Lucy Ellen Lwakatare

DEVOPS ADOPTION AND IMPLEMENTATION IN SOFTWARE DEVELOPMENT PRACTICE

CONCEPT, PRACTICES, BENEFITS AND CHALLENGES
LUCY ELLEN LWAKATARE

DEVOPS ADOPTION AND IMPLEMENTATION IN SOFTWARE DEVELOPMENT PRACTICE
Concept, practices, benefits and challenges

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Technology and Natural Sciences of the University of Oulu for public defence in Känuunsali (L2), Linnanmaa, on 27 November 2017, at 12 noon

UNIVERSITY OF OULU, OULU 2017
Abstract

In the last decade, the software industry has been marked by a growing trend of software companies’ ability to deploy new software features fast and frequently in short release cycle times. The companies’ software release cycles have been shortened to hours and minutes rather than months. To enable the transformation towards short release cycle times, companies have adopted several different strategies, including the DevOps approach. DevOps in the software industry emerged to represent a professional movement emphasising the collaboration between software development and operations. In practice, DevOps affects the company culture, processes, products, associated technologies and organisational structures used in software development and operations processes. The multifaceted nature of DevOps makes the concept ambiguous and difficult for software companies to adopt as there are many different paths to its adoption.

The purpose of the thesis is to provide detailed description of the adoption and implementation of DevOps in software development comprehending the DevOps concept definition, and its practices, benefits and challenges. The research was performed by systematically reviewing the literature, multi-vocal documents and making qualitative inquiries among software practitioners; and based on that the consolidated body of knowledge of DevOps was constructed.

The key finding of the research is that the DevOps approach includes an automated software deployment mechanism focusing on the rapid and repeatable release of software changes and automated management of operational infrastructure. The adoption and implementation of DevOps practices are prominent in software companies that use cloud computing technology, while its adoption is challenging in the embedded system domain. DevOps is not a silver bullet; challenges pertaining to the management of infrastructures due to legacy technologies still persist. The key lesson learned in the adoption and implementation of DevOps is that the software operational infrastructure is no longer considered separate from the development of software features; and this is achieved by having software development and operations teams jointly working together.

Keywords: Agile, continuous deployment, DevOps, software development, software operations

Tiivistelmä

Tämän väitöksen tarkoituksena on antaa selkeä kuvaus DevOpsista ja sen toteutuksesta ohjelmistokehityksessä niin, että sen käsitytöönotot, ja käyttämät edut ja haasteet tulevat ymmärtääksesi. Tutkimuksessa suoritettiin systemaattinen kirjallisuuskatsaus tieteellisiin julkaisuihin, prosesseihin, tuotteisiin, sekä teknologioihin ja organisaatiokeränteisiin, joita käytetään ohjelmistokehityksessä ja käyttöönotoprosessissa. DevOpsin käsittäen moniulotteisuudesta johtuen käsittää sisältöä jää usein epäselväksi ja samalla sen mukainen toiminta vaikeasti käyttöönotettavaksi ohjelmistoryrityksissä, koska toteutus on mahdollista tehdä monella erin tavalla.


Asiasanat: DevOps, jatkuva käyttöönotto, ketterä, ohjelmistokehitys, ohjelmiston käyttö
To my family
Acknowledgements

‘Life as a researcher’, was part of a lecture title presented back in 2012/2013 when I was a MSc. student at the University of Oulu. The contents of that presentation have ringered throughout my Ph.D. journey. Well, with bits of my own experiences now added to it...

This dissertation contains research work performed in January 2014 – November 2017 at M3S research group of the University of Oulu. These years mark important time periods of my personal, career and scholarly development and growth. Overall, the research work was made possible and comes with great appreciation to many people and research funding sources for the support. Funding support of the work was mainly from the DIMECC N4S program and the 2015 Nokia Foundation Grant of Finland.

First and foremost I would like to thank my principal supervisor, Adjunct Professor Pasi Kuvaja for his great and very encouraging support throughout all the phases of my Ph.D. journey. At the start, Dr. Pasi Kuvaja helpfully suggested that I focus my research on the topic of DevOps and over the years we have had several discussions on the research topic. In addition, Dr. Pasi Kuvaja has been very instrumental in providing connections to industry for empirical data collection. I have tremendously valued your supervision, encouragement and more importantly the opportunity he has given to me – no words can fully describe this! I would also like to thank my second supervisor Professor Markku Oivo for his assistance, feedback and direction in my research.

I will take this opportunity to also acknowledge and thank persons that have conducted research and co-authored some of the publications with me. From Software Centre in Sweden, my appreciation goes to Professor Jan Bosch and Assistant Professor Helena Holmström Olsson for their guidance and constructive feedback to my work. From different institutions in Finland, I acknowledge the contributions of Dr. Juha Itkonen, Dr. Ville Heikkilä, Professor Casper Lassenius, Dr. Terhi Kilamo and Professor Tommi Mikkonen. Lastly, my colleagues at M3S research group, Dr. Teemu Karvonen, Tanja Sauvola, Ali Reza Haghighatkhah and Dr. Pilar Rodríguez, who over the years have worked with me in the various research activities of the thesis.

My deep appreciation for the pre-examination of this thesis goes to Professor Kari Systä and Assistant Professor Bram Adams. The comments that they both provided have shaped and brought more clarity to this work. This is also the case for the members of my Follow-up group Adjunct Professor Raija Halonen, Professor Burak Turhan and
Professor Jaakko Sauvola. My sincere thanks goes to Professor Ivica Crnkovic from Chalmers University of Technology for agreeing to act as an opponent of the thesis.

My great appreciation is also given to all my colleagues at M3S research group, for their assistance and great discussions that I had with them during my Ph.D. journey. This goes without forgetting to mention some of you: Sandun, Ovais, Rahul, Itir, Nirmaya, Pertti, Iflah, Ahmad, Mari, Olli, Woub, Vitor, Marja-Liisa, Päivi. Also friends I have come to know here in Finland: Juliet, Marie, Emmanuel, Michael, Ibris, Joanne, Seun, Funke, Biodun, Lucia, Linda, Ilaria, Negar and many more not mentioned here.

Lastly, my warmest appreciation goes to the members of my family. First, to my dad and mom for their tireless support and encouragement. Second, to my sister and brothers Karen, Emil, Godfrey and Kevin; and their spouses Chris, Kelly, Jennifer and Juliana respectively. Third, to my very dear nephews and niece: Leonel, Mateo and Skylar. Aunty Mary Eyakuze, Aunty Josephine, Dr. Flora, Betty, Upendo, Kulthum and the many more I have not mentioned here. To God be the glory.
List of original publications

This thesis is based on the following publications, which are referred to throughout the text by their Roman numerals.


Author’s Contributions

Paper I Continuous Deployment of Software-Intensive Products and Services: A Systematic Mapping Study. Lwakatare was involved in all steps of conducting systematic mapping study research and reporting some of the findings in the paper. The specific steps performed by Lwakatare included contributing to defining the research search string and performing the primary study search using the string in the ACM database. In addition, Lwakatare participated in the inclusion and exclusion of primary studies as well as their data extraction, including thematic coding of some of the papers. In addition to reviewing the paper, Lwakatare was responsible for writing the following sections of the paper: continuous testing and quality assurance and organisational factors as well as the conclusion section.

Paper II Paper II Relationship of DevOps to Agile, Lean and Continuous Deployment: A Multivocal Literature Review Study. Lwakatare is the primary author of the paper.
This means that the research design, data collection, analysis and reporting of the results were performed by Lwakatare. Lwakatare was responsible for searching and analysing multivocal literature retrieved from the Google search engine. In addition, Lwakatare analysed the data that was collected from software practitioners in a workshop. The workshop was facilitated collaboratively with other researchers from three universities including Aalto University and Tampere University of Technology. Lwakatare participated in preparing the workshop, and took notes during the workshop as well as conducted a session of sharing initial findings from multivocal literatures analysis. All sections of the paper were written by Lwakatare and reviewed by co-authors.

**Paper III** *An Exploratory Study of DevOps Extending the Dimensions of DevOps with Practices*. Lwakatare is the primary author of the paper. The research design, data collection process, analysis and reporting were performed by Lwakatare. Similar to Paper II, the multivocal literature, which served as one source of data, was collected and analysed by Lwakatare. Interviews with software practitioners served as the second source of data. The interview guide used during the interviews was developed by Lwakatare. Lwakatare was responsible for writing all sections of the paper that were reviewed by co-authors.

**Paper IV** *Towards DevOps in the Embedded Systems Domain: Why Is It So Hard?* Lwakatare is the primary author of the paper. However, the research design, data collection and analysis were done by Lwakatare together with other researchers participating in the N4S project from the M3S research group of Oulu University. Additionally, the work was done in collaboration with researchers from the Software Centre in Sweden as part of a replication study, wherein the collaborators primarily provided the original interview guide and helped with the research design. The original interview guide and overall study design were modified collaboratively by researchers with Lwakatare participating in the data collection process and having the responsibility of analysing the parts related to DevOps.

**Paper V** *DevOps in Practice: A Multiple Case Study of Five Companies*. Lwakatare is the primary author of the paper. The research design, data collection and analysis were done in collaboration with nine other researchers from three research universities that are involved in the N4S project. Lwakatare was responsible for leading the aforementioned activities and writing and reviewing all sections of the paper.
Contents

Abstract
Tiivistelmä
Acknowledgements
List of original publications
Contents

1 Introduction
   1.1 Research motivation
   1.2 Research challenges and questions
   1.3 Research scope
   1.4 Outline of the thesis

2 Background and related work
   2.1 The DevOps concept
      2.1.1 Definitions of DevOps
      2.1.2 Key elements of DevOps
   2.2 The implementation of DevOps
      2.2.1 Industrial adoption of DevOps
      2.2.2 Implementation of DevOps: organisational perspective
      2.2.3 Implementation of DevOps: socio-technical perspective
   2.3 Benefits and challenges of DevOps
      2.3.1 Benefits of DevOps
      2.3.2 Challenges of DevOps

3 Methodology
   3.1 Research design
      3.1.1 Systematic mapping study
      3.1.2 Multivocal literature review
      3.1.3 Multiple case studies
   3.2 Research process
      3.2.1 Systematic mapping study
      3.2.2 Multivocal literature review
      3.2.3 Multiple case studies
   3.3 Synthesis of the findings
4 Findings of the original publications

4.1 Paper I: Continuous deployment of software-intensive products and services: a systematic mapping study ......................... 69

4.2 Paper II: The relationship of DevOps to agile, lean and continuous deployment: a multivocal literature review study ......................... 70

4.3 Paper III: An exploratory study of DevOps: extending the dimensions of DevOps with practices ........................................ 73

4.4 Paper IV: Towards DevOps in the embedded systems domain: why is it so hard? .............................................................. 75

4.5 Paper V: DevOps in practice: a multiple case study of five companies .... 78

4.6 Summary of the findings ...................................................... 81

5 Discussion

5.1 Thesis findings in relation to related work .............................. 85

5.2 Implications for practice ...................................................... 90

5.3 Implications for research .................................................... 91

6 Conclusion

6.1 Summary of the main contributions ..................................... 93

6.2 Validity of the findings and limitations of the research .............. 94

6.2.1 Construct validity ...................................................... 94

6.2.2 External validity ...................................................... 95

6.2.3 Reliability .......................................................... 96

6.3 Future research .......................................................... 97

References 99

Appendices 105

Original publications 109
1 Introduction

Software has become pervasive in day-to-day human activities and the economy. This in turn has increased the importance of having software-intensive products and services that are useful, secure and reliable at all times during operational use. Recently, the ability of an organisation to continuously and rapidly provide new software features that are suitable to use and innovate has become an important factor of competition in the software industry market. Given enough time, any software-intensive organisation can develop great software products and services. Due to this, speed — in the form of a faster time to market — is one of the most important success factor in software industry. Speed in the development and delivery of new software features\(^1\) provides the opportunity to respond quickly to customer needs, business opportunities and get quick feedback about the new software features (Anderson et al. 2017, Geurts 2016). Feedback loops facilitate information that is useful to make informed decisions regarding software development efforts done by different stakeholders of the software project.

For modern software companies, such as Facebook, that are developing web- and cloud-based software, speed translates to their engineering capability that is effective enough to facilitate fast and repeatable software development and delivery processes (Feitelson et al. 2013). This is evident by the emergence and the growing interest of a continuous deployment paradigm in the software industry. Continuous deployment entails the capability of an organisation to deliver new software features at multiple times and in the shortest time possible. DevOps is an approach that has been reported to enable the continuous deployment paradigm (Humble & Molesky 2011) as it embodies a set of useful principles crucial to the development and deployment of software, especially for cloud-native applications (Kratzke & Quint 2017). However, much of what the DevOps approach entails is unclear, for instance it has been described as a new role within a software organisation (Kerzazi & Adams 2016), a movement for change in software industry (de França et al. 2016) and a set of software development practices (Zhu et al. 2016). There is both a research challenge and an industrial need for developing a better understanding of the DevOps concept and approach (Dingsøyr & Lassenius 2016).

---

\(^1\)This thesis adopts the definition of a feature from (Bosch 2000, p.194) and is defined as ‘a logical unit of behaviour that is specified by a set of functional and quality (non-functional) requirements’.
1.1 Research motivation

This research is motivated by the growing trend of the software industry’s transitioning towards continuous delivery and deployment (Leppanen et al. 2015, Olsson et al. 2012, Rahman et al. 2015). Continuous deployment is an industrial paradigm characterised by the frequent, rapid and automated release of software changes, including new features, to customers as soon as the changes are committed by software developer so as to gain quick feedback (Humble & Farley 2010, Olsson et al. 2012). In contrast, continuous delivery ensures that software changes are developed rapidly but not released immediately to the customers after software developer’s code commit. Evidently in Finland, the country where this work was performed, many software-intensive organisations have shown interest in continuous deployment, as seen from the Need for Speed research program.

Empirical studies and experience reports on continuous deployment provide evidence of its increasing adoption (Claps et al. 2015, Leppanen et al. 2015, Olsson et al. 2012, Parnin et al. 2017). Successful adoption of continuous deployment practice in software-intensive organisations is reported to provide many benefits, including improvements in the delivery speed of software changes, improved software quality, improved developer productivity and improved customer satisfaction (Chen 2017, Leppanen et al. 2015, Parnin et al. 2017). Challenges related to the adoption are reported, including insufficient test automation, bureaucratic software deployment processes as well as unsupportive software architecture and structure of an organisation particularly, its separation into different divisions within the software development (R&D) unit (Laukkonen et al. 2017, Parnin et al. 2017, Shahin et al. 2017a). To ensure successful adoption of continuous deployment, several practices and strategies have been recommended, including the adoption of DevOps (Humble & Molesky 2011).

As background, DevOps emerged first as a movement in the software industry. The DevOps movement emphasises collaboration between software Development and Operations with the aim of achieving a fast release of high-quality software features. The term ‘DevOps’ was first coined in 2009 when initiating a practitioners’

---

2Software-intensive organisations are those that take part in the task of developing software-intensive systems. According to ISO/IEC 4210:2011, software-intensive systems are those in which software ‘contributes essential influences to the design, construction, deployment, and evolution of the system as a whole’.

3Need for Speed (N4S) research program is briefly explained in Chapter 3 and in details at http://www.n4s.fi/en/.

4Software development and operations are described in detail in Chapter 2.
event called DevOps Days. The event was arranged by Patrick Debois, a leading pioneer of the DevOps movement, who had recognised a growing interest amongst other practitioners on the need to improve the process of delivering software quickly to production (Debois 2011). Debois’ recognition for the interest in DevOps came about after giving a presentation at a 2008 agile conference titled ‘Agile infrastructure and operations’ (Debois 2008), which focused on applying agile principles to the management of infrastructure-related activities and projects. In his paper, Debois (2008) reported some useful patterns for managing software infrastructure in an agile way across different projects. In addition, a few months prior to DevOps Days event, two other pioneers, John Allspaw and Paul Hammond, shared their experiences of multiple deployments in a presentation titled ‘10 deploys per day: Dev & ops cooperation at Flickr’ at a 2009 velocity conference. Since its inception, DevOps has continued to gain more traction in the software industry, mostly occurring in practitioner-driven meetings, presentations and events, but with much debate surrounding the concept. In academia, research on DevOps seemed to be lagging behind, and not only in validating its claims (Dingsøyr & Lassenius 2016) but also in addressing many research challenges it has brought about, such as determining what practices are advocated and for which kinds of software products or organisations (Zhu et al. 2016).

1.2 Research challenges and questions

The first research challenge relates to ambiguity in the concept of DevOps and the subsequent need to understand its underlying assumptions that would be beneficial to assess its practices (Dingsøyr & Lassenius 2016). This research challenge is also important to stir a discussion on whether the concept and the underpinning assumptions of DevOps are novel or antecedents to the established knowledge base of software development methods. This research challenge was also true for the agile concept in software development in the early years of its introduction (Dingsøyr et al. 2010).

At the cornerstone of the DevOps concept is the early involvement of the operations process into the software development process. According to Hamilton (2007), most software issues that occur in production, such as software failures, originate or are best

---

5DevOps Days is a worldwide series of technical conferences covering topics of software development and IT infrastructure operations (http://devopsdays.org/about/).
6Velocity is an O’Reilly conference, which in 2009 was themed ‘web performance and operations’. (https://conferences.oreilly.com/velocity/velocity2009/public/content/about).
solved in the activities of the software development process. Hamilton (2007), based on his experience, observed that the low administration cost of software in production highly correlates with how close software development, testing and operations team work together. Prior literature on cooperation between software development and operations in software-intensive organisations reports of very few studies that are existing in the area (Nelson et al. 2000, Tessem & Iden 2007, 2008). In addition, several empirical studies report on poor cooperation and lack of early involvement of operations into the software development process, which causes several problems (Gotel & Leip 2007, Hamilton 2007, Iden et al. 2011, Nelson et al. 2000, Strode et al. 2012, Tessem & Iden 2007, 2008). A comprehensive list of the problems in the poor cooperation between software development and operations, gathered from a Delphi study of 42 Norwegian experts, is provided in the study by Iden et al. (2011). From the study, the most serious problems in poor software development – operations cooperation included (Iden et al. 2011):

- Operations not being involved in the requirements specifications
- Poor communication and information flow
- Unsatisfactory test environment
- Lack of knowledge transfer
- Systems being put into production before they are complete
- Operational routines not being established prior to deployment

Most of the problems in poor software development – operations cooperation provided by Iden et al. (2011) are supported by the other studies (Gotel & Leip 2007, Hamilton 2007, Strode et al. 2012). For instance, Hamilton (2007) reports on the problem that software engineers do not often take the time to think upfront about the impact of new software features on the load or the rest of the infrastructure. The problems for the lack of cooperation and early involvement of operations seem to be exacerbated when agile software development methods are being used by software development teams (Elbanna & Sarker 2016, Gotel & Leip 2007, Strode et al. 2012). For the latter studies, the concept of boundary spanners has also been used to describe collaboration of the software development team with other stakeholders other than the customer, such as the operations team (Baskerville et al. 2010, Nguyen-Duc et al. 2014, Strode et al. 2012). Interestingly, studies on boundary spanning report it as an area that is not adequately addressed in research, despite being crucial in agile software development projects.
(Strode et al. 2012). It should be noted that the term DevOps was not used in these early studies; rather, different terms were used, such as transfer by Tessem & Iden (2007).

From the early studies on cooperation between software development and operations, their main conclusion is that operations are important stakeholders whose involvement needs to occur throughout all activities of the software development process, particularly in design, testing and deployment (Hamilton 2007, Iden et al. 2011, Tessem & Iden 2007). This conclusion of the early studies give support to important underlying assumption of DevOps concept that posits to establish effective collaboration between software development and operations (Debois 2011), which agile software development had only emphasised on the customer, as illustrated in Figure 1. The present research was initiated in January 2014. At the time, very few studies were published on the topic of DevOps and controversy surrounded the concept, such as whether or not there is a DevOps engineer role or team (Earnshaw 2013, Humble 2012, Knupp 2014). As such, it is of interest in academia and software industry to not only understand precisely the concept of DevOps through an acceptable definition Dyck et al. (2015), but also its recommended practices.

