, “Adoption Issues in DevOps from the Perspective of Continuous Delivery Pipeline,” *the 2019 8th International Conference,* 2019.

[57] K. Nybom, J. Smeds and I. Porres, “On the Impact of Mixing Responsibilities Between Devs and Ops,” *International Conference on Agile Software Development,* 2016.

[58] D. Feitelson, E. Frachtenberg and K. Beck, “Development and Deployment at Facebook,” *IEEE Internet Computing,* 2013.

[59] J. Willis, “What Devops Means to Me,” 2010. [Online]. Available: https://blog.chef.io/what-devops-means-to-me/. [Accessed 30 03 2020].

[60] Beck, K., et al., “The Agile Manifesto,” Agile Alliance, [Online]. Available: http://agilemanifesto.org/. [Accessed 31 01 2020].

[61] Puppet Labs (2018), “State of DevOps Report 2018,” 2018.

[62] Puppet Labs (2013), “State of DevOps Report 2013,” 2013.

[63] A. Wiedemann and M. Wiesche, “ARE YOU READY FOR DEVOPS? REQUIRED SKILL SET FOR DEVOPS TEAMS,” *ECIS 2018 Proceedings,* 2018.

[64] I. Bucena and M. Kirikova, “Simplifying the DevOps Adoption Process,” *BIR Workshops,* 2017.

[65] L. E. Lwakatare and e. al, “DevOps in practice: A multiple case study of five companies,” *Information and Software Technology,* vol. 114, pp. 217-230, 2019.