Fig. 1. DevOps encourages collaboration between software development and operations (adapted from Rautonen 2013).

The second research challenge concerns about how DevOps is implemented in practice. As there is no generally agreed framework for DevOps adoption and implementation, the manifestation of DevOps concept varies from one organisation to another. From a conceptual understanding, software practitioners of the DevOps movement specify moving away from software development approaches that encourage the separation of software development and operations teams into silos. In such a setting, there is typically a hand-over of new software features to operations team from the software development team. Software-intensive organisation like Netflix upon employing DevOps report on not having a dedicated operations team (Parnin et al. 2017),

19
while others have created an entirely new role or team with the title of DevOps engineer (Kerzazi & Adams 2016). Discouraging silos in practice means making changes to not only the existing organisational structures, but also to the practice of releasing and operating software in production. The best practices for designing and deploying operations-friendly software products and services have been reported by Hamilton (2007) but without the knowledge of their applicability at a larger scale in software industry. In academia, not until the year 2016 had a significant number of studies on DevOps implementation in industry emerge. A large portion of the studies at that time came from the IEEE Software special issue on *Software Engineering for DevOps* of 2016 (Zhu et al. 2016). Evidently, there is a need for more empirical evidence and details of practices introduced by DevOps and contexts in which the practices apply (Zhu et al. 2016). This is in addition to identifying the key barriers between software development and operations, and the ways in which the barriers can be removed through the DevOps approach in continuous software engineering activities (Fitzgerald & Stol 2017). The need comes, especially after the early studies on DevOps have reported its affects on organisational structure (Nybom et al. 2016), software architecture (Balalaie et al. 2016) as well as software development and deployment processes (Feitelson et al. 2013). Qualitative analysis of DevOps in software-intensive organisation present the opportunity to provide additional empirical evidence with thick descriptions of the practices.

The third research challenge relates to the identification of benefits and challenges that software-intensive organisations are likely to experience when adopting and applying DevOps practices. Existing research on collaboration is software development projects report on the positive impact of careful coordination between several individuals across the different stages of software development process on software project success (Cataldo & Herbsleb 2013, Herbsleb & Roberts 2006). In particular, collaboration is reported to have an impact on software development time, software quality and developer productivity (Cataldo & Herbsleb 2013, Herbsleb & Roberts 2006). DevOps emphasises on collaboration, therefore it is interesting to investigate impacts of increased collaboration achieved through DevOps on software project. In addition, the benefits of a particular software development practice are amongst the key drivers for adopting the practice. In fact, benefits experienced by smaller groups provide a business case for adopting the practice on a larger scale within organisations, in addition to gaining management-level support of the practice. On the other hand, early studies have reported the adoption and implementation of DevOps in software organisations as not always
smooth, and some challenges may be experienced during the transition as well as after the transition (Ebert et al. 2016, Zhu et al. 2016). As such, in addition to investigating the benefits of DevOps, the thesis investigates the challenges pertaining to the adoption and implementation of DevOps.

The specified research challenges have elaborated on the need to explore and develop a better understanding of the adoption and implementation of DevOps in practice. Therefore, the aim of this thesis is to provide a deep understanding of the adoption and implementation of DevOps in practice. Specifically, the thesis explores the adoption of DevOps from the software practitioners’ understanding of the concept, practices, benefits and challenges. The main research question of the thesis is: **How is DevOps understood, implemented, and perceived in software development practice?**

The main research question is further divided into the following sub-research questions:

- **RQ1:** What is the understanding of the DevOps concept in software development practice?
- **RQ2:** What practices are found in DevOps implementation in software development practice?
- **RQ3:** What benefits and challenges of DevOps are perceived in software development practice?

### 1.3 Research scope

The aim of the thesis is to provide a thorough description of DevOps adoption and implementation in software development practice, by comprehending the concept’s definition, practices, benefits and challenges. The scope of the research, illustrated in Figure 2, is on the state-of-the-art and the state-of-the-practice of DevOps. Descriptions of the DevOps concept, practices, benefits and challenges are aggregated from existing scientific studies and non-scientific documents, such as online blog posts and white papers. In addition, thick description of the DevOps concept, practices, benefits and challenges is gathered through empirical investigation using qualitative inquiries in nine software-intensive organisations.

### 1.4 Outline of the thesis

This thesis contains six chapters, whose contents are logically structured as follows. In the first chapter, the research motivation, challenges, questions and scope are described.
The research motivation briefly describes the background of the research area. The research challenges are used to further elaborate on the background of the research area by introducing research problems and research questions of the thesis. The research scope is presented last to map the boundaries of the research area that are covered by this thesis.

The second chapter presents the background and existing literature that falls within the scope of thesis research area. Chapter 2 focuses on the description of the DevOps concept, and its key characterising elements, practices, benefits and challenges. It gives empirical evidence that the adoption of DevOps is grounded upon, and wherein the knowledge contributed by this thesis is supported and new insights added.

In Chapter 3, the research methods used to consolidate the knowledge of the adoption of DevOps in practice are presented. The chapter includes descriptions of the overall research design and research process, which includes activities of data collection and analysis. The descriptions of the research design and process clarify the rationale for the
selection of the research methods and the steps taken to conduct the research activities. In the last section of the chapter, a description about the synthesis of the findings of the thesis is given.

In Chapter 4, the findings of the thesis, which are synthesised from five original publications, are presented. The contributions and findings of the original publications are first presented individually in separate sections. The first publication briefly presents the practices of DevOps from existing literature and identifies areas wherein additional inquiries from software practitioners should be made. The second publication focuses on understanding the concept of DevOps from the perspective of existing and new software development methods in addition to identifying the benefits of DevOps. The third publication focuses on understanding the concept and practices of DevOps by software practitioners without specifying the context wherein DevOps is adopted. The fourth and fifth publications present findings of the adoption of DevOps in two different contexts, and investigate the practices, benefits and challenges of DevOps. In addition, in the fifth publication, findings about the understanding of the DevOps concept are presented. At the end of the chapter, a summary of the answers to the research questions of the thesis is presented.

In Chapter 5, the findings of the thesis are discussed in relation to related work in order to also identify and present the implications of the findings to practice and research. Chapter 6 concludes the thesis report by presenting the main contributions and the validity and limitations of the contributions. In addition, areas for future research are identified and presented.
2 Background and related work

The purpose of this chapter is to introduce the concept of DevOps as well as its adoption, implementation, benefits and challenges in software-intensive organisations based on existing literature. Most of the empirical studies about the adoption of DevOps were published in parallel or after publishing the findings of the thesis, and as such, they are considered related work.

2.1 The DevOps concept

In this section, proposed definitions and key characterising features of DevOps are presented. At the start of this research in 2014, only one systematic mapping study had been done on DevOps to explore the meaning of the concept and provide its empirical evidence (Erich, Amrit & Daneva 2014). The study reported on the existence of very few studies on the topic of DevOps. Other literature reviews were done at the end of 2014 to complement the first systematic mapping study with additional primary studies (Lwakatare et al. 2015, Smeds et al. 2015). Over the course of the study from 2014 to 2017, more literature reviews emerged, including reviews of non-scientific documents, such as online blogs, which provided more evidence of the concept (de França et al. 2016, Hussain et al. 2017, Jabbari et al. 2016, Kerzazi & Adams 2016). This section reviews these works as well as others that have described the concept without necessarily involving an explicit analysis of prior literature, such as the works by Bass et al. (2015), Dyck et al. (2015), and Penners and Dyck (2015).

2.1.1 Definitions of DevOps

There is no consensus among software practitioners and scholars of what the DevOps concept entails (de França et al. 2016, Lwakatare et al. 2015, Smeds et al. 2015). In the literature, several definitions of DevOps have been presented (see Table 1). The majority of the descriptions specify DevOps as a term that is used to emphasise the collaboration between software development and operations. The description of software development and operations used in this thesis and presented in Figure 3, is derived from ISO/IEC standards and empirical studies describing software development and operations processes.
According to system and software life cycle standards used for designing software-intensive products and services (ISO/IEC24748-1 2011), technical processes and corresponding activities in software development include: software requirements analysis, software architectural design, software detailed design, software construction – coding and testing, software integration and software qualification testing (ISO/IEC12207 2008). Meanwhile, the software operations process focuses on the practical use of the system, covering the operation of the software product as well as operational support to users (ISO/IEC15288 2008). The activities of the operation process include release and activation of software product for operational use and establishing procedures for testing software product in the operational environment, recording and resolving problems and modification requests of software product (ISO/IEC12207 2008). Looking at organisational structures and activities of software development and operations that are used in practice, i.e. real-life setting, empirical studies report the existence of software operations as a separate organisational unit from software development (Gotel & Leip 2007, Tessem & Iden 2008, Nelson et al. 2000). In some cases, the software operations function is outsourced to another company offering such services (Nelson et al. 2000, Tessem & Iden 2008). Concerning activities of the software operations, those reported in empirical studies include: performing acceptance tests, installing updates and conducting daily operations of software product, as well as handling upcoming errors (Tessem & Iden 2008).

---

**Software development (Dev)**
A process for creating a software product consisting of inter-related activities of software requirements analysis, architectural design, coding, testing, integration and acceptance testing

**Software operations (Ops)**
A process of putting into use, and supporting end user(s) in the use of software product in operational environment. Its activities include: installation, upgrade, migration, operational control, monitoring, configuration management, alerting, availability and support

---

Fig. 3. Description of software development and operations.

---

7Based on ISO/IEC 15288, System and software engineering – System life cycle processes and ISO/IEC 12207, Systems and software engineering – software life cycle processes standards. The ISO/IEC 15288 standard describes the processes at the system level and ISO/IEC 12207 applies the same processes to software.
### Table 1. Proposed Definitions of DevOps in the Literature.

<table>
<thead>
<tr>
<th>Source</th>
<th>Definition of DevOps</th>
<th>Focus of Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Bass et al. 2015)</td>
<td>‘DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production while ensuring high quality’</td>
<td>Goal-oriented (fast delivery of quality software)</td>
</tr>
<tr>
<td>(Dyck et al. 2015)</td>
<td>‘DevOps is an organisational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organisations, in order to operate resilient systems and accelerate the delivery of changes’</td>
<td>Means-oriented (empathy, cross-functional collaboration); and goal-oriented (operate resilient systems, accelerate change delivery)</td>
</tr>
<tr>
<td>(Penners &amp; Dyck 2015)</td>
<td>‘DevOps is a mindset, encouraging cross-functional collaboration between teams – especially development and IT operations – within a software development organisation, in order to operate resilient systems and accelerate the delivery of changes’</td>
<td>Means-oriented (engineer capabilities)</td>
</tr>
<tr>
<td>(Smeds et al. 2015)</td>
<td>‘A set of engineering process capabilities supported by certain cultural and technological enablers’</td>
<td>Means-oriented (engineer capabilities)</td>
</tr>
<tr>
<td>(de França et al. 2016)</td>
<td>‘DevOps is a neologism, representing a movement of ICT professionals addressing a different attitude regarding software delivery through the collaboration between software systems development and operations functions, based on a set of principles and practices, such as culture, automation, measurement and sharing’</td>
<td>Means-oriented (attitude, cross-functional collaboration)</td>
</tr>
<tr>
<td>(Jabbari et al. 2016)</td>
<td>‘DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasising communication and collaboration, continuous integration, quality assurance and delivery with automated deployment, utilising a set of development practices’</td>
<td>Means-oriented (cross-functional collaboration, automated deployment)</td>
</tr>
</tbody>
</table>

Regarding the common description of DevOps, some scholars have argued against limiting the definition of DevOps to just collaboration. Bass et al. (2013) argue that collaboration is a vague term, and as such, software developers can use sources, such as standards, organisational process descriptions and academic studies, to determine the requirements and actions needed to support software operations activities. Furthermore, Bass et al. (2015) argue that there is an unestablished causal link between the different forms of collaboration and the release cycle time or quality of software changes in production. Meanwhile, DevOps has also been used to represent several other things,
including a movement in the software industry (de França et al. 2016) as well as a role in software organisations (Hussain et al. 2017, Kerzazi & Adams 2016).

Comparing the definitions proposed by scholars, presented in Table 1, differences in the descriptions of DevOps can be observed with respect to the focus. From the descriptions of DevOps, the focus is given either to the goal or the means for achieving collaboration between software development and operations. The definition provided by Bass et al. (2015) is goal-oriented as it puts the emphasis on speeding up the delivery of quality software, which is achieved by employing a set of engineering practices from the time the software developer commits code up until the point when the code is deployed to production. Without being specific on the means, Bass et al.’s (2015) definition considers DevOps as any engineering practice that is found from code commit to production deployment. The latter is regardless of whether it involves an agile method or other form of coordination. On the contrary, the definition by Dyck et al. (2015) focuses on the means, in particular, cross-functional collaboration and empathy. In their extended report (Penners & Dyck 2015), where the authors perform an inquiry among software practitioners, including the pioneers of the DevOps movement and other scholars, they argue that the goal to be achieved through DevOps is completely missing or that strong differences may be observed. Through their correspondence with practitioners and scholars, the differences on the intended goal for DevOps were observed to range from the ability to quickly deploy quality software and accelerate feedback loops to the ability to just operate the software in production more resiliently. Despite the differences in the goals, their definition does include the goal of software development — operations collaboration. Meanwhile, the definition by Smeds et al. (2015) is largely means-oriented than goal-oriented, even though the engineering capabilities are not explicitly described in the definition. The latter is due to the fact that the authors had specified in their report that focusing on culture to define DevOps is not necessarily helpful in the software engineering field (Smeds et al. 2015). DevOps definition by de França et al. (2016) is means-oriented, and similar to that of Penners & Dyck (2015), it specifies attitude change and cross-functional collaboration. Lastly the definition provided by Jabbari et al. (2016) is means-oriented, in particular, collaboration, continuous integration, quality assurance and automated deployment.

The problem with the definitions by Bass et al. (2015) and Smeds et al. (2015) is that they do not specify practices explicit to DevOps, which might be useful for distinguishing the concept. Meanwhile, the definitions that somewhat describe the practices have great differences (de França et al. 2016, Dyck et al. 2015, Jabbari et al. 2016, Penners &
Dyck 2015). Moreover, differences in the practices can be further observed in studies that have explored what software organisations consider when it comes to the role of the DevOps engineer (Kerzazi & Adams 2016, Hussain et al. 2017). Scholars of the latter studies have identified the importance of infrastructure-related activities, such as monitoring, provisioning and managing environments used for software development, testing, pre-production and production (Kerzazi & Adams 2016, Hussain et al. 2017). Considering that Penners & Dyck (2015) make the first attempt to develop a scientific definition of DevOps, in this thesis their definition is used as a starting point and is explored further by making additional inquiries from software practitioners. The definition is also selected because compared with the other definitions, theirs has more inputs coming from software practitioners, including pioneers of DevOps, while also soliciting other scholars to improve on their definitions.

2.1.2 Key elements of DevOps

The key elements of DevOps constitute important features that characterise the concept. The key elements show and describe the different facets of DevOps, which can be identified from problems caused by poor cooperation between software development and operations, and the actions taken to address the problems. Through the different facets, software practitioners can begin to discuss the concept of DevOps, leading to a shared understanding of DevOps and identifying areas to begin to implement DevOps. The elements are captured by Lwakatare et al. (2015) as dimensions of DevOps. Table 2 summarises common key elements of DevOps reported in the literature that include collaboration/sharing, monitoring, measurement, automation and culture (de França et al. 2016, Erich et al. 2014, Humble & Molesky 2011, Lwakatare et al. 2015).

The problems addressed by DevOps are reported by Humble & Molesky (2011), and are supported by several pieces of empirical evidence in the literature (Bass et al. 2015, Elbanna & Sarker 2016, Gotel & Leip 2007, Iden et al. 2011, Tessem & Iden 2007). The problems fall into the following categories: (1) division and communication gap between software development and operations; (2) manual and bureaucratic steps in the processes of deploying software changes to end users; and (3) increased complexity and heterogeneity of large-scale infrastructures affecting software design and the management of software infrastructures. The problems have been discussed previously in Section 1.2.
Table 2. Key Elements (characteristics) of DevOps Reported in Literature.

<table>
<thead>
<tr>
<th>Key DevOps element</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>Automation of build, testing and deployment, particularly the implementation of the deployment pipeline.</td>
<td>(de França et al. 2016, Erich et al. 2014, Humble &amp; Molesky 2011, Lwakatare et al. 2015)</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Systems are designed to expose relevant information. Use of analytics to integrate the system and infrastructure performance data with customer-usage behaviour</td>
<td>(Humble &amp; Molesky 2011, Lwakatare et al. 2015)</td>
</tr>
<tr>
<td>Collaboration or sharing or culture</td>
<td>Collaboration enforced through information sharing, broadening of skillsets, shifting responsibilities and instilling a sense of shared responsibility between development and operations</td>
<td>(de França et al. 2016, Erich et al. 2014, Humble &amp; Molesky 2011, Lwakatare et al. 2015)</td>
</tr>
</tbody>
</table>

2.2 The implementation of DevOps

In this section, the adoption and state-of-the-practice of DevOps are presented. No generally accepted framework exists for DevOps adoption and implementation. In addition, very few empirical studies describe DevOps practices. Evidence of the adoption of DevOps can be found in empirical studies on continuous deployment (Fitzgerald & Stol 2014) and rapid releases (Kerzazi & Khomh 2014), but such studies tend to have a narrow focus on the specifics of DevOps. The implementation of DevOps in software organisations is not homogeneous due to many differences, such as the objectives and the starting baseline of the organisation among others. DevOps implementation entails a series of organisational and socio-technical practices employed by software organisations to facilitate collaboration between software development and operations.
2.2.1 **Industrial adoption of DevOps**

Surveys of the state of DevOps have been conducted annually for the past five years mainly by software practitioners such as Puppet (Puppet 2013, 2014, 2015, 2016). The surveys help explain the adoption of DevOps in the software industry. These surveys have sometimes been administered by pioneers of DevOps, such as Jez Humble and Gene Kim who wrote influential books about the practices in continuous delivery (Humble & Farley 2010) and DevOps (Kim *et al.* 2013). Despite the lack of a rigorous research method, the surveys have over the years receive a large number of responses from software practitioners of different organisations across the globe. As such, they are valuable for providing information on the state-of-the-practice but caution is taken when presenting their results. Particularly, claims about the adoption of DevOps, its implementation, benefits and challenges described in the surveys are narrowly presented here, while detailed explanations of these aspects are sought from empirical studies published in scientific venues.

The findings from the state of the DevOps surveys have reported a continued increase in the adoption rate of DevOps over the years since the first report published in 2013 (Puppet 2013, 2014, 2015, 2016). According to the survey by Gleanster & Delphix (2015), most DevOps initiatives are done by small teams of experts in deployment automation, data management and continuous integration. The top three reasons for adopting DevOps, which also ultimately constitute its benefits, include: fast delivery of software; faster identification of bugs; and frequent delivery of software\(^8\) (Gleanster & Delphix 2015, Puppet 2013). The 2014 survey by Puppet correlated the benefits of DevOps with business value, specifically organisations’ profitability, market share and productivity (Puppet 2014). The findings of the survey report that companies practicing DevOps see improvements in IT performance, and consequently, business value, since IT performance strongly correlates with business performance where it helps to boost productivity, profitability and market share (Puppet 2014). IT performance was measured in terms of deployment frequency, lead time of changes and mean time to recover from a failure. In the 2015 survey (Puppet 2015), claims of an improved organisational culture and employee engagement as benefits of incorporating DevOps practices in the organisations’ processes are reported.

\(^8\)Fast delivery of software, measured in terms of release cycle time, means a reduction in the overall time taken to release software from the time when requirements are specified. Frequent delivery of software, measured in terms of deployment frequency/rate, means the total number of releases per specified unit time.
Empirical studies describing the use of DevOps in the development of different software applications, presented in Table 3, report similar results with respect to the drivers of DevOps adoption. Particularly, they report the need to reduce the release cycle time (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017, Jones et al. 2016). The other drivers reported in the studies included increased availability of technologies such as Docker and the use of cloud computing in embedded system domains, dislike of manual and lengthy deployment procedures, as well as increased industrial talks about DevOps. Jones et al. (2016) reported that the adoption of DevOps is championed by the software development manager, even though there is little interest coming from senior management. Interestingly, Fazal-Baqia et al. (2017) report that their adoption of DevOps did not reach an end point, but progressed even after several improvements to the delivery process of software changes. Adoption of the DevOps is observed in the software development activities of different application types, including middleware and software applications for the cloud, configuration management tools and social media applications, e-commerce sites and platforms, and connectivity solutions for automotive vehicles, such as online navigation systems in cars. From the empirical studies, it is clear that companies involved in cloud and web development have been the early adopters of DevOps. Furthermore, DevOps is adopted in different organisation sizes and contexts, ranging from single organisations of small/large sizes to complex multi-provider environments. In the latter context, a software development project is carried out as multiple sub-projects done collaboratively between teams from the client organisation and several providers, some of which are located offshore (Fazal-Baqia et al. 2017).
Table 3. Application type and drivers of DevOps adoption at organisations.

<table>
<thead>
<tr>
<th>Type of application (organisation)</th>
<th>Driver of DevOps adoption at organisation</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Service PICCO – middleware and software applications for the cloud (Fujitsu)</td>
<td>Increased availability of new technologies like Docker and industrial talks about DevOps</td>
<td>(Elberzhager et al. 2017)</td>
</tr>
<tr>
<td>Facebook – social media web application and configuration tools (Facebook)</td>
<td>Reduction in release cycle time, reliable releases</td>
<td>(Feitelson et al. 2013, Tang et al. 2015)</td>
</tr>
<tr>
<td>Wotif – ecommerce travel website (Wotif)</td>
<td>Reduction in release cycle time, reliable releases</td>
<td>(Callanan &amp; Spillane 2016)</td>
</tr>
<tr>
<td>Ecommerce system and platform (anonymous company in the UK)</td>
<td>Reduction in release cycle time, an enthusiast software development manager</td>
<td>(Jones et al. 2016)</td>
</tr>
<tr>
<td>Web and mobile application for online betting and trading systems (Paddy Power)</td>
<td>Reduction in release cycle time, reliable releases</td>
<td>(Chen 2017)</td>
</tr>
<tr>
<td>Connectivity solution for cars (Elektrobit)</td>
<td>Availability of cloud computing technology</td>
<td>(Schneider 2016)</td>
</tr>
<tr>
<td>Commercial mobile back end as a service (Backtory)</td>
<td>Architectural refactoring to microservices</td>
<td>(Balalaie et al. 2016)</td>
</tr>
<tr>
<td>Web-based digital sales channel (multi-provider project in financial domain)</td>
<td>Apply several improvements to agile practices used in the project</td>
<td>(Fazal-Baqiae et al. 2017)</td>
</tr>
</tbody>
</table>

Elberzhager et al. (2017) start their DevOps adoption by identifying and answering some key questions to which they recommend other software organisations interested in adopting DevOps answer before embarking on the journey; otherwise, resources will be wasted. The key four questions centre on the following topics: (1) recognising goals that are to be achieved with DevOps; (2) choosing a suitable strategy for introducing DevOps; (3) explicitly defining responsibilities for software development and operations; and (4) identifying the impact of DevOps on architecture (Elberzhager et al. 2017). According to the authors (Elberzhager et al. 2017), obtaining a common understanding of DevOps based on state-of-the-art and state-of-the-practice analyses was one of their specific goals for DevOps adoption. The intention was to bring awareness and identify concrete improvement areas offered by DevOps in their context. In addition, they reported that the existing agile processes were a good starting point for the adoption of...
DevOps (Elberzhager et al. 2017). One chosen strategy for DevOps adoption for most empirical studies is to select one or a small group of pilot applications wherein DevOps is trialled (Callanan & Spillane 2016, Elberzhager et al. (2017), Bass et al. (2015, p.239)).

Adoption of DevOps in industrial fields with high requirements for safety, security, traceability and reliability, such as medical devices and automotive vehicles, is scarcely reported (Laukkarinen et al. 2017, Schneider 2016). From the studies (Laukkarinen et al. 2017), observable obstacles to the adoption of DevOps are reported by examining its role in specific standards, such as IEC/ISO standards of medical device and health software. On the contrary, Schneider (2016) reported on their DevOps adoption in the development of a software solution for connected cars. As a company that provides software solutions in the automotive industry, DevOps adoption was driven by the increased use of cloud computing, which is a key technology used in connectivity solutions. The rationale, according to the author (Schneider 2016) was that the expertise in connectivity and back-end infrastructure needed to be incorporated into the development team working in connectivity solutions, thereby enabling the company to cope with the technical challenges of cloud computing.

### 2.2.2 Implementation of DevOps: organisational perspective

From the organisational perspective of DevOps implementation, most of the practices focus on organisational structures, and concern the ways in which organisations deal with the relationships, communication gaps and responsibilities between software development and operations as well as the corresponding cultural changes (Cito et al. 2015, Chen 2017, Jones et al. 2016, Nybom et al. 2016, Shahin et al. 2017b, Smeds et al. 2015). On organisational structure, four practices can be observed from the responses of software practitioners for surveys presented by Cito et al. (2015) and Shahin et al. (2017b). These include: (1) separating software development and operations with high collaboration; (2) separating software development and operations with a facilitator (an individual or team) between the two groups; (3) software development team gaining more responsibility for software operations, and software operations existing in a small team; and (4) no visible software operations team.
In large organisations, the most common practice is reported to be first on the list, whereas the last two are more common in small- and medium-sized organisations (Cito et al. 2015, Shahin et al. 2017b). The latter is also supported in other empirical studies (Callanan & Spillane 2016, Feitelson et al. 2013, Nybom et al. 2016, Schneider 2016, Smeds et al. 2015, Tang et al. 2015). For instance, at Facebook, the practice was adopted by the configuration management team, which consisted of a small group of engineers responsible for implementing configuration tools (Tang et al. 2015) as well as engineers developing a Facebook web application (Feitelson et al. 2013). The engineers in the configuration management team are reported to be responsible for the development and production deployment of new features, monitoring as well as resolving production issues of configuration tools that are used by thousands of Facebook engineers to manage configurations of hundreds of thousands of servers and over one billion mobile devices (Tang et al. 2015). Additionally, through what Feitelson et al. (2013) describe as a culture of personal responsibility, Facebook engineers are reported to take it as a personal stake the task of writing and testing code that must support operational use and keeping the system running smoothly at all times. An alternative practice to software developers gaining more responsibility for operations, particularly in large organisations, was the establishment of a separate team that acted as a facilitator between the two groups. Evidence of the latter is presented by Bass et al. (2015, p.238), Callanan and Spillane (2016), Schneider (2016), Elberzhager et al. (2017), and Chen (2017), who argue for such a team, especially as it would help reduce the significant efforts and avoid disruptions for the software development team. In the large organisations, the authors (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017) report on establishing a dedicated team consisting of a small number of members with multidisciplinary backgrounds, including that of software development and operations. The dedicated team at both organisations are responsible for standardising software deployment mechanisms, such as building a deployment pipeline platform to be used by software development teams (Bass et al. (2015, p.238), Callanan & Spillane (2016), Chen (2017), Elberzhager et al. (2017)). Meetings between the dedicated team and software development and operations teams are reportedly held in order to solicit ideas and communicate the new practices (Callanan & Spillane

---

9The sizing of the organisations is adopted from the studies (Cito et al. 2015, Shahin et al. 2017b). A similar scale is used in both studies, though Cito et al. (2015) do not distinguish between medium and small organisations; large organisation > 1,000 staff, medium = 100-1,000 staff and small < 100 staff.

10Deployment pipeline is described below in Section 2.2.3
In some cases, the dedicated team is at first initiated by a consulting company tasked to design and deliver deployment pipeline fitting to the customer organisation and culture. Upon completion the consulting company identifies and trains personnel to bear the responsibilities of designing and managing the deployment pipeline platform (Bass et al. 2015, p.238). One thing to notice when knowledgeable software developers are given the responsibility to perform operations tasks is that they tend to be given access to a critical environment (Nybom et al. 2016, Schneider 2016). Such access allows software developers to deploy the new software features to different software environments, including the production environment which is a controlled environment and traditionally all alterations tend to go through a change control process and board (Kim et al. 2013).

2.2.3 Implementation of DevOps: socio-technical perspective

On the socio-technical aspects of DevOps implementation, practices with respect to defined steps, technical infrastructure and human agents involved in software development and operations are central (Cito et al. 2015). For cloud applications, in a survey of software practitioners by Cito et al. (2015), the authors report such practices upon identifying areas of software development changing by cloud computing. The practices fall into the following areas: (1) software deployment and provisioning; (2) software design and implementation; (3) software troubleshooting; (4) software monitoring and production data usage; and (5) tooling (Cito et al. 2015). Given that the early and prominent adopters of DevOps are software companies doing cloud and web applications, evidence of the practices are also reported in other DevOps studies (Callanan & Spillane 2016, Elberzhager et al. 2017, Feitelson et al. 2013). The specifics of the practices differ from one organisation to another; however, a common practice observed is the implementation of a deployment pipeline, illustrated in Figure 4 (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017).

The deployment pipeline entails the path taken by and the activities done to software changes on their way to production. The deployment pipeline incorporates the many aspects of software development that are not often focused on by the software development team. As such, the deployment pipeline is where the cross-cutting concerns and knowledge between software development and operations intersect (Bass et al. 2015). The deployment pipeline creates visibility to the software delivery process and allows instant feedback to be given with regards to the impact of the software changes.
on their production readiness. In addition, the deployment pipeline provides control over the software delivery process, enabling the software development and operations team to quickly release any chosen version of software into the environments that the teams manage. The high level overview of workflow activities in the deployment pipeline involve, at first stage, code commit by an individual software developer to a version control system in a development environment that is shared by all team members. Feature branches can be used to realise the task of making software change. At the next stage, continuous integration monitors the code commits to validate code changes against automated tests, such as unit and integration tests, and creates a build as an output if all tests pass. Thereafter, the build packages that have successfully passed the tests are deployed to subsequent environments and lastly to production environment.

![Deployment Pipeline](image)

**Automation in software deployment**

In their DevOps implementation, Callanan and Spillane (2016) report on standardising and automating software packaging and deployment mechanisms as the most important focus areas. The implementation also included automatic verification of software deployment to verify that the system behaved as expected after each software deployment in all environments (Callanan & Spillane 2016). The automated verification tests were annotated to ensure backward-compatibility of the tests that were run after every commit during deployment to the acceptance environment. When the tests failed, the release candidate would be blocked. Software developers were responsible for load testing as they could better determine how much of the load tests to perform for a given release, rather than relying on the use of heavyweight centralised load testing tools, which were a bottleneck to the release process (Callanan & Spillane 2016). Other forms of non-functional tests are not elaborated further by Callanan and Spillane (2016). An incremental-change approach was adopted to facilitate database schema upgrades using
tools, such as Liquibase, that allow for automated database manipulation during a release (Callanan & Spillane 2016). According to Callanan and Spillane (2016), the incremental-change approach involves decoupling database changes from software feature releases while making the changes forward- and backward-compatible with each other. The practice of automating software deployment allowed developers to get feedback on how their services behaved operationally and could quickly resolve potential issues as they arose.

In multi-provider software development environments where there are more software components and teams involved, it is reported that the long release cycle times increase the risk of failures in the release process (Fazal-Baqaie et al. 2017). In such environments, Fazal-Baqaie et al. (2017) report on implementing a backward-compatible database, and improvements in the release process, whereby software components developed by various vendors are delivered as soon as they are developed, automatically tested and verified for regressions. In addition, container technology, such as Docker, is used for preparing installation packages that are then pushed to various environments in the shortest time (Fazal-Baqaie et al. 2017).

**Automation in infrastructure management**

Among other practices for DevOps implementation, Schneider (2016) reports on employing the practice of *infrastructure-as-code (IaC)* while successfully working with Amazon Web Services (AWS) CloudFormation11 to make configurations reproducible. The practice of IaC for automatic and repeated facilitation of software application deployment and environment provisioning is central to DevOps implementation, evidence of which is reported several times in empirical studies (Cito et al. 2015, Elberzhager et al. 2017, Schneider 2016, Tang et al. 2015). The IaC concept entails handing infrastructure configurations in a similar manner as software application code. This is in addition to the use of programmatic language facilitated by tools to handle and make configurations in an executable manner, such that configurations can be applied to a target system efficiently and repeatedly. With IaC, software development teams are able to provision software development and testing environments that are almost similar to the production environment (Elberzhager et al. 2017). According to Tang et al. (2015),

---

11Amazon Web Services (AWS) CloudFormation is a service available to AWS customers for 'creating, managing, provisioning and updating a collection of related AWS resources in an orderly and predictable fashion' (https://aws.amazon.com/cloudformation/).
IaC has two different forms of realisation, and the main differences between the two forms is whether the focus is on software installation or managing software runtime behaviour. IaC during software installation is realised through the execution of scripts that install software packages on the target environment by using tools such as Chef (Tang et al. 2015). IaC for managing software runtime behaviour is realised through the execution of scripts that push the files to all environments by using tools such as Configurator, which is a holistic configuration management tool used at Facebook (Tang et al. 2015). According to Callanan and Spillane (2016), automation of the release process was orchestrated by applying simple rules that included: (1) keeping changes independent and preventing/minimising manual tests from occurring in a release; (2) no reservation of release slots in the calendar and hiding releases that are ahead of time behind feature flags; and (3) allowing operations to rollback the changes of a release whenever necessary. In addition, the authors report on allowing enough time for software development teams to get acquainted with the new release process, which was crucial, particularly since the deployment automation mechanism was built by a separate team (Callanan & Spillane 2016). In this period of acquaintance, the software development teams were able to bring to the surface issues in the new release mechanism incorporated into the deployment pipelines that required revisions. Monitoring through the use of tools is reported, which became important to verify deployments and identify problems and incidents in production as well as in development environments (Callanan & Spillane 2016, Elberzhager et al. 2017).

**Monitoring of Software and Infrastructure**

Problems that occur in production have their root causes in the decisions made during the software development process, such as appropriate test coverage, misconfigurations (Bass et al. 2013, Hamilton 2007). Cito et al. (2015) propose future research contributions on this aspect that focus on finding effective ways to correlate runtime data with software changes during development time for such purpose as automating troubleshooting activities amongst others. Tang et al. (2015) report on having a practice where, in addition to using monitoring tools, urgent issues were escalated to an on-call developer through automated phone calls. For the non-urgent issues, the on-call developer was reported to attend to the issues when time was available (Tang et al. 2015). Monitoring is also done for the purpose of getting customer feedback on software usage (Elberzhager et al. 2017). Analytical tools are incorporated into monitoring metrics to analyse customer usage.
behaviour of the deployed software. In a multi-provider environment, centralised logging was used for collecting runtime data from all software components. After fulfilling special legal requirements for centralised logging, it was used to create log dashboards for various teams that use them for debugging (Fazal-Baqie et al. 2017). Additionally, Fazal-Baqie et al. (2017) report on implementing monitoring techniques that detect performance issues and, in particular, used the failover clustering\textsuperscript{12} mechanism to automatically restart software components that go offline. When automation fail to restart the components, an emergency team is notified to resolve production incidents. An overview of production incidents and the associated responsible teams of the software components and component providers was maintained in the project (Fazal-Baqie et al. 2017). To test and influence the design of services for high availability and resilience, Netflix employs an approach of 'breaking things in production' (Basiri et al. 2016). In the latter approach, randomly selected virtual machine (VM) instances are terminated, or failures are injected into the production environment to experiment and ensure high reliability, availability and resilience of Netflix services (Basiri et al. 2016). When performing the experiments, software engineers closely monitor metrics that show the overall health and steady-state behaviour of the system, such as stream starts per second, new account signups per second, etc. The metrics in alternative domains can be the number of completed transactions per second or ads viewed per second (Basiri et al. 2016). According to the authors, such metrics have direct links to system availability and are characterised by fine-grained metrics such as CPU utilisation and request latency (Basiri et al. 2016).

**Software architectural design changes**

From a software architecture perspective, DevOps is reported to be implemented in both monolithic (Callanan & Spillane 2016, Elberzhager et al. 2017) and microservice architectures (Balalaie et al. 2016, Fazal-Baqie et al. 2017). The advantages that a microservice architecture has over a monolithic architecture in the context of DevOps is improved flexibility of applications and reduced dependencies between components (Fazal-Baqie et al. 2017). Thereby, microservices — loosely coupled independent services of applications — can be deployed individually without affecting the availability

\textsuperscript{12}Failover clustering mechanisms are high-availability systems that harness redundant computers in groups or clusters that provide continued service when system components fail (from Wikipedia – High-availability cluster).

40
of the entire service (Balalaie et al. 2016). In addition, teams can be organised around individual microservices that communicate via lightweight mechanisms such as RESTful API (Balalaie et al. 2016). Monitoring of software is reported to be important as well as the use of containers for bridging the gap between development and production (Balalaie et al. 2016). Empirical studies report on microservices as a separate initiative requiring consideration of several aspects, such as the amount of effort required to create its foundation and a supporting organisational structure (Elberzhager et al. 2017). Therefore, for the purpose of scoping this thesis, such an initiative is not investigated in-depth.

2.3 Benefits and challenges of DevOps

In this chapter, the benefits and challenges of DevOps are presented. The challenges of DevOps are presented in two categories. First are those that present barriers/obstacles/hindrances to its adoption. In the second category are the challenges stemming from the implementation of DevOps. The benefits of DevOps are those pertaining to successful implementation of DevOps in software organisations.

2.3.1 Benefits of DevOps

Several benefits of DevOps implementation are reported in the empirical studies and summarised in Table 4. The benefits include a reduction in the average release cycle time, improved quality and eased tensions between software developers and operations personnel.

*Improved release cycle time*

A common benefit of DevOps implementation is fast releases achieved through shortening release cycle times (Callanan & Spillane 2016, Elberzhager et al. 2017, Fazal-Baqie et al. 2017). Callanan and Spillane (2016) reported a reduction of the average release cycle time from two weeks to one day. Teams were able to deploy software changes as soon as they were ready, thereby, minimising their inclination to bundle many changes into one large batch. Not releasing in large batch sizes helped to minimise risks through the early identification of defects for which hotfixes could be released faster, owing to the streamlined and automated release process. To make the release progress visible to
Table 4. Benefits of DevOps reported in literature.

<table>
<thead>
<tr>
<th>Description</th>
<th>Actual measure/description (Source)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced release cycle time</td>
<td><em>About 20 mins</em> deployment to production, depending on exploratory tests and upload time to cloud (Elberzhager et al. 2017)</td>
</tr>
<tr>
<td></td>
<td><em>1.1 days</em> average release cycle time, an 85% reduction in release cycle time (Callanan &amp; Spillane 2016)</td>
</tr>
<tr>
<td></td>
<td><em>2–5 days</em> release cycle time from conception of user story to production (Chen 2017)</td>
</tr>
<tr>
<td>Improved system quality</td>
<td>50% reduction in releases with potential customer impacting problems (Callanan &amp; Spillane 2016)</td>
</tr>
<tr>
<td></td>
<td><em>Teams use common artifact repositories and common Docker base images, which enable quick reaction to security vulnerabilities and to performance issues</em> (Fazal-Baqie et al. 2017, p.20)</td>
</tr>
<tr>
<td>Improved development and operation collaboration</td>
<td><em>... development and operation engineers collaborate more strongly. This led to requirements from operations that can now be considered and incorporated earlier...</em> (Elberzhager et al. 2017, p.41)</td>
</tr>
<tr>
<td></td>
<td><em>... we decided to add a team member who had many years of experience and deep knowledge in system administration and operations... we not only accelerated progress in those work areas but also eased the tension with the system engineering team</em> (Chen 2017, p.76)</td>
</tr>
</tbody>
</table>

The software development team, automation was built to publish progress messages to a shared chat room (Callanan & Spillane 2016).

Some of the empirical studies have specified the metrics that they used to assess the benefits of DevOps implementation. Following their DevOps implementation (Callanan & Spillane 2016), cycle time metrics for each release were automatically generated from release sign-offs, time stamps and labels that were annotated in release notes of various release activities. To assess their DevOps adoption, Elberzhager et al. (2017) report on having specified 76 metrics from which the top six metrics were identified after refinement. The top six metrics for which concrete calculation rules were reported as defined but not elaborated in the paper included *deployment time and frequency, mean time to recover, developer productivity, time to market of new software features, feedback cycle and expense.*
**Improved quality of system**

Improved product quality is another commonly reported benefit of DevOps implementation (Callanan & Spillane 2016, Elberzhager et al. 2017, Fazal-Baqie et al. 2017). This is related to the early identification of bugs (Callanan & Spillane 2016) through improved quality assurance practices, such as automating the software test (Fazal-Baqie et al. 2017). Fazal-Baqie et al. (2017) report on using quality guards at different stages of the deployment pipeline, from unit tests to integration tests to monitoring in production, in order to ensure that requirements are met and that proper measures are applied when requirements are not fulfilled.

**Improved collaboration between developers and operations**

DevOps implementation is also reported to bring about cultural and organisational benefits (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017, Schneider 2016). Developers and operations are observed to collaborate more strongly and that the right expertise is embedded in the software development team, especially when the team has to cope with operations-related tasks (Callanan & Spillane 2016, Chen 2017, Elberzhager et al. 2017, Schneider 2016). The latter helped to ease communication between software development and operations as well as improve the work morale of the teams, especially since teams could release software frequently and with more confidence.

The improved cultural changes as a benefit of DevOps were reported difficult to measure, despite being recognised as very important (Elberzhager et al. 2017). In the report by Elberzhager et al. (2017) they were reported to be indirectly captured through questionnaires that gathered feedback of the experiences of DevOps implementation from people working on software development using DevOps approach (Elberzhager et al. 2017).

**2.3.2 Challenges of DevOps**

The challenges of DevOps are observed in those pertaining to its adoption and its implementation. Table 5 summarises the challenges.
Table 5. Challenges of DevOps reported in literature.

<table>
<thead>
<tr>
<th>Description</th>
<th>Actual description (Source)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of common understanding for DevOps concept</td>
<td>‘DevOps is understood very differently among the people asked to define it. There are many different, and sometimes false expectations in DevOps...A different understanding of the goals to be achieved with DevOps within a project may end up in chaos and frustration’ (Elberzhager et al. 2017, p.41)</td>
</tr>
<tr>
<td>The need to learn about new technologies parallel to performing daily tasks, particularly for software developers</td>
<td>‘... DevOps adoption is further compounded for the new system given the necessity for both the software developers and systems administrators to learn new technologies, tools and methods’ (Jones et al. 2016, p.9)</td>
</tr>
<tr>
<td>Limitations in technology and overall quality of legacy systems</td>
<td>‘Of more concern is the limitation of the legacy system to deprecated technologies potentially harming the new system development due to the need for developers to “switch” between technologies’ (Jones et al. 2016, p.9)</td>
</tr>
<tr>
<td>Difficulties in managing database schema upgrades in fast-paced release process</td>
<td>‘...most releases requiring database changes employed the traditional release process’ (Callanan &amp; Spillane 2016, p.59)</td>
</tr>
</tbody>
</table>

Lack of common understanding of DevOps concept

Despite having established a clear goal of understanding DevOps from state-of-the-art analyses prior to embarking on a DevOps journey, one obstacle that was identified in the adoption was that people understood the concept of DevOps differently. As a result, at times there were some false expectations for DevOps from the different stakeholders, which brought about chaos and frustrations (Elberzhager et al. 2017).

Blurred responsibilities between software development and operations

The close collaboration of software development and operations was reported to blur responsibilities between software development and operations (Nybom et al. 2016). As such, the absence of a shared code of conduct that involves a clear assignment of responsibilities causes the new culture not to be well understood or appreciated by those affected, including stakeholders in software development with the management role, such as product and project managers (Elberzhager et al. 2017, Jones et al. 2016). Additionally, the DevOps practice of mixing responsibilities of development and operations serves as a new source of friction, especially in the absence of trust (Nybom
et al. 2016). Lacking of skills in operations tasks amongst software developers limits operations personnel from fully relinquishing their responsibilities to software engineers, e.g. insist on limiting access to production (Nybom et al. 2016).

**Difficulties in managing database schema upgrades in rapid release cycle**

Difficulties in managing database schema upgrades in rapid release cycle is reported, by Callanan and Spillane (2016), to cause some software development teams maintain traditional release mechanisms and cycles rather than adopt an automated incremental-change approach to database upgrades (Callanan & Spillane 2016).

**Technology maintenance difficulties in legacy systems**

Dealing with a legacy system that is highly coupled and lacks proper documentation was identified as another challenge in the adoption of DevOps (Jones et al. 2016). While agile methods were reportedly used to maintain the legacy system, according to Jones et al. (2016), the software developers were concerned with the deprecated technologies in the legacy system, which required them to switch between technologies that were used in implementing new software features.

**Lack of senior management support**

The lack of support from senior management and business case analysis of DevOps adoption is reported to undermine effective implementation of DevOps, despite having such an initiative championed by an enthusiast software development manager (Jones et al. 2016). Two consequences were observed as a result of the latter. First, the extent to which the software development manager could support the initiative was limited to only individual employees in software development and operations, and changes could not be experienced by the entire organisation (Jones et al. 2016). Second, senior management still needed to be convinced about the benefits of the DevOps initiative if it was to be implemented in other parts of the organisation (Jones et al. 2016).
Resistance by operations team

Resistance by operations personnel has also been identified as a challenge of DevOps adoption (Jones et al. 2016). Resistance was observed in the form of objections raised by operations in using certain technologies proposed by software development as well as in the perception that software developers would have to do everything on their own (Jones et al. 2016).

Challenges of DevOps in high regulated domains

Laukkarinen et al. (2017) report the adoption challenges of DevOps in industrial fields with high requirements for safety, security, traceability and reliability. The authors report challenges in the context of medical device software development after having examined the role of DevOps with two IEC/ISO standards of medical device and health software. The challenges are described in the paper as areas of conflict as there was incompatibility between DevOps and the requirements specified in the clauses of the standards. The main findings were that the standards need to pay special attention to the use of continuous integration and on the requirements that prevent using post-deployment data after delivering the system to customer (Laukkarinen et al. 2017). In addition, the authors recommended that new tools and methods be specifically developed for using DevOps in regulated software development (Laukkarinen et al. 2017).

Fazal-Baqai et al. (2017) report on challenges of DevOps adoption in a multi-provider development project in financial domain limited by factors related to project scale and heterogeneity including legal requirements (Fazal-Baqai et al. 2017). The factors result in differences in delivery dates and the quality of software systems developed by various teams amongst others. Furthermore, the suggested tool support for DevOps processes, particularly quality assurance practices, are manifold and require proper selection and integration (Fazal-Baqai et al. 2017).

High learning curve of new technologies

DevOps implementation was observed to be challenged by the necessity for software developers to learn and get acquainted with new technologies, methods and tools, parallel to performing their daily tasks of developing new software features (Jones et al. 2016).
3 Methodology

The objective of the study is to provide a deep understanding of DevOps adoption and implementation in software development practice by comprehending the concept definition, practices, benefits and challenges. To achieve the objective, three research methods are used in the study: systematic mapping (Petersen et al. 2008), multivocal literature review (Garousi et al. 2016) and case study research (Runeson & Höst 2009). The research methods were selected because their qualitative analyses are the best fit for answering and providing thick descriptions to the main research question, which is to explore and aggregate the state-of-the-art and state-of-the-practice of DevOps. Furthermore, the motivations for conducting a systematic mapping study and a case study were driven by software practitioners of the DIMECC Need for Speed (N4S) program. Their need to understand the topics of continuous deployment and DevOps in a broader context in terms of the state-of-the-art, as well as share some lessons learned and best practices from their peers motivated the selection of these research methods. The N4S project provided the context for undertaking the empirical data collection of this thesis. The four-year DIMECC N4S program (2014–2017) was executed by leading Finnish software companies with the aim of providing the capability for an instant value delivery system based on deep customer insights and a real-time experimental business model. The program consortium consisted of 29 industrial organisations and 11 research institutes and universities. Figure 5 illustrates the research design and process of the thesis. In the next sections, detailed explanations of the research are presented. The research design that includes descriptions of the selected research methods is presented first. The research design is then followed by a description of the research process, which includes the steps undertaken to gather evidence. Finally, an explanation of how the results were synthesised from publications to answer the research questions of the thesis is presented.

13DIMECC Need for Speed (N4S) has more information at http://www.n4s.fi/en/. DIMECC stands for Digital, Internet, Materials & Engineering Co-Creation. DIMECC is an ecosystem company whose network consists of over 2,000 R&D professionals, 400 organisations, 69 shareholders and 10 facilitators. DIMECC gathers various projects into research programmes amongst other activities. More information at https://www.dimecc.com/company/
3.1 Research design

In this section, the overall design of the research is described. It includes the description of and rationale for the selected research methods of the thesis, which are a systematic mapping study, multivocal literature and multiple case studies. The systematic mapping study is applied first to consolidate the body of knowledge of the state-of-the-art and to identify research gaps that need to be explored further using multiple case studies. The multivocal literature review is included and used to provide more details about the state-of-the-practice that could not be captured from the systematic mapping study.

3.1.1 Systematic mapping study

A systematic mapping study is a form of systematic literature review (Petersen et al. 2008). A systematic literature review is a well-established research method for collecting evidence and structuring a body of knowledge about a given research area. It is used for identifying and aggregating previous research on a specific research topic (Kitchenham, Budgen & Pearl Brereton 2011). The relevant research papers are referred to as primary studies. A systematic mapping study uses similar methodological steps as a systematic literature review; however, the differences lie on the synthesis of research outcomes. On one hand, a systematic literature review aggregates the research outcomes of primary studies, and in addition, investigates whether the outcomes are consistent or contradictory.

---

Fig. 5. Research design and process of the thesis.
(Kitchenham et al. 2011). On the other hand, a systematic mapping study classifies the primary studies on the basis of publication information, research methods and other topic-specific themes that are relevant to the chosen research topic (Kitchenham et al. 2011). From the classification, an overview of the topic is provided and sub-topics requiring additional primary studies are identified. In general, a systematic mapping study covers a broader topic, while a systematic literature review has a specific and detailed research topic area (Kitchenham et al. 2011).

Given the described differences between a systematic mapping and systematic literature review, the selected research approach was a systematic mapping study. A systematic mapping study was selected because the main aim of the study is to characterise existing research on continuous deployment in order to identify its defining factors. At the time of the study, the meaning and main defining factors of continuous deployment, wherein the DevOps concept gained much popularity (see Section 1.1 for details), were largely still unclarified in the literature and software industry. As such, there was a need for a study that would help to identify and aggregate central key topic areas, limitations and research gaps that characterise continuous deployment.

A research protocol that is normally used to describe all steps taken in conducting a systematic mapping study was developed following the guidelines provided by Kitchenham and Charters (2007). The overall research design of the systematic mapping study, including a definition of research questions and creation of search strings, were described in the study research protocol. The research was done iteratively and continuous feedback from the iteration was gathered and taken into consideration as researchers improved their understanding of the topic. Researchers’ understanding of the topic came about while conducting the research steps. The research process undertaken is described in detail in Section 3.2.1.

3.1.2 Multivocal literature review

The multivocal literature review is a relatively new method for reviewing and synthesising knowledge about a chosen topic from documents published in non-scientific venues, which are often not peer-reviewed, such as blog posts, white papers, web pages and trade journals (Garousi et al. 2016, Ogawa & Malen 1991, Patton 1991). The documents, also referred to as ‘grey’ literature, are normally readily accessible in abundance in a wide variety of forms through the internet. A multivocal literature review is appropriate for investigating topics which are characterised by abundant ‘grey’ literature and
scarce scientific studies. The multivocal literature review has its origins in education research (Ogawa & Malen 1991), but it has recently made its way into the field of software engineering (Garousi & Mäntylä 2016, Garousi et al. 2016, Tom et al. 2013). According to Garousi et al. (2016), a multivocal literature review serves as an additional source of data for state-of-practice of the topic in question. Particularly in software engineering research, Garousi et al. (2016) report that there is a need for more awareness into the multivocal literature review and multivocal literature review studies, mainly because software engineering is a practitioner-oriented field. Thus, more insights would be achieved by combining and synthesising both state-of-the-art and practical knowledge. Including a multivocal literature review would serve the software engineering community with additional sources of knowledge and experience from software practitioners, additional viewpoints, ideas that could scientifically be investigated further, and explanations of their rationality drawn (Garousi et al. 2016). The validity section (see Section 6.2) discusses how validity concerns, such as a lack of rigour in multivocal documents, are addressed in this thesis.

In this thesis, a multivocal literature review was selected as an appropriate method for DevOps because a systematic mapping study and literature reviews (Lwakatare et al. 2015, Smeds et al. 2015) revealed very few scientific studies on the topic of DevOps in scientific venues as of late 2014. Additionally, Smeds et al. (2015) reported that DevOps as a phenomenon is largely discussed by software practitioners in non-academic circles. Furthermore, a decision support checklist on whether to perform a multivocal literature review on a particular topic revealed that DevOps was a good fit. The checklist, presented in Table 6, contains six Yes or No elements for which Garousi et al. (2016) recommend at least one Yes for deciding to include a multivocal literature review.

A multivocal literature review contains diverse views; therefore, it is important that the author sets a clear goal prior to conducting a multivocal literature review. The overall research goal for using a multivocal literature review in the thesis was to understand how software practitioners describe the DevOps phenomenon. To complement the data of multivocal literature review documents, interviews and a workshop with software practitioners were conducted on separate occasions. The research process that was followed in conducting multivocal literature review is elaborated in Section 3.2.2.
Table 6. A decision support tool for deciding on whether to include MLR in a state-of-the-evidence review, taken from Garousi et al. (2016) and applied to the topic of DevOps.

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex intervention</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex outcome</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lack of consensus about measurement of outcome</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Low volume of evidence</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Low quality of evidence</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Context important to implementing intervention</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

3.1.3 Multiple case studies

Case study is a research method in software engineering for observing, explaining and exploring a phenomenon in a real-life setting (Runeson & Höst 2009). In addition, it is largely concerned with increasing knowledge and the understanding of how development, operations and maintenance of software and related artefacts are conducted by software engineers and other stakeholders under different conditions (Runeson & Höst 2009). Yin (2009, p.18) defines a case study as ‘an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident’. Case study research includes both single and multiple case studies (Yin 2009). According to Runeson and Höst (2009), the results of case studies bring a deeper understanding of the phenomenon under study, but not causal relationships as pertains to experiments.

In this thesis, based on the objective of the research, an exploratory case study is selected with an interpretive approach in that insights and knowledge of DevOps are sought from software practitioners. Typically, case study research can be classified on the basis of its purpose, which may be descriptive, explanatory, exploratory or evaluatory/improving (Runeson & Höst 2009). Exploratory is the original purpose of case study research and deals with ‘finding out what is happening, seeking new insights and generating ideas and hypotheses for new research’ (Runeson & Höst 2009, p.135). The other purposes are concerned with ‘portraying a situation or phenomenon’ (descriptive), ‘seeking an explanation of a situation or problem, mostly but not necessary in the form of a causal relationship’ (explanatory), and ‘trying to improve a certain aspect of the studied phenomenon’ (improving) (Runeson & Höst 2009, p.135). From a research perspective, the three types of case studies are positivist, critical and interpretive (Runeson & Höst 2009). Interpretive case study tries to understand a phenomenon
through the interpretation of participants in their context and is similar to exploratory and descriptive purposes of case study (Runeson & Höst 2009). A case, i.e. study object, in software engineering case study research may include private organisations or units of public agencies developing software amongst others (Runeson & Höst 2009). A case in this thesis constitutes a software development team developing one or more software products or services for an internal or external customer of the company. By multiple case, the author of the thesis means that more than one case is included in the analysis. The process used to perform the multiple case study is explained in Section 3.2.3

3.2 Research process

Following the overall research design, the goal of this chapter is to describe the actual process of the research, including all the steps taken in the collection and analysis of data for evidence. Table 7 shows the steps taken in the research processes of the systematic mapping study and the multivocal literature review.
Table 7. Research steps of systematic mapping study and multivocal literature review.

<table>
<thead>
<tr>
<th>Research process step</th>
<th>Aspect</th>
<th>Systematic Mapping Output</th>
<th>Multivocal literature review Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Definition of research question</td>
<td>RQs</td>
<td>Based on the objectives</td>
<td>Based on the objectives</td>
</tr>
<tr>
<td>2. Searching for studies</td>
<td>Search string</td>
<td>'Software' AND ('deliver*' OR deploy* OR releas*) AND (continuous* OR rapid* OR fast* OR real-time OR agile* OR iterative* OR increment*)</td>
<td>'what is DevOps'</td>
</tr>
<tr>
<td></td>
<td>Selected databases</td>
<td>ACM Digital Library, Scopus, IEEE Xplore, ISI Web of Science, Science Direct</td>
<td>Google web search engine</td>
</tr>
<tr>
<td>3. Screening of studies for inclusion and exclusion</td>
<td>Inclusion criteria</td>
<td>scientific article, focuses on software or software-intensive product/system/service development, includes software engineering activity with intention of continuous deployment or delivery</td>
<td>Focuses on DevOps</td>
</tr>
<tr>
<td></td>
<td>Exclusion criteria</td>
<td>a duplicate, not a scientific article, not discussing aspect relevant to software-intensive products/services delivery or deployment</td>
<td>duplicate, video link, pointers to catalogue, course advert, certification advert, presentation slide</td>
</tr>
<tr>
<td></td>
<td>Included studies/documents</td>
<td>50 primary studies</td>
<td>documents</td>
</tr>
<tr>
<td>4. Data extraction</td>
<td>coding themes</td>
<td>main factors of continuous deployment</td>
<td>Definitions of DevOps, descriptions of DevOps as related to agile, lean, continuous deployment, benefits, metrics and challenges of DevOps</td>
</tr>
</tbody>
</table>

### 3.2.1 Systematic mapping study

The research process of the systematic mapping study included in the thesis (Paper I) was aligned with the general guidelines for conducting a systematic mapping study.
(Petersen et al. 2008) and for a systematic literature review (Kitchenham & Charters 2007) in software engineering. The guidelines include five main steps, which are defining the research questions, searching for primary studies, screening studies for inclusion and exclusion, extracting data and evaluating quality. The details of each step are explained in detail in the research protocol and in Paper I. In this thesis, the details are briefly presented in the following paragraphs. The overall research process started in January 2014 and continued until January 2015 when Paper I was first submitted to the Journal of Software and Systems. The research process then continued in August 2015 until October 2015 when the second submission was done after revising the manuscript. The whole process ended in December 2015 when the paper was finally accepted for publication.

**Definition of research questions**

The research questions of the systematic mapping study were defined by the researchers (authors of Paper I) based on the research objectives that took into account the interests of software practitioners of the DIMECC N4S program. Therefore, the research questions were defined from a broad perspective set forth by the following main research objectives:

- To create a body of knowledge of continuous deployment wherein scientific evidence is critically identified based on the research method, research contributions, publication information as well as the relevance and rigour of the selected studies; and
- To this body of knowledge of continuous deployment, to clarify the main factors that characterise and enable continuous deployment, the benefits and challenges of continuous deployment, and lastly, areas that should be addressed in future research based on the identified research gaps.

Based on the aforementioned research objectives, the following were the main research questions of the systematic mapping study:

- RQ1 of Paper I: What is the current state of the research pertaining to continuous deployment in the context of software-intensive products and services?
- RQ2 of Paper I: What are the main factors that characterise continuous deployment in the context of software-intensive products and services?
- RQ3 of Paper I: What are the reported benefits and challenges associated with continuous deployment in the context of software-intensive products and services?
RQ4 of Paper I: What are the research gaps in the area of continuous deployment in the context of software-intensive products and services?

Searching for primary studies

The searching for primary studies step of the systematic mapping study is concerned with the identification of primary studies by applying a chosen search string on selected scientific databases (Petersen et al. 2008). The search string of the mapping study was initially created after performing several pilots by at least six researchers in the course of three months (April–June 2014). The chosen string was later improved after receiving comments from the reviewers following the first submission of the Paper I manuscript to the journal. Specifically, the reviewers suggested including the term ‘release’ as a synonym for deploy. Among the pilot searches, the search string ‘software’ AND (‘continuous delivery’ OR ‘continuous deployment’) was applied to the Scopus database. This pilot search resulted in a small number of relevant primary studies (11 of 28) after excluding duplicates, non-software engineering and non-scientific papers. Upon going through the 11 papers, a broader search string was developed in order to increase coverage and also to form the search string into three parts of population and intervention, as recommended by Kitchenham and Charters (2007) and Petersen et al. (2008). The term ‘software’ made up the population to which the intervention was ‘continuous deployment’, which was put into separate words to allow for the use of synonym terms. Table 7 shows the search string, and Paper I contains the explanations of the rationale for using each term.

Five databases, wherein the search string was used, were selected to be used to search for the primary studies on the basis of their extensive coverage of software engineering literature, and the fact that they are often the most cited sources in many systematic literature reviews. The search string was adapted for each database and was used to search for the terms within the title, keywords and abstract sections of the scientific papers. The search for papers within the five databases was distributed amongst three researchers. The author of the thesis was mainly responsible for performing the search and retrieving studies from the ACM Digital Library database. At the end of the search process, all references retrieved from each database were imported to RefWorks14 for aggregation. From RefWorks, the references were exported to a Microsoft Excel

14RefWorks is a web-based reference management system (http://www.proquest.com/products-services/research-tools/refworks.html)
spreadsheet for further processing, including performing the next steps of screening studies for inclusion and exclusion. The total number of study references retrieved from the databases was 21,382. Detailed information about the actual number of study references retrieved from each database can be found in Paper I.

Screening studies for inclusion and exclusion

The screening studies step of the systematic mapping study applies inclusion and exclusion criteria to determine whether a study is to be included or excluded from the data extraction and quality evaluation steps. The exclusion criteria of the systematic mapping study of the thesis (see Table 7) was kept broad enough such that a study could easily be excluded simply by reading the title and publication forum. The exclusion criteria included checking for whether the study was a duplicate, non-scientific article, or not in the software domain. The exclusion criteria was applied first. In a Microsoft Excel spreadsheet, the title, author, publication forum and publication year fields were orderly sorted and read by two researchers, including the author of this thesis, to determine whether a study was to be excluded based on the exclusion criteria mentioned earlier. At the end of study exclusion process, a total of 9,924 study references proceeded to the process of screening the references for inclusion. The inclusion criteria was influenced by three aspects relevant to the scope and objective of the systematic mapping study. The three aspects that influenced the inclusion criteria were deployment, speed and continuity. The three aspects relate to the ability to bring the software product/system/service to production (deployment) at a much faster speed (speed) and more repeatedly in a continuous manner (continuity). The inclusion criteria were applied to the 9,924 study references that remained after applying the exclusion criteria. Again, two researchers, including the author of this thesis, separately applied the inclusion criteria first to the titles, and when inclusion could not be determined based on the titles it was applied to the abstract and later to the introduction and conclusion sections of the studies. Any conflicting and unsure results from the two researchers were resolved through discussions as well as by involving a third researcher. The exact details of the screening process are elaborated in Paper I. At the end of the screening process step, a total of 50 primary studies were included for data extraction and quality evaluation.
Data extraction

From the 50 primary studies included after the screening process step, different classification categories such as study properties, publication information and recurring themes were extracted from the contents of the primary studies. The study properties and publication information extracted from each primary study included the following attributes: type of paper; research method; contributions; domain; pertinence; publication year; and publication venue. Existing systematic mapping studies such as Paternoster et al. (2014) were used to determine the possible items within an attribute. For example, attributes in the study contribution category used by Paternoster et al. (2014) were adapted and included facets like tool, model, theory, framework/methods, guidelines, lessons learned and advice/implications. For the study properties and publication information, descriptive statistics were used to report the frequencies of the attributes that were relevant to answer the first main research question of the systematic mapping study. For the recurring themes, referred to as factors, thematic synthesis was used to extract them (Cruzes & Dyba 2011). In software engineering research, thematic synthesis is a method used in systematic reviews to help identify recurring themes (or patterns) within primary studies that are then analysed to draw some conclusions (Cruzes & Dyba 2011). Thematic synthesis is reported to be similar to grounded theory with respect to the process of identifying themes but differs from it with respect to the unit of analysis, which in thematic synthesis is a primary study rather than the original data used in primary studies (Cruzes & Dyba 2011). The thematic synthesis was performed by following the steps recommended by Cruzes and Dyba (2011) and both inductive and deductive coding techniques were used when coding data into recurring themes. The 50 primary studies that were selected for inclusion were distributed amongst six researchers for the coding task. Inductive coding was applied first to each primary study whereby codes emerged as a result of six researchers separately assigning labels to a segment of text. Following inductive coding, all the involved researchers then reviewed the labels and coding procedure during a two-day workshop in order to assign high-level themes to the codes. Following the identification of high-level themes, each paper was then deductively coded to a high-level themes, which served as a framework for the second round coding task. As a result, a total of 10 high-level recurring themes, referred to as factors, were presented in the systematic mapping study of the thesis (Paper I). For the data extraction, a data extraction form, available in a Microsoft Excel spreadsheet, were
Quality evaluation

The quality assessment information for the systematic mapping study of the thesis was mainly performed to determine the scientific rigour and industrial relevance of each primary study by applying the scale proposed by Ivarsson and Gorschek (2011). Scientific rigour and industrial relevance information of the primary studies were used to report descriptive statistics of frequency in the number of studies performed and reported by following a scientific research method (rigor) as well as the environment where the study was performed (industrial relevance). To perform the quality evaluation, six researchers in groups of two were assigned a specific number of primary studies to jointly discuss and assign values of rigor and industrial relevance based on the scale by Ivarsson and Gorschek (2011).

3.2.2 Multivocal literature review

Concerning the research process, Garousi et al. (2016) specify performing a multivocal literature review in addition to a systematic literature review. The multivocal literature review studies of the thesis (Paper II and Paper III) did not explicitly include a systematic literature review because such a guideline did not exist at the time the research was performed. However, the author of the thesis considered this part to be covered by the systematic mapping study included in this thesis as well as literature reviews done on DevOps by other authors (Erich et al. 2014, Smeds et al. 2015) as well as the author of the thesis (Lwakatare et al. 2015). This earlier literature review on DevOps by Lwakatare et al. (2015) was used in thesis publication Paper III while being extended in thesis publication Paper II in order to ensure that newer studies of DevOps were included. The literature review followed the guidelines by Webster and Watson (2002).

As there were no explicit guidelines for collecting grey literature at the time of conducting the multivocal literature review, the data collection process of grey literature was informed by procedures used in similar studies, specifically Kerzazi and Adams (2016), Ogawa and Malen(1991) and Tom et al. (2013). However, to triangulate data from grey literature, three interviews (for Paper III) and one workshop (for Paper II)

---

15NVivo is qualitative data analysis software.
were conducted with software practitioners. The interviews with the three software practitioners (one project manager and two software developers) from one company were semi-structured and conducted face-to-face using an interview guide. The workshop was conducted with a total of 11 software practitioners from different organisations and were facilitated by researchers from three universities. The workshops were done in a way that the researchers asked three questions, which were discussed amongst software practitioners in small groups. The questions concerned why they have implemented DevOps, how they have implemented DevOps and lastly what important impacts were experienced in DevOps (and the measures they used). Following their discussions in small groups, the software practitioners were asked to present their answers to the questions. A summary of the workshop participants, questions and answers can be found in Appendix 2 of the thesis. Interviews and the workshop session were recorded and later transcribed for data analysis. The data collection procedure of grey literature took into account the following steps: (1) data sources and search strategy; (2) inclusion and exclusion; and (3) data extraction and quality assessment. All data, including that of the grey literature, interviews and workshop, were stored in NVivo for data extraction and thematic coding (Braun & Clarke 2006). Themes were allowed to emerge from data by coding and analysing the text from the data.

Data sources and search strategy

The Google search engine was used to perform the search and retrieve grey literature in the multivocal literature review. The query used in the search engine was ‘what is DevOps’. The search query was selected on the basis of the overall goal of understanding and exploring the DevOps concept. From the retrieved search results, records were retrieved page-by-page while saving the contents of the links in a PDF up until links to job adverts started, at which point the retrieval process stopped. The total number of records retrieved was 230, which were imported to NVivo for processing in the next step of inclusion and exclusion. For the literature review part of the study, scientific documents were retrieved by searching the term 'DevOps' on the following databases: ACM Digital Library, ISI Web of Science, Science Direct, IEEE Xplore and Scopus. The literature review used for Paper III considered scientific documents up until 11.11.2014 as its results were published in Lwakatare et al. (2015). However, thesis publication Paper II included papers published after 11.11.2014.
Inclusion and exclusion

The inclusion and exclusion of the records were done in parallel to the thematic coding process of the grey literature by following the guidelines by Braun and Clarke (2006). In addition, the initial coding process involved the classification of grey literature with different attributes, such as name, role and place of work, and source information, e.g. publication year, forum and link. Generally, a grey literature document was excluded on the basis of being a duplicate, video link, a pointer to catalogues, course advert, certification advert or presentation slide. These were excluded mainly because analysing data of such documents would require additional effort such as transcribing video content or finding elaborative information from a presentation slide. For inclusion, a record was included if it focused on explaining any aspect of DevOps. As a result, a total of 201 records were included in Paper III because the paper had a broader goal of describing DevOps phenomenon. For Paper II, where analysis was more focused, a total of 75 records (DevOps related to agile, lean and continuous deployment) and 66 records (DevOps benefits) were included from the 201 records. For the literature review part, scientific documents were included on the basis that they were (a) relevant to the topic, (b) peer-reviewed, and (c) published in a scientific journal or in conference proceedings. Primary studies were excluded if they were duplicates or not scientific documents.

Data extraction and quality assessment

The initial coding scheme provided various classifications for the extraction of data. The initial coding was done inductively because categories emerged as the author of the thesis read through the data. Some of the initial categories that were coded included definition, practices, motivations, DevOps in relation to agile, lean and continuous deployment, benefits, and metrics amongst others. Following the initial coding, a second round of coding began and focused on coding sub-categories within the main categories. The second round of coding was done for the main categories of definition, practices, DevOps in relation to agile, lean and continuous deployment as well as the main categories of benefits and metrics. A similar inductive coding technique was applied to scientific documents, and interview and workshop transcripts that were included in the multivocal literature review studies.

A quality assessment of the grey literature was mainly done during the classification of records with additional information related to the author’s role and affiliation. This
process provided a minimum assessment of the content in terms of position, certainty and clarity. However, they were complemented with data from the interviews and the workshop with the software practitioners. Particularly for the workshop session, at the end of the workshop, the participants were presented with the initial results of the multivocal literature review and provided feedback.

3.2.3 Multiple case studies

The guidelines provided by Runeson and Höst (2009) and Yin (2009) for conducting a case study were followed in the thesis. In summary, these include case study design, data collection, data analysis and reporting.

Case study design

The case study design, including the selection of cases and development of the interview guide, was done iteratively in collaboration with researchers from other institutions in Sweden and Finland. The selected cases of the study can be classified as belonging to one of two groups: (1) embedded systems domain; and (2) web and cloud domain. The decision to select cases in the embedded systems domain came about as a result of an intention to replicate the original study by Olsson et al. (2012) that was done in Sweden. Four cases were conveniently selected from the pool of companies participating in the DIMECC N4S programme. Because the latter study revealed that the cases had not adopted DevOps, this led to the decision of purposely selecting software organisations that have successfully adopted DevOps. This intention was communicated to software organisations in the N4S programme. A total of five company representatives nominated themselves and allowed researchers to conduct the study in their respective companies. Their motivation to participate in the study was to share best practices amongst themselves as well as the N4S consortium. The selection of a specific team within the company was left to the company representatives. Typically in case study, a case is a unit of analysis. The unit of analysis in the context of the studies in the thesis is a software development team developing one or more software products or services for an internal or external customer of the company, as summarised in Table 8. The cases are referred as cases A–I because of the confidentiality that was agreed upon by the researchers and software practitioners.
<table>
<thead>
<tr>
<th>Domain</th>
<th>Case</th>
<th>Software product (Development model)</th>
<th>Interviewees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded</td>
<td>A</td>
<td>Special device platform product (In-house) and R&amp;D services (Customer-project)</td>
<td>Special device senior manager, special device product owner, sales and account manager, sales and account manager, senior specialist in software, quality manager in wireless segment</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Compact mobile broadband solution (In-house)</td>
<td>Test automation manager, senior developer, program manager, operations manager, technical coordinator</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Factory automation platform solution (In-house)</td>
<td>Project manager, program manager, user experience designer, product manager, developer</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Networking monitoring solution (In-house)</td>
<td>System verification engineer, program manager, software architect, product line manager, software engineer</td>
</tr>
<tr>
<td>Web</td>
<td>E</td>
<td>Web-based road maintenance reporting tool (Customer-project)</td>
<td>Developers (3), project manager, site manager</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Websites for safety at work, occupational health and quality portal (Customer-project)</td>
<td>Developers (3), product owners (2), director of culture and competencies</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>Web-based application to access and manage acquired services by customer (In-house)</td>
<td>Developers (2), product owner, head of development, IT department-Ops team lead</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>REST API for media content (In-house)</td>
<td>Developers (2), product owner, product owner of operations team, system administrator</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>Security cloud services (In-house)</td>
<td>Developers (4), team lead</td>
</tr>
</tbody>
</table>

Interviews were the main data collection technique selected for all cases. For the cases in the embedded systems domain, the first the interview guide was provided by the authors of the original study (Olsson et al. 2012), which in collaboration with other researchers was modified and some questions relating to DevOps were added to the original guide. The interview guide had open-ended questions divided into three parts:
current ways-of-working in software development, deployment and post-deployment; (2) strengths and weaknesses in the ways of working; and (3) barriers experienced when moving towards continuous deployment. The interview guide 1 for the other set of cases in the web and cloud domain was developed collaboratively by researchers from three different universities in Finland: University of Oulu, Tampere University of Technology and Aalto University. The interview guide had open-ended questions divided into six themes: 1) the background of the case and interviewee; 2) software development practices; 3) software build, integration and release practices; 4) software monitoring and infrastructure management practices; 5) perceived impacts; and 6) software development culture in organisation.

Data collection

The data from all cases was primarily collected through semi-structured interviews with software practitioners in various roles (Table 8). The interviews were performed by at least two researchers where one was primarily responsible for asking the questions and the other(s) taking notes. From each case, at least five software practitioners were interviewed, taking on average 1–2 hours. All interviews were recorded and later transcribed for analysis. Prior to the actual data collection process, the interview guide was trialled in pilot cases for feedback purposes. The actual data collection process from the companies was done first over the course of three months, November 2014 – January 2015, for companies in the embedded system domain, and later during the months of March 2016 – May 2016 for companies in the web and cloud domain. In addition to interview data, the case data included observations that were made in some cases, especially those in the web and cloud domain.

Data analysis

All interview transcripts were stored in NVivo for data analysis. Data were analysed mostly by following a thematic coding technique (Braun & Clarke 2006). The coding was first done for individual cases and later a cross-case synthesis was performed, as is often done in multiple case studies (Runeson & Höst 2009). Themes used for coding the data were iteratively created and piloted amongst researchers prior to the actual coding. As several researchers were involved in coding the data, in some cases, a coding agreement level was determined. Thus, when a low level was determined, several
approaches were used to improve the number, including conducting coding discussion workshops and creating an explicit coding guideline document that was called ‘coding cookbook’. For cross-case synthesis, a recommendation by Yin (2009, p.156), ‘creation of word tables that display individual cases according to some uniform framework’, was used. Figure 6 shows an extract of Microsoft table displaying individual cases and the uniform framework. The uniform framework was taken from a theoretical framework of the activities in the deployment pipeline, as well as other themes of interest, such as benefits and challenges.
3.3 Synthesis of the findings

The research design and process of each method resulted in publication(s). From the systematic mapping study, the results were reported in Paper I. From the multivocal literature review, the results were reported in Papers II and III. From the multiple case studies, the results were reported in Papers IV and V. All Papers (I–V) contribute to
answering the main RQs of the thesis. Therefore, the results from each of the papers are synthesised to give the overall research results of the thesis on the basis of the links between the RQ of the paper and the RQs of the thesis. The links to the RQs are presented in Table 9.

Table 9. Links between the RQs of the publication papers and the thesis.

<table>
<thead>
<tr>
<th>Publication</th>
<th>RQ in Publication</th>
<th>Thesis RQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper I</td>
<td>What are the main factors that characterise continuous deployment in the context of software-intensive products and services?</td>
<td>RQ2</td>
</tr>
<tr>
<td>Paper II</td>
<td>How does DevOps relate to agile lean and continuous deployment?</td>
<td>RQ1, RQ2</td>
</tr>
<tr>
<td></td>
<td>What are the claimed effects of DevOps, and what metrics can be used to assess those effects?</td>
<td>RQ3</td>
</tr>
<tr>
<td>Paper III</td>
<td>How do software practitioners describe DevOps as a phenomenon?</td>
<td>RQ1</td>
</tr>
<tr>
<td></td>
<td>What are the DevOps practices according to software practitioners?</td>
<td>RQ2</td>
</tr>
<tr>
<td>Paper IV</td>
<td>What are the key challenges for DevOps adoption in the embedded systems domain?</td>
<td>RQ3</td>
</tr>
<tr>
<td>Paper V</td>
<td>What is DevOps according to industry practitioners?</td>
<td>RQ1</td>
</tr>
<tr>
<td></td>
<td>What key practices are employed in the DevOps approach?</td>
<td>RQ2</td>
</tr>
<tr>
<td></td>
<td>What are the perceived benefits and challenges of the DevOps approach?</td>
<td>RQ3</td>
</tr>
</tbody>
</table>
4 Findings of the original publications

The findings of the thesis are presented in five peer-reviewed publications in international conferences and journals in the field of software engineering. Four of the five publications have been published and are currently available online through the publishers’ digital libraries (or scientific databases). The fifth publication was submitted for review in the *Information and Software Technology (IST)* journal on 31.3.2017, and the reviews have yet to be received. This chapter presents the findings of the thesis by first providing a summary of the main findings of each publication in separate subsections. Thereafter, at the end of the chapter, a summary of the findings is presented in accordance with the three research questions of the thesis. Table 10 summarises the main research findings of the publications that are related to this thesis.

<table>
<thead>
<tr>
<th>Publications</th>
<th>Purpose of the study</th>
<th>Highlights of the findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Rodríguez, P., Haghighatkhah, A., Lwakatare, L E, Teppola, S., Suomalainen, T., Eskell, J., Karvonen, T., Kuvaja, P., Verner, J., &amp; Oivo, M. (2017) Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, 123, pp 263–291</td>
<td>To identify the state-of-the-art of the continuous deployment phenomenon</td>
<td>DevOps is an organisational and technical enabler of continuous deployment. From an organisational perspective, it focuses on the integration of corporate functions of R&amp;D and operations. From a technical perspective, it focuses on automation of the delivery pipeline with a specific focus on configuration management and post-deployment activities.</td>
</tr>
<tr>
<td>II. Lwakatare, L E., Kuvaja, P., &amp; Oivo, M. (2016). Relationship of DevOps to agile, lean and continuous deployment: A multivocal literature review study. In 17th International Conference on Product-Focused Software Process Improvement, pp. 399–415. Springer International Publishing.</td>
<td>To clarify the relationship of DevOps to associated software development approaches</td>
<td>DevOps originates from continuous deployment as an evolution of agile software development and is informed by lean principles. Three propositions are given and include: (1) agile software development principles, values and practices are required for successful adoption of DevOps; (2) DevOps implementation is necessary for continuous deployment; and (3) lean software development principles and practices inform DevOps implementation</td>
</tr>
<tr>
<td>Publication</td>
<td>Purpose of the study</td>
<td>Highlights of the findings</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>III. Lwakatare, L E., Kuvaja, P., &amp; Oivo, M. (2016). An exploratory</td>
<td>To consolidate the understanding of the DevOps phenomenon as described by practitioners</td>
<td>DevOps is best understood as a mindset change substantiated with practices to ensure effective collaboration between software development and operations for the purposes of releasing quality software features more rapidly to end users. DevOps practices are diverse; however, patterns to the practices can be observed from its key dimensions of collaboration, culture, automation, monitoring and measurement.</td>
</tr>
<tr>
<td>study of DevOps extending the dimensions of DevOps with practices. In 11th International Conference on Software Engineering Advances, pp. 91–99. IARIA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IV. Lwakatare, L E., Karvonen, T., Sauvola, T., Kuvaja, P., Bosch, J., Olsson, H. H., &amp; Oivo, M (2016). Towards DevOps in the embedded systems domain: Why is it so hard? In 49th Hawaii International Conference on Systems Science pp. 5437–5446. IEEE.</td>
<td>To identify key obstacles for the adoption of DevOps in the embedded systems domain</td>
<td>DevOps in the context of the embedded systems domain is challenging due to its dependency on hardware, limited visibility to customer environments, lack of deployment technology for customer-specific environments, and inability to get usage data from the customer environment to support the phenomenon</td>
</tr>
<tr>
<td>V. Lwakatare, L E., Kilamo, T., Heikkila, V., Karvonen, T., Sauvola, T., Ikkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., &amp; Lassenius, C, (Submitted for Review). DevOps in practice: A multiple case study of five companies. Journal of Information and Software Technology</td>
<td>To expound DevOps practices found in the deployment pipeline, in addition to exploring the culture and mindset necessary for DevOps adoption</td>
<td>DevOps implementation from deployment pipeline perspective include practices for automating software deployment, employing the infrastructure-as-code practice as well as monitoring of software in various environments as an activity that is performed by software developers. These practices are incorporated into other software development practices of trunk-based development, code reviews, continuous integration, and agile and lean practices.</td>
</tr>
</tbody>
</table>
4.1 Paper I: Continuous deployment of software-intensive products and services: a systematic mapping study

The aim of the study of Paper I was to identify the state-of-the-art of continuous deployment, and to determine the underlying factors that characterise the continuous deployment phenomenon. In relation to the thesis, the contribution of Paper I is the body of knowledge of continuous deployment wherein DevOps is recognised as an aspect in its characterising factors. From the characterising factors where DevOps is presented, descriptions of DevOps practices are identified and presented below.

The findings showed that the majority of the primary studies included in the systematic mapping study on continuous deployment constitute case studies (24%) and industrial reports (36%), of which most contributions were perceptions of software practitioners presented in the form of lessons learned (27%), advice (16%) and guidelines (5%). Therefore, the state-of-the-art and the state-of-the-practice on continuous deployment are largely driven by contributions of software practitioners. An in-depth analysis of the primary studies revealed 10 recurring themes/factors that characterise continuous deployment. The 10 factors were: (1) fast and frequent releases; (2) flexible product design and architecture; (3) continuous testing and quality assurance; (4) automation; (5) configuration management; (6) customer involvement; (7) continuous and rapid experimentation; (8) post-deployment activities; (9) agile and lean; and (10) organisational factors that included integrated corporate functions, transparency, and an innovative and experimental organisational culture. The first factor is a defining factor as it describes the continuous deployment phenomenon, and the remaining factors constitute enabling mechanisms of continuous deployment. DevOps emerged as a theme explicitly discussed in factors 4, 5, 8 and 10, contributing to both social and technical mechanisms. The following paragraph elaborates more on the latter.

Descriptions of DevOps practices were found in the organisational factor of integrated corporate functions (factor 10), which focused on the integration of corporate functions, and particularly the software development unit and the operations unit. In the latter, DevOps was identified as important in facilitating transparency and minimising inter-organisational problems between the two units. In particular, problems such as poor communication and hand-overs of software changes were causing delays in the delivery process of software changes. Some DevOps practices that were mentioned explicitly in the primary studies for the integration of the software development unit and operations unit included: (1) the use of cross-domain competencies to constitute a software
development team, since this would enable the exchange of skill sets; and (2) the use of common practices and platforms across the R&D unit and operations unit.

Descriptions of DevOps practices were also linked to automation (factor 4), configuration management (factor 5) and post-deployment activities (factor 8). These were reported as important for speeding up the release cycle of software changes (Section 5.4.2 of Paper I) and provisioning of system environments (Section 5.4.3 of Paper I). DevOps practices and their corresponding tools were mentioned in automation of the deployment process of software changes and monitoring in post-deployment activities. Some excerpts of the practices in the configuration management activity presented in the paper include: "mimicking the production environment in test environments and using tools to manage configurations in a similar way as source code, and monitoring deployments" (Section 5.4.2 of Paper I). Tools, such as Chef and Puppet, are reported to be used in the configuration management of environments in the deployment pipeline. The latter configuration management tools use a declarative domain-specific programming language to specify the desired system configuration parameters and configuration actions that are applied during the deployment process of software changes (Section 5.4.3 of Paper I). Furthermore, to ease the management of system configurations and prevent configuration drifts, two practices are suggested (Section 5.5.2 of Paper I). The two practices are: (1) maintaining runtime configurations in files separate from source code; and (2) deploying the same software binaries to all system environments. The post-deployment factors focus on monitoring the system in production for the purpose of ensuring that system quality attributes are met, identifying and resolving unexpected patterns and runtime issues, as well as monitoring usage behaviour of the end users (Section 5.8 of Paper I). Based on the findings, Paper I provides answers to RQ2 of the thesis.

4.2 Paper II: The relationship of DevOps to agile, lean and continuous deployment: a multivocal literature review study

The aim of the study of Paper II was to explore the relationship of DevOps with conflated software development approaches, specifically agile, lean and continuous deployment. In addition, the purpose was to gather the claimed benefits and metrics for assessing the benefits of DevOps. The contributions of the paper in relation to the thesis are threefold. The first contribution is a summarised description of the DevOps concept that takes into consideration its relationship with the three software development approaches of agile,
lean and continuous deployment. The second contribution comprehends propositions of DevOps’ relationship to agile, lean and continuous deployment, which are useful when adopting and implementing DevOps. The third contribution consists of a list of benefits and metrics for assessing the benefits of DevOps.

The analysis of the multivocal documents (grey literature) and scientific studies showed that 37% of multivocal documents and 30% of scientific studies describing software practitioners’ understanding of DevOps were conflating DevOps with agile, lean and continuous deployment. When DevOps is conflated with agile, lean and continuous deployment, it is often done from four perspectives: origin, adoption, implementation, as well as goals and values. The finding that DevOps is related to agile, lean and continuous deployment was summarised in the following description of DevOps: *DevOps phenomenon originated from continuous deployment as an evolution of agile software development, informed by lean principles.* The next paragraphs elaborate further on the latter, and in addition, present some propositions that were reached as a result of the data analysis of Paper II. The last paragraph presents the benefits of DevOps and the metrics for assessing the benefits.

DevOps is observed as either an evolution of agile software development, particularly the principles and values of the Agile Manifesto, or alternatively, as a part of agile software development. The adoption and subsequent establishment of agile practices in software organisations have played a role in highlighting most of the problems that DevOps is trying to solve. Collaboration and frequent interactions between the software development unit and the customer enabled incremental development of software features, but production releases occurred less frequently compared to the speed of development within an iteration (sprint or increment). To facilitate production releases at a similar speed of software development within an iteration, DevOps is claimed to build upon the agile practice of continuous integration from the implementation perspective. It was noted by one practitioner in the workshop (Appendix 2) that breaking communication silos with an operator customer, as a large organisation in the telecommunications industry, enabled the delivery of software build packages from a continuous integration system to a live environment in a day rather than the usual six months (see quotation in Paper II). In addition, small increments are not just done for software functionality but also for the software infrastructure, which need not be fully built at the beginning of the project. As such, collaboration that was once emphasised between the software development unit and the customer is extended to system operators so as to efficiently release software changes and effectively operate the
system in production. Agile and DevOps share similar values and goals of breaking down organisational silos with a focus on the rapid development of software features. As such, it was postulated that agile software development principles, values and practices are required for successful adoption of DevOps. Notice that in the proposition, attention was not given to a specific agile method or framework, but rather to the principles and values of agile that are generally applicable in agile methods and frameworks.

The popularity of DevOps is a result of the emergence of continuous deployment. If one is to consider continuous deployment as an end goal, DevOps represents one of the collective means to achieve the end goal. A software practitioner participating in the workshop (Appendix 2) pointed out that they implemented DevOps due to the software organisation’s need to go beyond the established practice of continuous integration (see quotation in Paper II). Furthermore, workshop practitioners from the embedded systems domain in telecommunications reported that the adoption of DevOps principles for short release cycle times was challenging, due to factors such as complexities in network integration and verification processes within the organisation without considering the challenges in the inclusion of the operator customer (see quotation in Paper II). DevOps constitutes one of the many ways for achieving continuous deployment because continuous deployment as a paradigm is much broader and requires the consideration of many other aspects, such as a new business model. As such, DevOps brings focus to specific practices in the delivery process of software changes. Those that were mentioned in the paper involve: (1) organisational and cultural practices designed to eliminate the gap between software development and operations; (2) automation of the deployment process in the deployment pipeline; (3) automation in infrastructure management for reliability and reproducibility; and (4) monitoring. The proposition given in the paper is that DevOps implementation is necessary for continuous deployment, and this summarised the relationship of DevOps with continuous deployment.

The paper proposed that lean software development principles and practices inform about DevOps implementation. This means that lean makes the case for continuous deployment and, in turn, gives reasons why much of what is done in DevOps is important. Lean thinking and principles emphasise that software development takes into account the entire software development value stream such that the whole development process is optimised for the delivery of valuable system changes. As an example, Kaizen, a lean principle for continuous improvement, focuses on increased feedback and a culture of collaboration. Through DevOps, collaboration extends software development and business to complete the entire delivery chain. Furthermore, waste is minimised, if not
eliminated, through short feedback loops that include knowledge from operations for informed insights into software quality, as well as optimised performance of the entire system delivery chain.

Several benefits were claimed to be achieved as a result of DevOps adoption and implementation. Among the most common benefits of DevOps was the ability to release software quickly, frequently and with improved quality. Popular metrics for assessing the claimed benefits were reported to include deployment rate, cycle/lead time and mean time to recovery. However, a point of notice was given pertaining to the claimed benefits. The fact was that the claimed benefits were too generic, and it was difficult to argue that they can be achieved by simply integrating software development and operations. Even in empirical scientific studies, other factors such as the organisational context, type of customer and system as well as the maturity of the system significantly influenced the claimed benefits.

4.3 Paper III: An exploratory study of DevOps: extending the dimensions of DevOps with practices

The aim of the study of Paper III was to consolidate software practitioners’ understanding of the DevOps phenomenon. The contributions of the paper in relation to the thesis are twofold. First, an enhanced scientific definition of DevOps is presented that is based on the original definition proposed by Penners and Dyck (2015). The second contribution is to present key elements of the DevOps concept that also include a set of DevOps practices or patterns of the practices.

The findings of the research question of Paper III, how software practitioners describe the DevOps phenomenon, showed that among the most common terms used to refer to DevOps, the leading terms were cultural and professional movement. These signify a desired change in culture within the software industry in the ways of working, especially where software development and operations processes intersect. When DevOps is described further with the common terms, the descriptions were observed as dependent on the given emphasis, which was mostly either the goal or the means for achieving collaboration between software development and operations. The common goals identified include reducing response time and fast deployment of high-quality and reliable software features. The common means identified were advanced automation, and evolving roles between software development and operations. The latter aspect —both organisational and technical mechanisms —also emerged from the responses of software
practitioners during interviews. Based on these findings, the scientific definition proposed by Penners and Dyck (2015) was enhanced to: ‘DevOps is a mindset substantiated with a set of practices to encourage cross-functional collaboration between teams —especially development and IT operations —within a software development organisation, in order to operate resilient systems and accelerate the delivery of changes’. The argument was that since the majority of the software practitioners referred to DevOps as a cultural and professional movement, this emphasised the importance of including ‘mindset’ in the original definition. This was similar to the goal of the definition, ‘in order to operate resilient systems and accelerate delivery of change’. However, it was deemed important to not leave the definition to only mindset, as several practices (elaborated in the next paragraph) were attributed to DevOps. Therefore, mindset had to be substantiated with a set of practices; hence, its inclusion in the definition was crucial.

On the research question of the paper, what are the DevOps practices according to software practitioners, the finding was that DevOps practices are diverse, and it is more useful to describe patterns of the practices, particularly as an individual practice often embodies contextual considerations, such as the organisational structure of software development and operations teams. The highest level concepts that emerged from coding DevOps practices into patterns during data analysis constituted its dimensions. Five dimensions, were identified: collaboration, automation, culture, monitoring and measurement. Each dimension has one or more patterns to a set of DevOps practices. First, in the collaboration dimension, the observed pattern of DevOps practices was the reorientation of roles and responsibilities between software development and operations units. In this pattern, the two common organisational DevOps practices observed were: (1) an increased scope of responsibilities for developers; and (2) intensified cooperation between software development and operations. Importantly, the practice of increasing the scope of the developers’ responsibilities was reported to be the most common and was aimed at empowering developers as they gained control of operations activities, allowing for both software development and operations to be done in a single team. This helped to broaden the skills and knowledge of the developers, but with the requirement that at least some of the software developers in the team be proactive and willing to learn the newly required skills. Furthermore, the organisational DevOps practices were observed to involve a change of culture and mindset that encourages empathy, support and a good working environment for those involved in software development and operations. The rest of the dimensions — automation, monitoring and measurement — are related to the technical practices of DevOps. Patterns in the automation dimension were concerned
with the automation of the deployment process, wherein the practice of infrastructure-as-code was predominant. An example of the practices in the automation dimension included the use of tools, such as Chef and Puppet, to automate and maintain code for infrastructure provisioning and configuration settings. The monitoring dimension had patterns of instrumentation of software applications for the purpose of monitoring and the aggregation of monitored data into useful insights. The measurement dimension specified production metrics for monitoring and assessing the performance of the processes in software development and operations. The culture dimension had patterns of empathy, support and a good working environment between software development and operations. Examples of the practices for the culture dimension included having both software developers and operations personnel responsible for handling production incidents.

4.4 Paper IV: Towards DevOps in the embedded systems domain: why is it so hard?

The aim of the study of Paper IV was to identify the main obstacles for DevOps adoption in the embedded systems domain. The analysis of the obstacles was based on cases A–D from Table 8, whereby the companies behind cases A–D are large software-intensive organisations. Two main contributions of the paper are relevant to the thesis. The first contribution is the introduction of the DevOps concept in the embedded systems domain, which is a different context from where DevOps originated. The second contribution pertains to the identification of challenges for DevOps adoption.

Software development and deployment practices of the studied cases revealed that all companies, with the exception of case A, had not institutionalised the practice of continuous integration (Karvonen et al. 2015), mostly due to significant efforts that are required to automate testing. As such, continuous deployment, and subsequently DevOps, were not institutionalised across all the companies. The release cycle time of a software development project before product launch ranged from six months to a year, sometimes several years for case A due to different constraints, such as new hardware testing and certification requirements. In two cases, B and C, the release cycle of new features followed company-wide half-year release periods. Interim releases of much shorter cycle times, e.g. two weeks to customers, were observed for new features during

---

16In this thesis, embedded systems are computer systems that are developed concurrently with the hardware. As such, they are just one part of a much larger system whose requirements must be taken into consideration.
the development phase at case A, and others had on-demand maintenance or company-specific releases. For the interim releases, deployment to the customer was done either through over-the-air technology or by pushing files to customers’ repositories for Case A. At the end of the development phase and prior to product launch, customer acceptance testing was conducted with the lead customer to validate software functionality in a production-like simulated environment or a real field environment for case B. In two cases, A and C, the release manager made the new releases and corresponding documentation available to the customer (case A) and operations (case C). At case A, the operations team was responsible for serial production of devices together with the customer and external manufacturing companies. At case B, customer units and service personnel at local sites were responsible for product installations at the customer site. At case C, operations was responsible for customer delivery projects wherein software features are installed and configured for specific customers. At case D, technical support units assisted the customer with the installation of the software that was made available online.

The main challenges of DevOps adoption were related to: (1) hardware dependency and multiple version compatibility; (2) limited visibility to the customer environment when configuring test environments; (3) lack of technology to automatically and reliably deploy new features repeatedly in customer-specific environments; and (4) absence of feature usage data in system performance data. The first challenge area, *hardware dependency and multiple version compatibility*, relates to the organisational structure used in the development of an embedded system and the interactions across the teams. In the cases, there were mono-disciplinary ‘module’ teams in addition to cross-disciplinary ‘feature’ teams. Because of the hardware, module teams tended to work on the lower layers of the system stack close to the hardware, which required some specialisations. Cross-disciplinary ‘feature’ teams tended to work at the user-interface level and almost covered the entire system stack. The results showed that there was a greater need for cross-disciplinary ‘feature’ teams, since fewer interactions between module teams made it challenging for the teams to effectively and quickly integrate system changes and on a continuous basis. Both a senior developer from case B and a project manager from case C gave evidence of the desired cross-discipline interactions (see quotation from Section 4.2.1 of Paper IV). Particularly interesting is the quote from the project manager, which details his assumption and unsuccessful act of bringing to the software development team a person from the operations team for the purpose of sharing customer knowledge with the team. The action failed due to what the project manager described
as a personality problem; the person was more like a scientist, signifying the need for cultural and mindset changes in the adoption of DevOps. The second challenge, *limited visibility to the customer environment when configuring test environments*, relates to the DevOps practice of ensuring that environments used during development are representative of the production environment. For example, the telecommunications network and the factory contributed to the unrepresentativeness of the test environments when compared to the actual environments when configuring an environment for a single customer. Moreover, customer environments were reported to also include various other components, of which some were provided by other vendors. The latter limitations resulted in numerous bugs being discovered during acceptance tests that were done at the customer environments of selected lead customers. In addition, the limitations made the task of configuring test environments to mirror customers’ environments very complex.

The third challenge, *lack of technology to automatically and reliably deploy new features repeatedly in customer-specific environments*, relates to the need for more advanced technological support for the automatic deployment of software features. In contrast to the cloud domain where technological support is vast and the infrastructure is owned by the company, it was observed that the lack of tool support and the required customer consent were among the factors inhibiting fast deployment of software features in the embedded systems domain. The factors represent the need for ensuring that software versions remain compatible after every software update, despite the long life cycle and the large amount of legacy code. In addition, every software update must be done with almost zero downtime due to reliability, safety, security and traceability requirements in embedded systems. The last challenge of the paper, *absence of feature usage data in system performance data*, relates to the DevOps practice of using post-deployment monitored data to inform about and validate design decisions pertaining to system feature requirements. Post-deployment data and monitoring, e.g. in the form of logs, is legally required for products developed by the companies of the studied cases for such purposes as traceability. The data, upon request to and consent by the customer, is made available to the case companies for the purpose of fixing and improving testing, e.g. generate load scenarios, but is often not done for the purpose of monitoring software feature usage behaviour. Therefore, the value obtained in monitoring software feature usage behaviour has yet to be determined and effective mechanisms to facilitate the data need to be implemented.
4.5 Paper V: DevOps in practice: a multiple case study of five companies

Paper V presents the state-of-the-practice of DevOps from development teams of five companies that have adopted DevOps. The aim of the study of Paper V was to expound on DevOps practices found in the deployment pipelines of development teams from cases E–I (see Table 8). In addition, the purpose was to identify the culture and mindset necessary for DevOps. The main contributions of the paper that are relevant to the thesis are threefold. The first contribution is providing software practitioners’ with a shared understanding of the DevOps concept. The second contribution is giving detailed descriptions of the practices of DevOps. The third contribution is providing detailed descriptions of the benefits and challenges of implementing DevOps practices.

All the studied cases were developing web-based software products and services. The companies behind cases E–G are small- and medium-sized, and those of cases H–I are large organisations. The companies of cases E and F are consultancy companies providing software product engineering services to customers. The customers of consultancy companies in the studied cases were a public transport agency (case E) and an independent occupational health and safety research and development organisation (case F). The companies behind the other cases were developing their own software products and services for either an external (case G) or internal customer (cases H and I). In the latter companies, software development was also performed by employees from consultancy companies in addition to their own employees, apart from the company of case I. Consultants performing software development work were co-located with the customer except for case E. Three cases (cases E, G, H) had a separate operations team, which interacted with the software development team on an as-needed basis, often using informal channels. Four teams (cases E, F, G, I) used the Amazon Web Services (AWS) cloud platform. Case H was transitioning from an internal private VMware-based cloud to the AWS cloud. DevOps was largely introduced by enthusiast software practitioners within the organisation, including software developers, product owner, and heads of software development and operations. Drivers for the adoption of DevOps included customers’ willingness to use the AWS infrastructure, the development teams’ willingness to improve their ways of working, and the need to reduce the time-to-market. Deployment frequency is daily to staging (case E) and production (cases G, H) environments. For case F, deployments to production environment are made several times during two weeks sprint cycle. Case I employs canary deployment...
that takes about 1–2 weeks from Alpha-to-Beta-to-Production environments, the main differences in the environments being the number and type of users.

The findings of software practitioners’ understanding of the concept of DevOps included cultural and mindset changes that support the software development goal of making software changes visibly fast in production for fast feedback. Several software practitioners reported achieving DevOps as a result of a supportive organisational culture, including the software developers and operations personnel as well as other stakeholders, in particular, the customer (case F) and management (cases E, G, H, I). As such, the majority of the practitioners understood DevOps as having members of the software development team gain more control and responsibility to design, test and operate software products and services, which they were developing, in production. The ability of a software development team to take more responsibility was largely facilitated by increased automation in both software development and operations activities, particularly in software testing, deployment and monitoring. The operations team were observed to take a supporting role. They supported software developers in various activities, including sharing and providing software developers with access rights to environment clusters, tools for software deployment and monitoring, and knowledge of security, scalability and performance issues. This is in addition to jointly resolving problems and incidents that occur in the production environment.

Key practices in DevOps implementation observed in the deployment pipeline include: an automated software deployment mechanism; infrastructure as code practice; and continuous monitoring of software in various environments, including production by the software development team. In addition, other software development practices were reported as important to achieve the software development goal. The practices were identified in at least four cases and included trunk-based development, change-based code reviews, continuous integration, and agile and lean practices. On the practices of automating the software deployment mechanism and infrastructure-as-code, configuration management tools, such as Chef and Puppet, were used to create scripts for environment provisioning and software deployment. A common practice that was reported as useful was giving the software development team access and the ability to modify these scripts, which were also version-controlled. In most cases, a software developer’s changes to the scripts were also reviewed by either another software developer or a system administrator from operations, often depending on the types of changes and the level of confidence in the developer who made the changes. Other than version-control and reviews, it was observed that there was no active practice to test the
script other than monitoring them after their execution. The most common deployment strategy was mostly blue-green deployment, but rolling-upgrade and installing build packages to existing virtual instances were also reported. Developers monitored systems particularly during software deployments to production, and the data and patterns of different monitored metrics were made visible through information radiators found at the software development teams’ workspaces. The key lesson in the implementation of DevOps from the cases was that the infrastructure is no longer considered in isolation to the development of software features.

The perceived benefits of DevOps from the cases were improvements in the areas of: speed (release cycle time) and continuous deployment of system changes; productivity of operations’ work; morale, knowledge and skills, particularly of developers; quality of the system; and the culture and mindset for a wider dissemination of DevOps practices across the company. The detailed descriptions clearly showed that improvements in delivery speed and software quality did not only result from DevOps practices, but with the incorporation of other practices, such as code reviews, trunk-based development and automated testing. The other benefits were mostly due to DevOps practices, which were perceived as resulting from the removal of organisational boundaries, communication gaps and a reduction of tacit knowledge for the productivity of operations work. Furthermore, improved morale and skills stemmed from increased motivation, happier staff, and the autonomy of software developers. For the perceived challenges, those identified included: project and resource constraints; insufficiencies in the management of the infrastructure; high demands for required skills and knowledge, particularly among software developers; difficulties in balancing speed and quality; and difficulties in monitoring microservices. For consultancy companies, the customer determines how software development work is performed (case E), unless the customer is willing to learn and adopt the provider’s ways of working (case F). As was observed in case E, in a large fixed-price project with multiple providers, production releases are pre-defined and agreed upon in the early stages of project definition and negotiations. Therefore, the production release is not only delayed until the scheduled time, but this also causes problems for test automation and leaves little time for piloting production deployments. For a large organisation with a private cloud, resource constraints in terms of a limited number of employees in both software development and operations teams as well as a complex infrastructure built using old technologies were presenting challenges to the implementation of DevOps. An even more important observation of the findings was that it was seen as difficult for software developers, but preferred by operations
personnel, not to fully automate the network component of the infrastructure due to legacy relational databases that could not be clustered, so as to prevent oneself from getting locked (see quotation in Section 6.2 of Paper V). Related to the challenges in the automation of the infrastructure, in case G, erroneous data was displayed to customers causing frustrations for the software development team. According to the product owner in case G, the operations team were responsible for the data platform that was used to store customer data. The head of operations of case G accounted for the problem as due to incorrect updates that were made to the service index of the search engine data, which was loaded every night to the data platform. As such, frustrations regarding the problem faced were expressed by the product owner, who decided to have frequent meetings with the operations team. In all cases, it was reported that the adoption of DevOps demanded that at least some members of the software development team acquire new skills and knowledge of technologies. In cases F and I, software developers were struggling with the learning curve of the DevOps practices and associated technologies.

### 4.6 Summary of the findings

This section summarises the findings of the original publications in the form of the research questions of the thesis. A discussion on the findings is presented in the subsequent chapter. Table 11 presents a summary of the findings according to the research questions of the thesis. The systematic mapping study briefly introduced some of the DevOps practices but did not describe them in detail because of their narrow descriptions in the identified primary studies. As such, detailed descriptions of DevOps practices, the need for the practices and the domains in which they are applicable remained unclear but were identified as suitable areas for future research, which this thesis also corroborates with the other four publications. Multivocal literature review studies and multiple case studies explored and presented detailed descriptions of the DevOps concept and practices that were identified in the systematic mapping study. Particularly in the multiple case studies, the detailed explanations of DevOps practices as well as their benefits and challenges are given from different software development contexts. In the following paragraphs, a summary of the findings of the thesis is presented for each research question based on the findings of the research activities undertaken and resulting in the publications discussed in the previous sections.
Answers to thesis RQ1: What is the understanding of the DevOps concept in software development practice?

Based on narratives of software practitioners, it was found that DevOps is a concept that embodies a cultural and mindset change that is substantiated with a set of practices to ensure cross-disciplinary collaboration between software development and operations within a software-intensive organisation. The main purpose for the collaboration is to enable the fast release of quality software changes while simultaneously operating the system reliably and resiliently in production. These findings for understanding DevOps are found in the most common terms (cultural and professional movement) and common goals (to reduce cycle time and fast deployment of high-quality and reliable software products and services) of software practitioners’ descriptions of the DevOps concept reported in Paper III. The findings for understanding the DevOps concept are found in multiple case studies reported in Paper V, where there was a recognition of the importance of supporting culture and mindset in DevOps. Furthermore, DevOps is a phenomenon that was observed to originate from continuous deployment as an evolution of agile software development and is informed by principles of lean, based on the findings of Paper II. Key characterising elements of DevOps are identified from Paper III and Paper V and include collaboration, culture, automation, monitoring and (or) measurement.

Answers to thesis RQ2: What practices are found in DevOps implementation in software development practice?

Based on the findings of the mapping study, multivocal literature review and multiple case studies of cases E–I, DevOps practices are observed from organisational and technical perspectives. From an organisational perspective, the most common DevOps practice was reorientation of responsibilities between software development and operations, wherein the software development team was gaining more responsibility for operations and operations existed as a small team, offering their support to the software development team. From a technical perspective, DevOps practices were manifested in building automation in the software delivery pipeline. The findings of DevOps practices were briefly presented in Paper I and Paper III, and explained in more detail in Paper V. In the multiple case study of Paper V, it was observed that software developers, in addition to their usual tasks of developing software, took on the responsibility to specify, create
and modify environments and their configuration options when deploying software changes across the different environments. The deployment pipeline constituted the practice of automating the software deployment, managing software infrastructure using configuration management tools and infrastructure-as-code practice as well as continuously monitoring software across different environments in the deployment pipeline as an activity that is performed by software developers. The latter involved incorporating other supporting software development practices, such as continuous integration, quality assurance practices of code reviews and test automation, in which agile and lean practices were also observed.

Answers to thesis RQ3: What benefits and challenges of DevOps are perceived in software development practice?

Several benefits, such as improved speed in release cycle times, and challenges, such as high skill requirements for software developers, are identified in the multivocal study of Paper II. In multiple case studies, in-depth descriptions are given particularly of the rationale of the benefits and challenges. The perceived benefits of DevOps are largely captured in the drivers of DevOps adoption in the multiple case study of Paper V. The findings of Papers IV and V reveal that DevOps in some contexts is difficult to adopt. Furthermore, within the cloud domain, it is adopted by software companies of different sizes, ranging from small to large companies. For companies that apply DevOps, challenges are observed in difficulties in infrastructure management, particularly in automating the network component in a private cloud infrastructure, as well as the high skill demand for software developers when performing operations tasks. In the embedded systems domain, DevOps is challenging to adopt due to reasons such as a lack of multidisciplinary teams impacting the practice of continuous integration, and a lack of visibility to customer environments that can be used to configure the test environment.
Table 11. Summary of thesis findings.

<table>
<thead>
<tr>
<th>Thesis RQ</th>
<th>Findings</th>
</tr>
</thead>
</table>
| RQ1. What is the understanding of the DevOps concept in software development practice? | – DevOps is a cultural change substantiated with a set of practices to encourage cross-disciplinary collaboration between software development and operations within a software company. The main purpose is to enable the fast release of quality software changes while simultaneously operating the system reliably and resiliently.  
– DevOps is a phenomenon that was observed to originate from continuous deployment as an evolution of agile software development and is informed by principles of lean in the software development process |
| RQ2. What practices are found in DevOps implementation in software development practice? | – The common organisational DevOps practice is reorientation of responsibilities between software development and operations, wherein the software development team gains more responsibility for operations, and operations offers their support to the software development team in tasks, such as building the deployment pipeline  
– The common technical DevOps practices focused on automating the software deployment process, automating the management of software infrastructure using configuration management tools and the infrastructure-as-code practice and continuously monitoring software across the different environments by software developers. These practices incorporate other software development practices such as continuous integration, quality assurance practices of code reviews and test automation, as well as agile and lean practices |
| RQ3. What benefits and challenges of DevOps are perceived in software development practice? | – The common perceived benefits of DevOps adoption and implementation are: improvement in release cycle time and continuous deployment of software changes; productivity of operations work and improved work morale, knowledge and skills  
– The common perceived challenges of DevOps implementation are: resource constraints; insufficiencies in the automation of infrastructure management; high demands for required skills and knowledge; and difficulties in monitoring microservices. DevOps in the embedded systems domain is challenging for reasons such as: hardware dependency, which challenged the creation of multidisciplinary teams; limited visibility to the customer environment when configuring test environments that challenged the provision of representative test environments of the target environment; and the lack of technology to automate deployment of new software features repeatedly and reliably fast in customer-specific environments |
5 Discussion

In this section, the findings of the thesis are discussed in relation to the related work presented in Chapter 2. In addition, the main implications of the findings for practitioners and research are presented in Section 5.2 and Section 5.3, respectively.

5.1 Thesis findings in relation to related work

Early studies reported the existence of very few studies in the area of software development and operations collaboration, despite it being a very important area for research (Nelson et al. 2000, Tessem & Iden 2007, 2008). This, however, changed significantly starting in 2010, whereby more studies started to appear in scientific publication venues following the inception of the DevOps concept.

Similar to other previous studies (Bass et al. 2015, Dyck et al. 2015, Jabbari et al. 2016, Penners & Dyck 2015, Smeds et al. 2015), this thesis explored the understanding of DevOps given the many controversies that surrounded the term. The disadvantage in many of the DevOps definitions proposed in the literature was that their descriptions did not include the viewpoints of many different software practitioners. The proposed definition that included at least a dozen viewpoints of both scholars and software practitioners, including the pioneers of DevOps, was that provided by Penners and Dyck (2015). However, Penners & Dyck (2015) inquired for other scholars to conduct additional inquiries from software practitioners in order to improve further their definition.

This thesis explored the understanding of DevOps using three different research approaches that incorporated the views of several software practitioners. In answering the thesis’ research question, what is the understanding of DevOps in software development practice, the proposed DevOps scientific definition by Penners and Dyck (2015) was enhanced to refer DevOps concept as ‘a mindset substantiated with a set of practices to encourage cross-functional collaboration between teams —especially development and IT operations —within a software development organisation, in order to operate resilient systems and accelerate the delivery of changes’. The latter definition is presented in thesis Paper III. The enhanced definition is similar to that by de França et al. (2016) and is broad enough not to exclude its various interpretations found in the existing literature (Bass et al. 2015, Jabbari et al. 2016, Smeds et al. 2015). However, the presented
definition is more detailed and specific while it also discards other interpretations, such as DevOps being a development methodology, as presented by Jabbari et al. (2016). The discard is given because the findings of the thesis that are supported by de França et al. (2016) did not reveal the existence of a systematic approach to performing DevOps.

Furthermore, this thesis conceptually compared DevOps with established software development methods of agile and lean. As a summary, it was observed that the DevOps concept extends agile principles to stakeholders beyond the software development to operations while it is informed by principles of lean in software development. Existing literature on agile software development (Abrahamsson et al. 2003) reported that most agile methods do not cover the system in use (operations) life-cycle phase. Agile methods are more focused on the different life-cycle phases of software development, specifically requirement specifications, design, code, unit test, integration test and system test phases (Abrahamsson et al. 2003). This observation by the authors (Abrahamsson et al. 2003) is in support of the findings that DevOps extends agile principles and practices to operations. As such, lacking to use agile principles and practices in operations is reported in several empirical studies as causing several problems, such as delays in the software release process (Gotel & Leip 2007, Iden et al. 2011). As noted by Iden et al. (2011), improvement initiatives of software development practices are limited to either the software development domain or the operations domain. Issues that concern both domains are seldom jointly addressed (Iden et al. 2011). Findings from empirical studies of the thesis, especially Paper V, have shown that this perception of isolating software development and operations is changing through awareness and introduction of the DevOps concept and practices into software-intensive organisations.

From empirical studies of the thesis, Paper IV and V, the findings of DevOps adoption and implementation in software-intensive organisations extend those of existing literature, Cito et al. (2015), Balalaie et al. (2016), Schneider (2016), Elberzhager et al. (2017), Chen (2017) and Shahin et al. (2017b), by providing additional evidence from nine different software development contexts and detailed descriptions of the different aspects of DevOps practices, benefits and challenges. From the nine different development contexts, the adoption and implementation of DevOps was observed to be successful in five cases but challenging in four cases.

Regarding the implementation of DevOps, the practice of reorientation of responsibilities between software development and operations by giving more responsibility for the operations activities to software developers was more prominent in the findings in Paper V of the thesis. This practice took place while operations existed as a small group.
The findings is similar to empirical studies reported by Shahin et al. (2017b) and Cito et al. (2015). However, the practice was largely observed possible due not only to the size of the organisation and the type of software application being developed but also the supporting culture and attitude of personnel in software development and operations teams. In larger organisations, particularly in the embedded systems domain, operations were observed to have less collaboration with software development. In such a context, a trialled practice to include a person from operations in one software development team proved to be unsuccessful because of the lacking culture and mindset advocated in DevOps for both teams.

DevOps implementation from a technical perspective, observed through the deployment pipeline of empirical study Paper V, gave evidence and detailed descriptions about automation mechanisms used in software deployment, practices in infrastructure-as-code and software monitoring practices used by software developers. In the existing literature, very few empirical studies provide detailed descriptions of activities and practices used to implement DevOps through the deployment pipeline (Callanan & Spillane 2016). Often, details such as adopted software deployment strategy and mechanism, are not reported in the studies (Cito et al. 2015, Elberzhager et al. 2017, Fazal-Baqiae et al. 2017, Shahin et al. 2017b). The availability of such information is useful to explain about some observed benefits or challenges. As an example, in Paper V of the thesis, it was observed that the blue-green deployment strategy in one case involved the process of baking a VM image, a step which had some observable effects to the deployment time of software to production.

Furthermore, detailed descriptions in the thesis findings explain the tasks required for the DevOps engineer’s role as reported in job postings (Hussain et al. 2017, Kerzazi & Adams 2016). Kerzazi & Adams (2016) and Hussain et al. (2017) report on deployment pipeline optimisation as well as defining and managing the infrastructure as important activities for DevOps engineers. From the cases in Paper V, continuous monitoring was an activity that was observed to be done increasingly by the software development team in addition to the operations team. Monitoring software was observed to be of interest to software developers, particularly during the deployment of software changes and in keeping services up and running in production. In all cases, the practice of ‘breaking things in production’ as in Netflix (Basiri et al. 2016) so as to design services with high resilience was not observed. In fact, software developers were struggling to determine the type of data to monitor, as often interesting metrics to monitor were found after a failure had occurred.
Meanwhile, from the findings of Paper V, agile and lean practices of small and incremental development and continuous integration, trunk-based development and visualising the flow of work using a Kanban board were observed to be in use. Quality assurance practices, including change-based code reviews and automated tests executed by a continuous integration system, were also employed. The details of the latter practices showed some variations across the organisations. For example, in code reviews, differences were observed when the reviews were conducted, such as either before or after merging code to the mainline. Similarly, differences were observed in the details of continuous integration practices, such as the types of tests executed and the build packages, similar to findings about the practice of continuous integration in industry (Ståhl & Bosch 2014). The information of agile and lean practices provide some preliminary support to the propositions provided in the findings of the multivocal study of Paper II. The proposition was that agile principles and practices are necessary for the successful adoption of DevOps. This is also supported by Elberzhager et al. (2017), who saw their existing agile practices and quality assurance procedures as a good starting point for their DevOps adoption and implementation.

Generally, the findings about DevOps implementation are reported in the existing literature on best practices for DevOps (Bass et al. 2015) and the design of operationally friendly software systems (Hamilton 2007). However, there is still a lack of evidence of several other recommended best practices, such as software engineers’ need to understand network design, which can be reviewed together with networking specialists in operations teams (Hamilton 2007). More important than this, the automation of the network was perceived as a difficult task by software developers in one case of Paper V of the thesis. As such, the detailed descriptions presented in multiple case studies that specify the development context in detail help provide explain and rationalise the implementation, or lack thereof, of certain practices in different cases.

Most of the perceived benefits of DevOps adoption and implementation, such as improved speed and quality, presented in the thesis findings are similar to the existing literature of DevOps (Callanan & Spillane 2016, Elberzhager et al. 2017). In addition, the challenges of DevOps adoption, such as the lack of management support and resistance from operations personnel reported in the existing literature (Jones et al. 2016) were observed in some cases of the thesis. However, the latter was not a problem, especially for the five cases that had successfully adopted DevOps because of the supporting culture and positive attitude amongst employees in software development and operations teams. The perceived challenges of DevOps practices from the multiple
case studies that had applied DevOps were the high demand for software developers to possess the skills and knowledge to perform operations tasks. It was observed that software developers often specified that operations tasks involve a different competency. In a study by Jones et al. (2016), the operations team had claimed that not having overlapping tasks with software developers served as one source of resistance from operations staff. However, in our cases, the developers reported the existence of an overlap but only that it demanded a new competency – alternatively being open-minded to input from operations experts – than their typical task of writing software code. This as a result would constitute a multi-disciplinary software development team. Furthermore, full automation of the infrastructure is perceived as a challenging and complex task, even with the existence of vast infrastructure management tools due to legacy technologies that still need to be maintained. The latter finding was also recently reported by Chen (2017).

Before the publication of Paper IV, no study existed on DevOps adoption in industries with stringent requirements for safety, reliability and availability, such as embedded systems domain. The finding was that in such a domain, DevOps was hard to adopt and implement in the studied cases. A recent study presented a similar finding that DevOps is difficult to adopt in such industries with strict standards and regulations (Laukkarinen et al. 2017). However, the finding of the reasons why DevOps is challenging in an embedded systems domain is not new in the existing literature covering cross-discipline collaboration of software development in this domain. In particular, Li et al. (2017) and Verhoef et al. (2014) noted that the lack of cross-discipline design interaction is a dominant problem in the embedded systems domain, and is often a root cause of numerous problems that occur during the system integration phase. Further, Verhoef et al. (2014) reported that it is a root cause of integration problems because system-level requirements, such as performance, cannot be assigned to a single discipline, since the responsibility to meet such a requirement is shared across all disciplines. The authors contend that even though making cross-discipline teams can be difficult to achieve – owing to such factors as using different languages and attending to different problems by engineers across the disciplines – cross-discipline communication is mandatory (Verhoef et al. 2014). Additionally, regarding the challenge related to lacking feature usage data in system performance data, Olsson and Bosch (2014) also report on the limited usage of post-deployment data in the embedded systems domain.
5.2 Implications for practice

The findings have the following implications for software practitioners interested in DevOps. First, a key implication for software developers and operations engineers is the need to better understand the concept of DevOps and its practices – from both organisational and socio-technical perspectives – that are appropriate for their context prior to embarking on the DevOps adoption and implementation journey. It was found that DevOps is a concept that includes a change of culture and mindset, which are aspects that are not technically engineered and may take a long time to achieve. Furthermore, the value of DevOps may not even be recognised simply by bringing software developers and operations together into one team without them fully understanding and agreeing on the implications of their collaboration. It was observed in the findings of the thesis and reported in the existing literature that the two teams have different competencies, and as such, there must be a mutual understanding and a willingness to share and learn competencies from each other. In addition, an understanding of the DevOps concept would clarify the skills that are required, which in turn would facilitate better employee training and hiring given the increasing job adverts for DevOps engineer (Hussain et al. 2017, Kerzazi & Adams 2016).

Second, as much as DevOps is about cultural and mindset change, automation of activities in software development and delivery play a major role in DevOps. Automation serves as: (i) a control and Standardisation mechanism of quality software development and delivery process; (ii) an approach to remove or reduce human-errors that are associated with the hand-over and bureaucratic process in the release of software changes to production; and (iii) a process to manage and document activities, including infrastructure provisioning and configuration management. As a result, this reduces the release cycle time of software changes, which leads to fast feedback to software developers and other stakeholders regarding the quality and impact of the software changes.

Third, efficiency that comes through automation comes at the cost of additional resource requirements, particularly in terms of the time and effort put in by software developers to learn about the tools and competencies, in addition to the efforts required for test automation. It was observed in a consulting project that automating tests were feasible, but to a customer of the project, it was a cost that needed to be taken into consideration, especially when the automation gets closer to the user interface. At
times, the customer was not willing to pay for the cost, thus impacting the quality of the system.

5.3 Implications for research

The findings have the following implications for research. First, the descriptions briefly presented in the multivocal literature review studies and in-depth in the multiple case studies give clarity and evidence of how DevOps is understood and practiced by software practitioners. Unlike practitioner-led survey reports that do not employ rigorous research logics, the studies included in the thesis applied and followed recommended steps for conducting research in empirical software engineering to provide evidence and in-depth explanations about the adoption of DevOps. The studies explored the DevOps phenomenon starting with what the concept meant, and then to its key attributes, practices, benefits and challenges, thus providing the benefit to other researchers for a phenomenon where little knowledge is available. Despite the fact that a definitive description of the concept of DevOps and practices cannot be reached nor enforced in the community, an attempt has been made to make such knowledge explicit that other researchers can build upon. This knowledge is believed to be helpful in future works for criticism, strengthening or generating new insights (see Section 6.3).

Second, guidelines for performing a multivocal literature review need to be established in software engineering methods. The work for this is underway (Garousi et al. 2016). In this thesis, through a multivocal literature review, several DevOps practices were identified, some of which were supported in multiple case studies of the thesis. The value of the multivocal literature review in the thesis was not only to identify DevOps practices but also to define the concept of DevOps from a large number of practitioners. Guidelines for conducting multivocal studies should pay attention to the rigorous steps for their data extraction.
6 Conclusion

This thesis presents findings of DevOps adoption and implementation in software development practices, and concerns the conceptual understanding of DevOps, its practices, benefits and challenges. The descriptions of these aspects of DevOps are briefly identified in multivocal studies and explained in-depth in the multiple case studies performed during the thesis research. The contribution is a rich analysis of the studied cases, showing how software practitioners understood and implemented DevOps practices that fit their development context in order to achieve its benefits, such as improved speed in the delivery of software changes. The key contributions of the thesis are explained in the following section followed by a validity discussion and identified areas for future research.

6.1 Summary of the main contributions

Software practitioners and researchers seeking to explore the DevOps phenomenon can learn from this thesis about the meaning of the DevOps concept, its practices, and benefits and challenges, which are explained in detail. The main contributions are as follows.

– The first contribution of the thesis is the definition of DevOps and its main characterising elements. The thesis explores the definition of DevOps from the existing literature and software practitioners. Based on the findings, the definition by Penners and Dyck (2015) is enhanced by incorporating the different views of software practitioners’ understanding of the concept. As such, DevOps in the literature and amongst software practitioners can best be understood as: ‘a mindset change substantiated with a set of practices to encourage cross-functional collaboration between teams – especially development and IT operations – within a software development organisation, in order to operate resilient systems and accelerate the delivery of changes’

– The second contribution of the thesis is descriptions of the adoption and implementation of DevOps in different contexts. DevOps practices are grouped into two categories: organisational and socio-technical. DevOps from an organisational perspective manifests in different forms of riorienting responsibilities between software development and operations teams; and from a socio-technical perspective, it focuses
on automation in the software delivery process. Increasing responsibility on software
developers and operations taking a supporting role is the most common organisational
manifestation of DevOps in practice, particularly for cloud-based systems. From a
socio-technical perspective, the most common is increased automation for deployment
of system changes and infrastructure management through the practice of IaC.

– The third contribution of the thesis is about the benefits and challenges of DevOps
adoption and implementation. Through DevOps, software practitioners perceive
benefits of improvements in the speed of delivering software changes, the quality
of the system, collaboration and productivity amongst stakeholders in software
development and operations teams. However, it was observed that the benefits are not
just as a result of DevOps, but by also applying supporting practices and principles of
agile and lean software development, code review, test automation in quality assurance,
etc. DevOps is no silver bullet, since challenges such as the accumulation of technical
debt and difficulties in achieving full automation in infrastructure management were
still observed in the findings of the thesis.

6.2 Validity of the findings and limitations of the research

This thesis acknowledges several threats to the validity of the study findings against
which several strategies were employed. The thesis fulfils the thick descriptive purpose
of qualitative research, which centres around explaining the meaning and perspectives
constructed by individuals, groups or both in a particular context, rather than causal
relationships (Runeson & Höst 2009). Threats to the validity of the findings are found in
construct validity, external validity and reliability. The validity threats and strategies
used in the studies of the thesis are explained in the following paragraphs.

6.2.1 Construct validity

Construct validity is concerned with the extent to which operational measures that
were used in the thesis represent what is investigated according to research questions
(Runeson & Höst 2009). In the thesis, construct validity deals with the processes of
identifying and extracting data from primary studies and multivocal documents as well
as their publication bias. In addition, from multiple case studies, construct validity
deals with whether the interview questions were interpreted as intended as well as the
selection of cases and interviewees.
For the systematic mapping study (Paper I), in addition to conducting several pilots when formulating the search string as well as the inclusion and exclusion criteria, the strategy that was employed was not to incorporate a very precise search string, which in turn resulted in the retrieval of several articles. Similarly, the multivocal literature review employed a broad search of understanding DevOps phenomenon rather than a specific aspect of DevOps. Furthermore, for the data extraction of primary studies, each primary study was given a score of rigour and industrial relevance based on the model provided by Ivarsson and Gorschek (2011). For the multivocal literature review studies, based on recommendations by Ogawa and Malen (1991), two strategies for dealing with publication bias were employed. The first one was consulting other forms of data sources during data collection, such as interviews and workshop informants. The second strategy was to determine the position and certainty of the author during analysis. This involved classifying each multivocal document with additional attributes, such as the role and affiliation of the author. For the interviews, in addition to the development of an interview guide with several researchers, pilot interviews were performed in order to improve the interview guide. Furthermore, prior to the interviews, the interviewees were told of the objective of the research and structure of the interview. For the selection of cases in the multiple case studies, the first set of cases used in Paper IV were selected as a result of the intention to replicate another study conducted by different researchers. As such, the researchers of the original study collaborated with the author of the thesis as well as other researchers and gave more details about the cases included in the original study, and similar cases were then selected. The second set of cases for Paper V were selected on the basis of the findings of IV, which had revealed challenges to the adoption of DevOps. As a result, the author of the thesis wanted to investigate cases that had successfully adopted DevOps in their context. This intention was made known to the software practitioners. In all cases, different roles were interviewed to a point of saturation, which was a point where no new viewpoints were emerging by adding more interviewees.

6.2.2 External validity

External validity is concerned with the extent to which it is possible to generalise the findings of the thesis, and the extent to which the findings are of interest to other cases (Runeson & Höst 2009).
Triangulation in terms of collecting evidence by using different methods is one external validity strategy used in the thesis (Runeson & Höst 2009). The use of different methods helped to consult different sources of data in order to generate the findings of the thesis. For each research method employed, explicitly available guidelines were followed, such as those by Kitchenham and Charters (2007) and Petersen et al. (2008) for the systematic mapping study, and for the multiple case studies, by Runeson and Höst (2009). For the multivocal literature review, triangulation at the source level was also done, and included data from interviews and workshops with software practitioners. Despite the use of method triangulation, the findings of the thesis cannot be generalised to the entire population. In fact, the findings are used to provide insights into the adoption and implementation of DevOps in software development in cloud and embedded systems domain contexts, which are limited by the contextual situations of the studied software companies.

6.2.3 Reliability

Reliability is concerned with the extent to which the data and the analysis are dependent on the author of the thesis (Runeson & Höst 2009). Examples of the threats to reliability are a lack of clarity of how data was coded or interviews being unclear.

The involvement of more than one researcher in research activities is another strategy that was used in the thesis to strengthen the reliability of the findings (Runeson & Höst 2009). The strategy was used to deal with validity threats related to researchers’ bias in the findings of the research, especially during the collection and analysis of the data. Researcher’s bias constitutes the different types of effects that the researcher may have on the study while conducting research activities. Examples of the effects include the thesis author’s influence on data sources and her subjectivity. The strength of the findings of the thesis reside in the fact that a number of researchers, including the author of the thesis, were involved in the research activities of the systematic mapping study and multiple case studies. The involvement of several researchers in the selection process of the primary studies and the development of the interview guide, including piloting of the inclusion-exclusion criteria as well as the interview guide, helped to minimise subjectivity. Furthermore, although in the multivocal literature studies care was taken to minimise the threats as only the author of the thesis was involved in data collection and analysis, the findings of the multivocal literature review were similar to
the findings of the study by de França et al. (2016), who used a similar method on the topic of DevOps.

Member checking is another strategy that was used in the thesis (Runeson & Höst 2009). The strategy relates to presenting the findings to the subjects for the purpose of including them in the research process and gaining their feedback on the conclusions. Software practitioners of the companies involved in the multiple case studies as well as the interviews in the multivocal literature studies were presented with the findings in workshops, company presentations and in DIMECC N4S project reviews. Two DevOps workshops were arranged by the author of the thesis together with other researchers, during which findings of Paper II and Paper V were presented. One company presentation was done by the author of the thesis to present the findings of Paper III. Findings of Paper I and Paper II were mostly presented in DIMECC N4S project reviews in the form of posters. In all the different activities, feedback was gathered and taken into consideration to improve the research activities.

6.3 Future research

The thesis by no means provides answers to all research questions pertaining to the DevOps phenomenon. For instance, many research questions raised by Adams & McIntosh (2016) remain unanswered. Furthermore, by extending the scope of the thesis, more areas of research remain uncovered. However, from the scope of the thesis and based on its findings, the following areas of research are solicited.

The author considers it important to carry out additional research, preferably with a large number of software practitioners, for instance, through a survey to investigate further the findings of the thesis. To date, the state-of-DevOps surveys are conducted annually by practitioners, and these receive much interest and have more visibility amongst software practitioners. Competing with such surveys might be difficult, but one approach is to conduct country-specific state-of-DevOps surveys as such findings may be more focused. Regardless, new questions for the surveys can be derived from the findings of the thesis that pertain to the definition, practices, benefits and challenges of DevOps. In addition, research that focuses on mining software repositories can be useful to give quantifiable results of the facts on deployment frequency, software quality, etc., which are not just based on the perceptions of software practitioners.

Based on findings of this thesis, future research can investigate changes that would be required to be made in the software development process to enable the adoption and
implementation of DevOps in the embedded systems domain. Alternatively, it would be interesting to investigate the impact of DevOps on specific roles, especially those in software development organisations who represent or interact with the customer. It was reported in the findings from the multiple case studies that the role of the product owner may become obsolete as software developers are gaining a better understanding of their systems and the customer usage behaviour of the systems in production. The product owners in one consulting company, where the role was represented by the client, reported that the responsibility of manually approving changes for deployment is not only time consuming but also at times unnecessary.

Generally, as a future outlook for DevOps adoption and implementation in the software industry, several issues remain unclear. Most of the DevOps initiatives are trialled in a small teams or in few number of software products and services. It is not yet clear how hard it will be to extend the practices across teams as well as in aligning DevOps with existing software development practices that are used at larger scale? In addition, how such practices will be used in implementing enterprise applications such as ERP systems.
References


# Appendix 1 Interview guide

## Introductory Questions
- Role and daily responsibilities in the organisation/team?
- Length of experience working in the organisation/team?

## Management overview of the development process
- How are development and operations activities organised?
- How many teams do you have? Sizes of the teams?
- How are responsibilities given/shared to the teams?
- How would you describe the case? Is it typical or specific in the organization?
- What kind of development process do you use?
- How are requirements and features specified, by whom?
- How are prioritization and decisions made?
- How are the features released to production, how often?
- How are the release decisions made? By whom?
- Why did you decide to implement DevOps?
- When did you start to implement DevOps?
- Who was/is the driver for the DevOps implementation?
- What does DevOps mean in your organization?
- What were the expected impacts (benefits and drawbacks)
- What kind of impacts do you see so far? Any hard data?
- What has been challenging in implementing DevOps?
- How far along are you now in implementing DevOps?
- What advice would you give to somebody who is considering implementing DevOps?
- Describe the case in more detail i.e., System/product; Teams involved; Use of DevOps?

## Software Development
- How is the project team organised?
- Who do you interact with the most? When? How?
- How are requirements determined and incorporated in product development process?
- How often do you release the software and for what types of software changes?
- Can you describe your working flows/practices until deployment and operational use in production (draw a picture and explain)?
- What tools are used to support development work?
- When do you consider an implementation to be done?
Software Integration and Deployment

- Describe in more detail at which stages software changes are integrated, the levels and process of testing software changes?
- What tools are used to support building and testing?
- What practices, other than testing, contribute to quality?
- What is the process of deploying/delivering to customers?
- How is quality assured in the deployment pipeline?
- What are the most important challenges regarding quality?

Infrastructure management and monitoring

- Who/team is responsible for managing, maintaining system infrastructure?
- Does development team have access to critical resources (servers, logs) in production?
- How are non-functional requirements and information about configurations communicated to developers?
- How do you provision and manage different environments of your application?
- How are software changes deployed in production?
- What monitoring is performed in production? For what purpose?

Perceived impacts of DevOps

- What does DevOps mean to you?
- What impacts have you seen after introducing DevOps? Do you have any data to support this?
- Considering current practices in this team, how do they differ compared to before introducing DevOps?
- What have been the main benefits of DevOps?
- Are there any drawbacks in using DevOps?
- What are the main challenges of using DevOps?

Culture and mind-set

- Is your company culture supporting the DevOps approach?
- How would you describe your development culture and mind-set at organisation and team?
- How is management support for the development practices and culture visible in your organisation?
Appendix 2 Summary of DevOps workshop with practitioners

<table>
<thead>
<tr>
<th>Purpose</th>
<th>To gather software practitioners’ rationale and mechanisms of DevOps adoption, implementation as well as benefit and metrics</th>
</tr>
</thead>
</table>
| Participants | – P1 senior software designer at consulting company-one  
– P2 senior software designer at consulting company-one  
– P3 team leader of support operations at IT software and service company-two  
– P4 senior consultant/software architect at consulting company-three  
– P5 software engineer at consulting company-four  
– P6 team lead at embedded system domain telecom company-five  
– P7 system architect at embedded system telecom company-five  
– P8 senior design engineer at embedded system automotive company-six  
– P9 head of quality at embedded system automotive company-six  
– P10 software architect at consulting company-seven  
– P11 head of continuous services at consulting company-seven |
| Working groups | Team A (P1, P2, P3); Team B (P4, P5); Team C (P6, P7); Team D (P8, P9, P10, P11) |

| Why will/have you implemented DevOps at company? | – To remove silos and barriers between Dev team and Ops/Infra team (Team A, D)  
– To share and exchange knowledge between Dev team and Ops (Team A)  
– To enable developer implement and deploy changes faster (Team B)  
– Developers wanting to enhance their competency (Team B, P5)  
– To reduce and save associated costs (Team C)  
– To have better visibility to customer requirement (Team C)  
– To improve collaborations across disciplines and teams (Team C)  
– For efficient ways to update a running software in production (Team D, P10)  
– To deal with increased complexity in software and environment (Team D) |

| Why have you not implemented DevOps at company? | – Company is not using DevOps principles due to the challenges related to network integration and verification caused by large scale and the multiplicity of different organisations, as such this is an existing challenge not considering operations yet (Team C, P5)  
– Limited connection to customer (Team C, P6)  
– Do not only focus on software rather telecom networks which involve hardware (Team C, P6) |
### How will/have you implemented DevOps at company?

- Dev and Ops paring to form one team (Team A)
- Employing infrastructure-as-code practice (Team A, B, D)
- Early incorporation of Ops in Dev projects (Team A)
- Increase competence of Ops in Dev teams especially in outsourced Operations (Team A, P3)
- Need to increase the level of automation in testing, documenting procedure for configuring and monitoring services (Team C)
- Through organisation structural changes e.g. incorporation of feature teams (Team C)

### What has been the most important impact of DevOps in your organisation?

- Lowered stress levels (Team A)
- Faster time to production (Team A, D)
- Development is more cost-effective (Team A)
- Improved product quality (Team A)
- Close collaboration between Dev and Ops (Team A, D)
- Efficiency in operational tasks (Team D)

### What suitable metrics to be used to assess DevOps implementation?

- Customer satisfaction (Team D)
- Release cycle time (Team D)
- Quality of software product by measuring code quality, performance metrics, customer satisfaction, customer usage (Team B)
Original publications


Reprinted with permission from Elsevier (I), Springer (II), IARIA (III), and IEEE (IV).

Original publications are not included in the electronic version of the dissertation.
686. Taušan, Nebojša (2016) Choreography modeling in embedded systems domain

687. Yläne, Henri (2017) Herbivory control over tundra carbon storage under climate change


689. Singh, Sujeet Kumar (2017) Conservation genetics of the Bengal tiger (Panthera tigris tigris) in India


691. Hens, Hilde (2017) Population genetics and population ecology in management of endangered species


694. Schneider, Laura (2017) Mechanocatalytic pretreatment of lignocellulosic barley straw to reducing sugars


697. Havia, Johanna (2017) Trace element analysis of humus-rich natural water samples: Method development for UV-LED assisted photocatalytic sample preparation and hydride generation ICP-MS analysis

698. Dong, Yue (2017) Bifunctionalised pretreatment of lignocellulosic biomass into reducing sugars: Use of ionic liquids and acid-catalysed mechanical approach

699. Leinonen, Marko (2017) On various irrationality measures


Book orders:
Granum: Virtual book store
http://granum.uta.fi/granum/
Lucy Ellen Lwakatare

DEVOPS ADOPTION AND IMPLEMENTATION IN SOFTWARE DEVELOPMENT PRACTICE

CONCEPT, PRACTICES, BENEFITS AND CHALLENGES

UNIVERSITY OF OULU GRADUATE SCHOOL;
UNIVERSITY OF OULU,
FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING